Wesleyan University

An Adjunction-Theoretic Foundation for Proof Search in Intuitionistic First-Order Categorical Logic Programming

By
Edward Morehouse
Faculty Advisor: Dr. James Lipton

A Dissertation submitted to the Faculty of Wesleyan University
in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Middletown, Connecticut April 2013

To my parents, without whose love and support this would not have been possible.

ACKNOWLEDGEMENTS

My path to this thesis has been longer and harder than I could have imagined when I set out. But as with any journey of discovery, nor could I have imagined all the wonderful things that I would encounter along the way.

I'd like to thank my thesis advisor, Jim Lipton, for standing by me throughout the journey, and the other members of my committee, Norman Danner and Mark Hovey, for helping to steer me to its successful conclusion. I am grateful for the resources that they and the rest of the department have invested in me over the years. I feel privileged to have known my fellow travelers, the other graduate students with whom I overlapped during my time at Wesleyan, and especially my officemate and friend of many years, Weiwei Pan, whose curiosity, determination and kindness have inspired me to persevere even when the road ahead seemed too difficult to go on.

I would not have found my way out of the wilderness were it not for a number of fortuitous encounters. I owe a huge debt of thanks to Gianluca Amato and Francesca Scozzari for hosting me at the University of Pescara in 2012. The opportunity that visit provided me to attend conferences and meet people in my field throughout Europe was invaluable to the development of this work, as was Gianluca's uncanny ability to find counterexamples to my not always very well thought-through "theorems". I also owe much to the organizers and instructors of the Oregon Programming Languages Summer School, which has provided a great service to a generation of of budding theoretical computer scientists, and I hope will continue to do so in the future. It was during one of Frank Pfenning's lectures on proof theory there that I first became aware of the connection between derivation systems and adjunctions, which is the seed from which this thesis grew.

The path that lies before me now feels more uncertain than ever, but I am eager to see where it may lead. Life is, after all, all about the journey.

Abstract

In this thesis we compose a proof-theoretic approach to logic programming with a category-theoretic approach to proof theory. This allows us to present the computation mechanisms of several systems of logic programming as proof search strategies within an intuitionistic first-order sequent calculus with logic variables and to analyze aspects of their behavior algebraically.

Beginning from the basic categorical construction of an adjunction, we present Gentzen's formal derivation system of natural deduction for intuitionistic first-order logic as a categorical graphical language, in which equivalence classes of derivations under the usual relations of convertibility correspond precisely to arrows in a freely-generated hyperdoctrine categorical semantics. We show that the inference rules and conversion relations on derivations have uniform adjoint-theoretic interpretations, and that the connectives are naturally partitioned into two chiral classes, depending on whether they are characterized by a right or a left adjoint functor. We observe that the adjoint-theoretic descriptions of the non-invertible rules for quantifiers each decompose their natural deduction counterpart into a purely logical rule and a substitution.

By using the same categorical semantics for the formal derivation system of sequent calculus, we are able present a version in which the generic parameters, logic variables and substitutions of logic programming are given first-class status. This in turn allows us to present the operational semantics of SLD-resolution for Horn logic (the logic underlying the programming language Prolog), and that of uniform proof for hereditarily Harrop logic (the first-order fragment of the logic underlying the programming language λ-Prolog), as search strategies in this sequent calculus. We show that the adjoint-theoretic description of the connectives permits a natural extension of the strategy of uniform proof to a syntactically richer language, which we call the language of constructive sequents.

We adopt Andreoli's idea of focused proof search from linear logic as a search strategy for intuitionistic logic within our sequent system, where it becomes a natural generalization of the strategy of uniform proof. We show that one of the two principles underlying focusing is immediately justified by the adjoint-theoretic properties of the connectives. We compare this focused proof search strategy with a different one proposed by Dyckhoff and Pinto in the context of the input-output semantics of logic programming.

Contents

Chapter 1

Introduction

This thesis straddles the fields of categorical logic and logic programming, placing it somewhere in the synthetic field of "categorical logic programming", as it were. It does this by composing a proof-theoretic approach to logic programming with a category-theoretic approach to proof theory. In themselves, neither of these perspectives is novel, but we believe that the composition permits a useful, algebraic, point of view to logic programming that has been under-appreciated and thus underutilized.

The conceptual heart of this work is the idea that we can reflect the categorical semantics of intuitionistic first-order logic embodied in the structure of a hyperdoctrine (a kind of indexed bicartesian closed category) back to the proof theory of Gentzen systems (natural deduction and sequent calculus). In a hyperdoctrine, each universal construction interpreting a logical connective is characterized by an adjunction. By uniformly translating the constituent parts of these adjunctions into inference rules, we obtain systems of formal derivation very close to the familiar Gentzen systems. However, crucially, the non-invertible quantifier rules thus obtained decompose their Gentzen counterparts into a purely logical rule and a substitution. By adopting the versions of these rules derived from the categorical semantics, we are able to present a sequent calculus for logic programming in which generic parameters, logic variables and substitutions are first-class citizens; that is, part of the object-theory rather than the meta-theory of logic programming. Therefore, this work can be seen as a contribution to the abstract syntax of logic programming, since it gives a declarative, algebraic accounting of formerly meta-logical operations.

1.1 Context

Logic programming has its roots in classical clausal logic, where the resolution method is used to show that a conjecture is a consequence of a set of assumptions by proving the unsatisfiability of the negation of the conjecture together with those assumptions [Llo84].

Unfortunately, this technique relies on the model theory of classical clausal logic and does not have a direct extension to other logical systems. However, there is a classically equivalent presentation of clausal logic, formulated in terms of implication and positive literals, which affords better opportunities for computational and algebraic interpretation.

Logic programming systems based on classical clausal logic, such as Prolog, must resort to extra-logical facilities in order to provide features typically found in high-level programming languages. Investigating this phenomenon, Dale Miller observed that some of these features, such as memoization, modules, abstract datatypes and local parameters, can indeed be given a purely logical description – but in intuitionistic, rather than classical logic [Mil89a; Mil89b]. In hindsight, the appeal of intuitionistic logic in applications of automated deduction should be apparent. After all, intuitionism was conceived by Brouwer as the logic of constructive mathematics.

The transition from classical to intuitionistic logic suggests the transition from a model-theoretic to a proof-theoretic perspective as well. It had long been argued by Dyckhoff, Pfenning, Miller and others that proof theory, rather than model theory, provides the best foundations for logic programming. There are deep connections between intuitionistic logic, in the form of natural deduction, and computation, in the form of the typed λ-calculus, described by the *Curry-Howard correspondence* [How80]. This provides a natural congruence relation on derivations, allowing us to determine when two syntactically distinct proofs are "essentially the same". Some such relation on derivations is essential to a proof-theoretic foundation for logic programming since searching the space of all syntactically distinct proofs is generally neither feasible nor desireable. That space tends to be quite large and to contain considerable redundancy in the form of distinct proofs with the same input-output behavior. It has been argued that normal natural deduction derivations – corresponding to normal typed λ-terms – should be considered to be the "real proof objects" [GLT89; DP99].

Categorical logic effectively erases the distinction between proof theory and model theory. Proof theory becomes equivalent to an initial model. While in the category of sets, first-order theories do not necessarily have initial models, given an appropriate category, this is no longer the case. Categorical approaches to proof theory are nothing new, going back at least to the work of Lawvere [Law69], Lambek [Lam68], Seely [See83] and Makkai [Mak93b], among others. However, categorical proof theory continues to be an active area of research, especially with respect to dependently typed intuitionistic logics underlying automated proof assistants such as Coq, realizability models for classical logics, and linear logic with its deep connections to quantum mechanics [BS09].

Just as a set theorist might ask, why study proofs when we have models, a proof theorist might ask, why study proofs categorically rather than as trees, tableaux, nets, etc.. To this, the best response we have heard comes from Paul-André Melliès [Mel09]:

> In this atomic vision of logic, proof theory becomes a linguistic laboratory, where one studies the logical connectives defined by tradition, and tries to decompose them as molecules of elementary particles[...]. This quest is driven by the hypothesis that these basic particles of logic should be regulated by purely algebraic principles, capturing the essence of language and interactive behaviors. Seen from this angle, categorical semantics becomes the cornerstone of proof theory, extracting it gradually from its idiosyncratic language (sequent calculus, etc.) and offering a promising bridge with contemporary algebra.

We believe that by exposing this algebraic structure behind the syntax of formalized proof, the categorical perspective has much to offer the study of proof search, and by extension, logic programming.

1.2 Methodology and Results

In this thesis we tame the great potential complexity involved in categorical approaches to proof theory in four basic ways. First, we limit our attention to typed intuitionistic first-order logic. Next, we take the categorical notion of adjunction as a starting point for interpreting Gentzen's systems of natural deduction and sequent calculus. Third, we consider derivations modulo $\beta\eta$-equivalence, thereby keeping the category theory one-dimensional. Finally, we consider a sequent calculus very close to natural deduction, encapsulating the complications introduced by more exotic sequent systems within a notion of "strategy". These choices allow us to present a simple, flexible and extensible framework for the categorical treatment of proof search and logic programming.

As a result, we are able to present a novel "indexed" sequent calculus, in which the metatheoretic notions of generic parameter, logic variable and substitution are given first-class status. We use this indexed sequent calculus to present the logic programming computation mechanisms of SLD-resolution and uniform proof as simple search strategies and observe that the strategy of uniform proof is complete for an even larger language, which we call "constructive sequents". Additionally, we apply Andreoli's principles of focusing [And92] to this system to present a complete search strategy for full intuitionistic first-order logic. This strategy has a much smaller search space than that of the naïve strategy of building all possible derivation trees.

The foundation for all of this is a new perspective on the classification of the inference rules of Gentzen systems. Rather than the introduction–elimination or right–left dichotomy typical in proof theory, we present a classification based on the concept of connective chirality, which only becomes apparent when connectives are seen from the perspective of their adjoint-theoretic characterizations. From this perspective, the inference rules and derivation

conversions of natural deduction can be formulated uniformly in terms of the adjunctions characterizing the connectives. This adjoint-theoretic description of derivations extends to sequent calculus as well. In both systems, the adjoint-theoretic descriptions of the non-invertible quantifier rules decompose their traditional versions into a purely logical rule and a substitution. It is by reflecting this decomposition back into proof theory that we obtain our indexed sequent calculus.

From the adjoint-theoretic perspective, issues such as the harmony of connectives become quite straightforward, proof search strategies that eagerly apply natural isomorphisms become obvious, and much else follows. We believe that the methods and results of this thesis are only examples of what may be achieved by composing a categorical perspective of proof theory with a proof-theoretic perspective of logic programming.

1.3 Outline

A brief outline of the rest of this thesis follows.

In chapter 2 we introduce Gentzen's derivation system of natural deduction, including the concept of connective harmony and the permutation conversions. Together these account for many of the system's proof-theoretic properties. Among these properties, the unique normalization of derivations is of particular importance as it provides a congruence relation on derivations, the equivalence classes of which will coincide with the arrows in our categorical semantics.

In chapter 3 we summarize some of the categorical constructions used in the sequel. Most important among these is the concept of adjunction, which we will use to interpret all of the connectives of intuitionistic first-order logic. The closely-related notion of comonad is used to give a clear and modular interpretation to the property of this logic of being cartesian, as opposed to linear, which loosely means that it is free from considerations of resource accounting. The brief introduction to bicartesian closed categories summarizes the categorical constructions that will be used to interpret intuitionistic propositional logic (without quantifiers). Indexed categories, also introduced in this chapter, will be used to provide interpretations for substitution and quantification.

In chapter 4 we present the hyperdoctrine interpretation of intuitionistic first-order logic. This is based on an indexed category first described by William Lawvere that has facilities for interpreting universal and existential quantifiers. The version of hyperdoctrine that we present differs slightly from the original concept since the logic we wish to interpret includes disjunction and falsehood, but not equality. We present two examples of the concept of hyperdoctrine, in the forms of the subset hyperdoctrine and the sieve hyperdoctrine, which

correspond respectively to the set-based semantics of classical model theory and a presheaf semantics commonly used in categorical logic.

Chapter 5 provides the foundation for the rest of the thesis. There, we show how each of the connectives of intuitionistic first-order logic may be understood to arise from an adjunction and that natural deduction inference rules and derivation conversions may be "read off" of this adjunction in a uniform way. This leads to the insight that, from one perspective at least, the primary difference between connectives governing their proof-theoretic behavior is whether they are characterized by a right or a left adjoint functor. This in turn leads us to the concept of connective chirality, which will be very useful for reasoning about proof search strategies in the sequel. We observe that the non-invertible inference rules of quantification derived from adjunctions decompose their traditional versions into a purely logical part and a substitution. Finally, in this chapter we demonstrate the genericity of freely-generated hyperdoctrines, which is what allows us to conduct proof theory in a categorical setting.

In the brief chapter 6 we consider the fact that the proof theory we have been considering is cartesian rather than linear, in the sense that assumptions may be duplicated and discarded at will without requiring an accounting. This property turns out to be a less than ideal fit for the categorical interpretation we have presented, necessitating some rather tedious "context shuffling". Fortunately, the categorical construction of comonad is able to encapsulate this, at the cost of interpreting our logic in a slightly different category. Two ways of doing this are discussed.

Chapter 7 introduces Gentzen's derivation system of sequent calculus and extends our categorical interpretation of the proof theory of intuitionistic first-order logic from chapter 5 to this system. Here, the connection between the adjoint-theoretic and proof-theoretic semantics is not quite as direct. In particular, the left rules for conjunction and implication are different from, though equivalent to, those we would get by a direct translation. We also see that, as in natural deduction, the non-invertible sequent rules for quantifiers incorporate a term instantiation that is not a part of the corresponding adjoint-theoretic description. By following the categorical semantics more faithfully, we decompose these rules into a purely logical part and a substitution part that need not be applied together. This leads us to the formulation of an indexed sequent calculus in which the context variables and reindexings of the hyperdoctrine interpretation have first-class status. It is revealed that these context variables interpret both the generic parameters and logic variables that arise in logic programming; the only difference being the parity of the chirality of the quantifier rule bringing a variable into scope with that of the quantifier. In this regard, there is seen to be a perfect symmetry between the two quantifiers.

Chapter 8 examines the logical systems of Horn logic and hereditarily Harrop logic used

in logic programming and shows how their computation mechanisms may be characterized as proof search strategies in the indexed sequent calculus of chapter 7. Central to this analysis is our new adjoint-theoretic understanding of the connectives, where some of their rules are interpreted as natural isomorphisms, which may be freely permuted and eagerly applied. We see that hereditarily Harrop logic has a natural syntactic extension, which we call the language of constructive sequents. Finally, we see that the principles of focused proof search, as formulated by Andreoli, give rise to a complete search strategy for all of intuitionistic first-order logic that subsumes the other systems we have considered.

We conclude by summarizing, surveying related and possible future work, and restating our central thesis, that the algebraic characterization of proof theory afforded by the categorical perspective provides novel insights useful in the study of proof search and logic programming.

Chapter 2

Harmony in Gentzen's Proof Theory

2.1 Derivations in Gentzen Systems

In the 1930s Gerhard Gentzen introduced two systems of formalized inference for proof theory [Gen35]. The first of these, called "natural deduction", is the subject of this chapter, while the second, known as "sequent calculus", is introduced in chapter 7. Intuitively, natural deduction may be thought of as a 1-dimensional proof theory, whose objects of study are essentially inferences between propositions. In contrast, sequent calculus may be though of as a 2-dimensional proof theory, whose objects of study are essentially inferences between inferences between propositions. Sequent Calculus was originally introduced to study the meta-theory of natural deduction, but was quickly recognized as a formalism important in its own right. These two formalisms, known together as "Gentzen systems", share a basic underlying structure. In Gentzen systems, derivations are graphical representations of formalized inferences taking the form of trees.

Gentzen systems are characterized by a set of **inference rules** (also called "rule figures"), which determine the primitive forms of reasoning available within the system. Each inference rule consists of a finite, possibly empty, set of **premises** and a single **conclusion**. A rule with no premises is an **axiom**. Intuitively, an inference rule expresses the idea that taken together, the premises are sufficient to justify the conclusion. Inference rules are represented graphically by writing the conclusion below the premises, separated by a rule line:

$$\frac{\text{premise}_1 \qquad \cdots \qquad \text{premise}_n}{\text{conclusion}} \text{ rule name}$$

If the conclusion is sufficient to justify the premises as well, then the rule is called **invertible**. Invertible rules are represented graphically with a double rule line. Variables occurring in inference rules are schematic, that is, implicitly universally quantified at the meta-logical level. We will refer to substitution instances of inference rules as **primitive inferences**,

although informally, these are often called "inference rules" as well.

Derivation trees are constructed by composing primitive inferences such that the conclusion of one is identical to a premise of another. Such a tree is a derivation of the the conclusion residing at its root, called the "goal" or **end-formula**. The premises at the leaves of a derivation are its **assumptions**, and the set of such is its **frontier**. A derivation with empty frontier is a **proof**. The base case in the inductive construction of derivation trees is that of an **identity derivation**, comprising no primitive inferences and thus having a singleton frontier containing, identically, the end-formula.

2.2 Natural Deduction

In the derivation system of **natural deduction**, each logical constant (henceforth **connective**, for short) is characterized by a set of introduction and elimination inference rules. Intuitively, **introduction rules** describe from what *evidence* a proposition with a given principal connective may be inferred. Thus the conclusion of an introduction rule for a connective is a proposition in which that connective is principal. Dually, **elimination rules** describe the *consequences* that may be inferred from a proposition with a given principal connective. Thus some premise of an elimination rule for a connective is a proposition in which that connective is principal. This premise is called the **major premise** of the rule, and any other (logical) premises are known as **minor premises**. By convention, the major premise of an elimination rule is written first, that is, on the left.

The conclusion of each natural deduction rule is a logical proposition, as are most of the premises. An exception arises in our treatment of this formal system because the logic that we wish to study is typed. This results in inference rules for quantifiers having premises that are typing judgements. A **typing judgement** is a judgement that a particular term in the language of individuals inhabits a particular type. Typing judgements, along with their means of inference, are parametric in the logic. In other words, we assume the existence of a type theory independent of the logic by which such judgements may be inferred. Natural deduction derivations are also known simply as "deductions". The frontier of a deduction is called its (global) **context**.

Essential to the meta-theory of natural deduction is the concept of **hypothetical judgement**, whereby certain inference rules involve **local assumptions**, which may be used only in the subderivation rooted at a particular premise of a rule instance without requiring justification. A local assumption is **discharged** by the rule instance that introduced it, and discharged assumptions do not enter the global context. Under the Curry-Howard correspondence, local assumptions correspond to bound variables, and the sites at which the former are discharged to those at which the latter are bound. Local assumptions are

conventionally indicated in rule figures using square brackets:

$$\dfrac{\mathsf{P}_1 \quad \cdots \quad \dfrac{\begin{array}{c}[\mathsf{A}]\\ \overline{\mathcal{D}_i}\end{array}}{\mathsf{P}_i} \quad \cdots \quad \mathsf{P}_n}{\mathsf{C}}\ r$$

where $\mathcal{D}_i$ is a meta-variable for the subderivation rooted at P_i. This notation is meant to indicate that A is an assumption local to $\mathcal{D}_i$ and discharged by the rule r. Conventionally, rule lines are not drawn between a derivation meta-variable and its assumptions or conclusion. Indeed, such a derivation could be an identity derivation and thus contain no primitive inferences at all. We choose to insert these (meta) lines, however, to help keep clear the tree structure of derivations that can otherwise become obscured in more complex cases.

Within an actual derivation we indicate the discharge of a local assumption by "closing it off" with a rule line above. This line should generally be labeled in some way to relate it to the inference responsible for the assumption, though the label is often omitted when the correlation is obvious. The difference between this and the bracket meta-notation is that a bracket indicates that a formula *may* occur among the (unspecified) assumptions, and if it does, then it may be discharged. Whereas, in an actual derivation we must state explicitly what the assumptions actually are and which among them we actually do discharge.

At any point in a derivation, the collection of locally available assumptions – those in the global context plus any local assumptions in scope – is called the **local context**, and typically referred to by some variant of "Γ". It is customary to abuse notation and write "Γ, A" for the context containing all elements of Γ as well as A. Rules that do not involve hypothetical judgement preserve the context; that is, the collection of assumptions available in each branch of a derivation subtree is the same as that at the root.

A set of natural deduction inference rules axiomatizing typed intuitionistic first-order logic is presented in figure 2.1. In the quantifier rules, capture-avoiding substitution performed at the meta-theoretic level is indicated by the notation "$\mathsf{A}[x \mapsto t]$", which expresses the result of safely substituting the expression t for free occurrences of the variable x within the expression A.[1] This may involve the renaming of bound variables, known as "α-conversion", which is considered to be a no-op. In other words, bound variable names are just a notational convenience and could be done away with entirely, for example, with de Bruijn indices.

The term "t", occurring in the rule $\forall -$ is called a **representative**, and in the rule $\exists +$, a **witness**, of its type. The term "e" occurring in the rules $\forall +$ and $\exists -$ is called an

[1] Note that the square brackets indicating substitution have nothing to do with those indicating local assumptions. It is just an unfortunate coincidence of two entrenched notations, which should anyway be unambiguous since a substitution may never be a local assumption, or vice-versa.

introduction rules	elimination rules
$\dfrac{}{\top}\ \top+$	no rule for $\top-$
no rule for $\bot+$	$\dfrac{\bot}{A}\ \bot-$
$\dfrac{A \quad B}{A \wedge B}\ \wedge+$	$\dfrac{A \wedge B}{A}\ \wedge-_1 \qquad \dfrac{A \wedge B}{B}\ \wedge-_2$
$\dfrac{A}{A \vee B}\ \vee+_1 \qquad \dfrac{B}{A \vee B}\ \vee+_2$	$\dfrac{A \vee B \quad \begin{matrix}[A]\\ \mathcal{D}_1\\ C\end{matrix} \quad \begin{matrix}[B]\\ \mathcal{D}_2\\ C\end{matrix}}{C}\ \vee-$
$\dfrac{\begin{matrix}[A]\\ \mathcal{D}\\ B\end{matrix}}{A \supset B}\ \supset+$	$\dfrac{A \supset B \quad A}{B}\ \supset-$
$\dfrac{\begin{matrix}[e : X]\\ \mathcal{D}\\ A[x \mapsto e]\end{matrix}}{\forall x : X \,.\, A}\ \forall+^{\dagger}$	$\dfrac{\forall x : X \,.\, A \quad t : X}{A[x \mapsto t]}\ \forall-$
$\dfrac{t : X \quad A[x \mapsto t]}{\exists x : X \,.\, A}\ \exists+$	$\dfrac{\exists x : X \,.\, A \quad \begin{matrix}[e : X]\,,\,[A[x \mapsto e]]\\ \mathcal{D}\\ B\end{matrix}}{B}\ \exists-^{\dagger}$

† e may not occur outside of $\mathcal{D}$ or in any open premise

Figure 2.1: Inference rules for intuitionistic first-order natural deduction

eigenvariable (also **generic variable** or **generic parameter**). The side-conditions of these rules ensure that it represents a generic term of its type, in the sense that nothing apart from its type can be inferred about it in the course of a derivation. We sometimes refer to the eigenvariable of the rule $\exists-$ as a **generic witness** and to that of $\forall+$ as a **generic representative**. This raises the question of what happens if the type of the eigenvariable is uninhabited. The answer is that local to the hypothetical subderivation $\mathcal{D}$, it is not. Within the hypothetical subderivation, by assumption, the type contains at least the eigenvariable; though this may not be true globally. It is precisely the local assumption that *some* term of the given type exists (and possibly satisfies some properties) that makes the subderivation hypothetical.

Since we equate propositions that differ only in their bound variables, we may choose to use the bound variable itself as the eigenvariable for the rules $\forall+$ and $\exists-$, after first safely renaming it away from any other variables in the local scope. We will see shortly the advantages of doing so, and will do this by default in subsequent chapters. To keep track of the scope of free term variables, we add a partition to our contexts, splitting each into a **propositional context** (or "logical context"), containing propositional assumptions, and a **typing context**, typically denoted "Φ", specifying the typed free variables in scope at any particular point in a derivation. We say that propositions and logical derivations are *dependent* on their typing contexts in the sense that they are only well-formed in typing contexts that include all of the free variables occurring within them. We will have much more to say about this dependence shortly. An **empty context** will either be written as "$\emptyset$", or suppressed entirely.

The derivability or **consequence relation** is indicated with the infix symbol "$\vdash$". So "$\Phi \mid \Gamma \vdash A$" expresses the judgement that there exists some derivation of conclusion A from logical assumptions Γ in typing context Φ. If there is more than one derivation system under consideration, a subscript is placed on the turnstile to disambiguate. We will refer to derivability in the system of natural deduction of figure 2.1 by "$\vdash_{\mathrm{I}}$", representing intuitionistic first-order derivability.

2.3 Harmony of the Connectives

The idea of connective harmony is presented in the philosophical investigations into logic by Michael Dummett [Dum91]. It is closely related to the notion of term equality in the type theory of Per Martin-Löf [Mar84]. The idea originates from an assertion by Gentzen [Gen35] that the meaning of a connective is characterized by its introduction rules,

> An introduction rule gives, so to say, a definition of the constant in question [...while...] an elimination rule is only a consequence of the corresponding introduction rule.

Prawitz [Pra65] interpreted this to mean that derivations ending in an introduction rule were canonical verifications of their conclusions. Dummett proposed to make this principle hereditary by requiring canonical verifications to end in a series of introductions. Furthermore, he recognized that this perspective has a dual: rather than connectives being characterized by their introduction rules, they could alternatively be characterized by their elimination rules, from which canonical consequences could be derived by a series of eliminations. Dummett called these two perspectives **verificationism** and **pragmatism**.

The verificationist perspective upholds the primacy of the introduction rules of natural deduction, and the pragmatist perspective, that of the elimination rules. However, the two perspectives can be reconciled since both insist that there be a **harmony** between the introduction and elimination rules of each connective. This harmony consists of two principles.

The principle of **local soundness** ensures that collectively, the elimination rules of a connective are not stronger than its introduction rules (slogan: "you may get out only what you have put in"). This means that in a derivation, if there is a proposition, which Prawitz called a "maximum formula", that is both the conclusion of an introduction inference and the major premise of an elimination inference (necessarily for the same connective) then there exists a derivation of the conclusion of the elimination inference directly from the collection of subderivations justifying the premises of the two inferences in question. Such a derivation will not contain these two inferences, nor the maximum formula. A transformation of derivations doing away with such a detour through a maximum formula is a **local reduction**. Local reduction can also be thought of as a **computation principle** for a connective because it transforms derivations, and by the *Curry-Howard correspondence* their realizer terms, into (in a sense to be made precise) simpler ones. This transformation is known as (generalized) **β-reduction**.[2]

The principle of **local completeness** ensures that collectively, the elimination rules of a connective are not weaker than its introduction rules (slogan: "you may get out all that you have put in"). Intuitively, this means that given any non-atomic proposition, we should be able to extract its consequences using its elimination rules, and use those consequences as evidence by which to reconstitute the proposition by reintroducing its principal connective. However, the tree-structure of natural deduction derivations makes the concrete expression of this intuition rather difficult to recognize in some cases. In chapter 5 we will see that there is a perfectly symmetrical categorical semantics lurking behind this asymmetrical tree syntax. A transformation of derivations of this sort is a **local expansion**. Local expansion can also be thought of as a **representation principle** for a connective because under

[2] We hedge with the word "generalized" here because in some contexts "β-reduction" is interpreted to refer narrowly to a reduction rule for function (arrow) type, or in logic, implication. But we use it more broadly to refer to the reduction that results from eliminating the detour just described.

Curry-Howard it guarantees that an arbitrary term of a given type has a canonical top-level representation. This transformation is known as (generalized) **η-expansion**.[3]

The local reductions and expansions witnessing the local soundness and completeness of the connectives of intuitionistic first-order logic, as presented in [Pfe09], are shown in figures 2.2 and 2.3, respectively. These are the typed versions of derivation conversions described by Prawitz in [Pra71]. Local soundness and completeness alone are not sufficient to ensure certain desirable global properties of natural deduction derivations. Chief among these is the existence of unique normal forms for derivations.

Given a rewriting relation on terms, $(- \rightsquigarrow -)$, a term is a **normal form** if it is not related to any terms:

$$t \text{ normal} \qquad \Leftrightarrow \qquad \forall u \,.\, \neg(t \rightsquigarrow u)$$

The uniqueness of normal forms, independent of their existence, is guaranteed by the property of **confluence**, which states that for $\rightsquigarrow^*$ the reflexive-transitive closure of $\rightsquigarrow$,

$$\forall t\,, u\,, v \,.\, (t \rightsquigarrow^* u \wedge t \rightsquigarrow^* v) \supset (\exists w \,.\, u \rightsquigarrow^* w \wedge v \rightsquigarrow^* w)$$

In particular, if u and v are both normal then they must be the same term. The **normalization** property states that every term rewrites to some normal form:

$$\forall t \,.\, \exists u \,.\, u \text{ normal} \wedge t \rightsquigarrow^* u$$

If every sequence of rewrites terminates, then the rewriting system is called **strongly normalizing**. In the presence of confluence, this implies that it doesn't matter how you do the rewriting, you will always reach the unique normal form.

Intuitively, we would like to take as the rewriting rules for natural deduction derivations the local reductions and local expansions. Unfortunately, such a naïve approach does not quite work. We consider first the rewriting system generated by the local reductions. In order for this system to have unique normalization we need to add the **permutation conversions** (also called "commuting conversions") for the connectives $\{\bot\,, \vee\,, \exists\}$ depicted in figure 2.4 as well. These are needed in order to transform derivations (and their proof terms) in such a way as to unite a "trivially separated" β-reducible pair of rules so that the β-reduction may be applied. For example, the disjunction permutation allows the conver-

[3] Again, we use the word "generalized" to indicate that we are not referring only to the expansion at arrow type.

no local reduction for $\top$

no local reduction for $\bot$

$$\dfrac{\dfrac{\dfrac{\mathcal{D}_1}{\mathrm{A}_1}\quad\dfrac{\mathcal{D}_2}{\mathrm{A}_2}}{\mathrm{A}_1 \wedge \mathrm{A}_2}\,\wedge{+}}{\mathrm{A}_i}\,\wedge{-}_i \quad \overset{\wedge>}{\longmapsto} \quad \dfrac{\mathcal{D}_i}{\mathrm{A}_i}$$

$$\dfrac{\dfrac{\dfrac{\mathcal{E}_i}{\mathrm{A}_i}}{\mathrm{A}_1 \vee \mathrm{A}_2}\,\vee{+}_i \qquad \dfrac{[\mathrm{A}_1]^u}{\dfrac{\mathcal{D}_1}{\mathrm{C}}} \qquad \dfrac{[\mathrm{A}_2]^v}{\dfrac{\mathcal{D}_2}{\mathrm{C}}}}{\mathrm{C}}\,\vee{-}^{u,v} \quad \overset{\vee>}{\longmapsto} \quad \dfrac{\dfrac{\mathcal{E}_i}{\mathrm{A}_i}}{\dfrac{\mathcal{D}_i}{\mathrm{C}}}$$

$$\dfrac{\dfrac{\dfrac{[\mathrm{A}]^u}{\dfrac{\mathcal{D}}{\mathrm{B}}}}{\mathrm{A} \supset \mathrm{B}}\,\supset{+}^u \qquad \dfrac{\mathcal{E}}{\mathrm{A}}}{\mathrm{B}}\,\supset{-} \quad \overset{\supset>}{\longmapsto} \quad \dfrac{\dfrac{\mathcal{E}}{\mathrm{A}}}{\dfrac{\mathcal{D}}{\mathrm{B}}}$$

$$\dfrac{\dfrac{\dfrac{[e : \mathrm{X}]}{\dfrac{\mathcal{D}}{\mathrm{A}[x \mapsto e]}}}{\forall x : \mathrm{X} \,.\, \mathrm{A}}\,\forall{+}^e \qquad \dfrac{\mathcal{T}}{t : \mathrm{X}}}{\mathrm{A}[x \mapsto t]}\,\forall{-} \quad \overset{\forall>}{\longmapsto} \quad \dfrac{\dfrac{\mathcal{T}}{t : \mathrm{X}}}{\dfrac{\mathcal{D}[e \mapsto t]}{\mathrm{A}[x \mapsto t]}}$$

$$\dfrac{\dfrac{\mathcal{T}}{t : \mathrm{X}} \qquad \dfrac{\mathcal{E}}{\mathrm{A}[x \mapsto t]}}{\exists x : \mathrm{X} \,.\, \mathrm{A}}\,\exists{+} \qquad \dfrac{[e : \mathrm{X}] \qquad \dfrac{[\mathrm{A}[x \mapsto e]]^u}{\dfrac{\mathcal{D}}{\mathrm{B}}}}{\mathrm{B}}\,\exists{-}^{e,u} \quad \overset{\exists>}{\longmapsto} \quad \dfrac{\dfrac{\mathcal{T}}{t : \mathrm{X}} \qquad \dfrac{\mathcal{E}}{\mathrm{A}[x \mapsto t]}}{\dfrac{\mathcal{D}[e \mapsto t]}{\mathrm{B}}}$$

Figure 2.2: Local reductions for intuitionistic first-order natural deduction ($* >$)

$$\frac{\mathcal{E}}{\top} \quad \overset{\top<}{\longmapsto} \quad \frac{}{\top}\,\top+$$

$$\frac{\mathcal{E}}{\bot} \quad \overset{\bot<}{\longmapsto} \quad \frac{\dfrac{\mathcal{E}}{\bot}}{\top}\,\bot-$$

$$\frac{\mathcal{E}}{\mathrm{A}\wedge\mathrm{B}} \quad \overset{\wedge<}{\longmapsto} \quad \frac{\dfrac{\dfrac{\mathcal{E}}{\mathrm{A}\wedge\mathrm{B}}\,\wedge-_1}{\mathrm{A}} \qquad \dfrac{\dfrac{\mathcal{E}}{\mathrm{A}\wedge\mathrm{B}}\,\wedge-_2}{\mathrm{B}}}{\mathrm{A}\wedge\mathrm{B}}\,\wedge+$$

$$\frac{\mathcal{E}}{\mathrm{A}\vee\mathrm{B}} \quad \overset{\vee<}{\longmapsto} \quad \frac{\dfrac{\mathcal{E}}{\mathrm{A}\vee\mathrm{B}} \qquad \dfrac{\overline{\mathrm{A}}\,u}{\mathrm{A}\vee\mathrm{B}}\,\vee+_1 \qquad \dfrac{\overline{\mathrm{B}}\,v}{\mathrm{A}\vee\mathrm{B}}\,\vee+_2}{\mathrm{A}\vee\mathrm{B}}\,\vee-^{u,v}$$

$$\frac{\mathcal{E}}{\mathrm{A}\supset\mathrm{B}} \quad \overset{\supset<}{\longmapsto} \quad \frac{\dfrac{\dfrac{\mathcal{E}}{\mathrm{A}\supset\mathrm{B}} \qquad \dfrac{}{\mathrm{A}}\,u}{\mathrm{B}}\,\supset-}{\mathrm{A}\supset\mathrm{B}}\,\supset+^{u}$$

$$\frac{\mathcal{E}}{\forall x : \mathrm{X}\,.\,\mathrm{A}} \quad \overset{\forall<}{\longmapsto} \quad \frac{\dfrac{\dfrac{\mathcal{E}}{\forall x : \mathrm{X}\,.\,\mathrm{A}} \qquad \dfrac{}{e : \mathrm{X}}}{\mathrm{A}[x\mapsto e]}\,\forall-}{\forall x : \mathrm{X}\,.\,\mathrm{A}}\,\forall+^{e}$$

$$\frac{\mathcal{E}}{\exists x : \mathrm{X}\,.\,\mathrm{A}} \quad \overset{\exists<}{\longmapsto} \quad \frac{\dfrac{\mathcal{E}}{\exists x : \mathrm{X}\,.\,\mathrm{A}} \qquad \dfrac{\dfrac{}{e : \mathrm{X}} \qquad \dfrac{}{\mathrm{A}[x\mapsto e]}\,u}{\exists x : \mathrm{X}\,.\,\mathrm{A}}\,\exists+}{\exists x : \mathrm{X}\,.\,\mathrm{A}}\,\exists-^{e,u}$$

Figure 2.3: Local expansions for intuitionistic first-order natural deduction ($* <$)

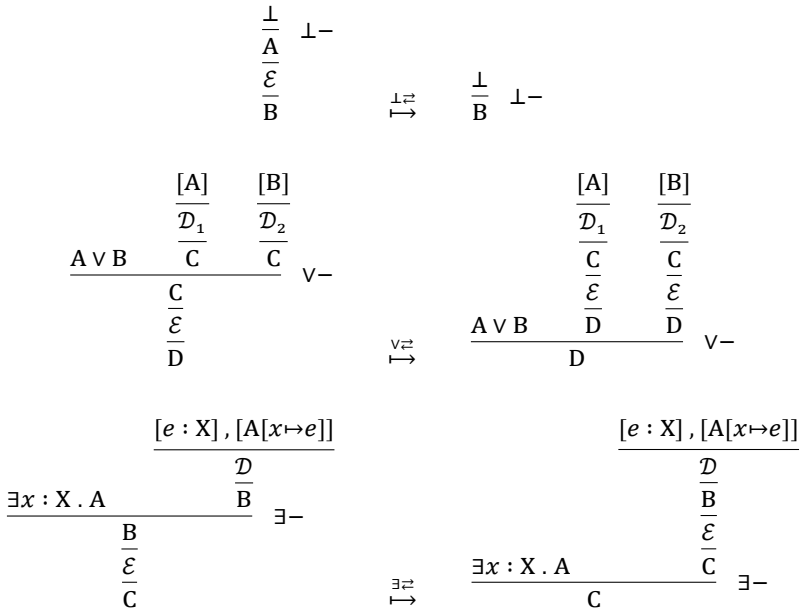
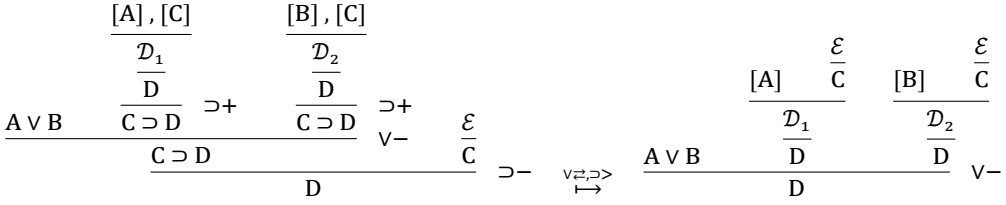


Figure 2.4: Permutation conversions for intuitionistic first-order natural deduction ($* \rightleftarrows$)

sion:



(2.1)

The proximate cause of the need for the permutation conversions is the nondeterministic decision of when to conclude pursuing a hypothetical subderivation and "return to the real world".[4] But as we will see in chapter 5, the deeper reason is *naturality*.

Prawitz proved the existence of unique normal forms for natural deduction derivations under the relation generated by the local reductions and permutation conversions in [Pra65]. We will call these the **β-normal forms**. The proof is constructive in that it allows one to actually compute β-normal forms. A simplified proof using the realizer terms under the Curry-Howard correspondence is presented by Girard in [GLT89]. Prawitz [Pra71] and Girard [Gir72] subsequently proved that the system is in fact strongly normalizing (indeed, Girard's result applies also to higher-order logics). This makes precise the sense in which

[4] Yes, this implies that there is a hypothetical subderivation lurking inside the ⊥− rule too, though it is, as yet, hard to see.

performing local reductions simplifies derivations: it brings them closer to their unique normal forms.

Allowing unrestricted local expansion as well leads to non-termination because the expansions can be applied repeatedly. Reorienting the local expansions as contractions is also problematic because it leads to a loss of confluence. However, we may safely compute the β-normal form of a derivation and then locally expand the assumptions. This results in the β-normal–η-long forms, henceforth **normal forms**, familiar from typed λ-calculus under the Curry-Howard correspondence. See [Dou93], [JG95] and [Gha95b] for details on the subtleties introduced by η-conversion.

Together, the local reductions, local expansions and permutation conversions generate an equivalence relation on the set of natural deduction derivations where each equivalence class contains a unique normal form. These equivalence classes, or equivalently, their normal representatives, will be the objects of study in our categorical semantics. As we will see in the sequel, each of these properties, local soundness, local completeness and the permutation conversions is a direct consequence of the definability of the connectives by adjoint functors. Indeed, the categorical perspective will show us that in fact all of the connectives can be seen to have permutation conversions, and that the connectives may act on derivations as well as on propositions.

Chapter 3

Categorical Constructions

In this chapter we summarize the main categorical constructions and properties that we will use in the sequel. All of the results presented here are "standard" in the mathematician's sense of being widely known and documented. Therefore, proofs are included only selectively in an attempt to help guide intuition or illustrate a particular style of reasoning. For a more thorough treatment of the topics described here, as well as the basic categorical concepts on which they rely, we recommend [Awo10], [BW98] or [Mac98]. We should mention that our categorical notation may differ from that to which the reader is accustomed. If this should be the case, please consult the brief appendix A explaining our notation.

3.1 Adjunctions

Adjunction is a surprisingly useful gadget from the field of category theory. It is a relationship that may hold between a pair of anti-parallel functors that generalizes the concept of Galois correspondence. There are several useful equivalent characterizations of the concept of adjunction, we begin with the following:

Definition 3.1.1 (adjunction) Let $\mathtt{F} : \mathbb{A} \longrightarrow \mathbb{B}$ and $\mathtt{G} : \mathbb{B} \longrightarrow \mathbb{A}$ be a pair of anti-parallel functors. $\mathtt{F}$ and $\mathtt{G}$ form an **adjunction**, written "$\mathtt{F} \dashv \mathtt{G}$", if any of the following circumstances, which are equivalent, obtain.

natural isomorphism of hom bifunctors

There exists a natural isomorphism of hom bifunctors,

$$\theta : ((\mathbb{A}^{\circ} \times \mathbb{B}) \supset \textsc{Set})\,(\mathbb{A}\,(\tfrac{1}{-} \to \mathtt{G}(\tfrac{2}{-})) \to \mathbb{B}\,(\mathtt{F}(\tfrac{1}{-}) \to \tfrac{2}{-}))$$

that is, for any objects $\mathtt{A} : \mathbb{A}$ and $\mathtt{B} : \mathbb{B}$, there is a bijection of hom sets:

$$\frac{\mathbb{A}\,(\mathtt{A} \to \mathtt{G}(\mathtt{B}))}{\mathbb{B}\,(\mathtt{F}(\mathtt{A}) \to \mathtt{B})}\ \theta(\mathtt{A}\,,\mathtt{B})$$

and for any arrows $a : \mathbb{A}\,(A' \to A)$ and $b : \mathbb{B}\,(B \to B')$, the following diagram commutes:

$$\begin{array}{ccc} \mathbb{A}\,(A \to G(B)) & \xrightarrow[\cong]{\theta(A\,,\,B)} & \mathbb{B}\,(F(A) \to B) \\ \downarrow{\scriptstyle \mathbb{A}\,(a \to G(b))} & & \downarrow{\scriptstyle \mathbb{B}\,(F(a) \to b)} \\ \mathbb{A}\,(A' \to G(B')) & \xrightarrow[\theta(A'\,,\,B')]{\cong} & \mathbb{B}\,(F(A') \to B') \end{array} \tag{3.1}$$

universal property of the counit

There exists a natural transformation $\varepsilon : G \cdot F \longrightarrow \mathrm{id}_{\mathbb{B}}$ such that for any arrow $g : \mathbb{B}\,(F(A) \to B)$, there is a unique arrow $g^\flat : \mathbb{A}\,(A \to G(B))$ such that $F(g^\flat) \cdot \varepsilon(B) = g$:

$$\begin{array}{ll} \mathbb{A}: & A \overset{g^\flat}{\dashrightarrow} G(B) \\ & \Vert \\ \mathbb{B}: & F(A) \xrightarrow{\;g\;} B \\ & \quad{\scriptstyle F(g^\flat)}\searrow \quad \uparrow{\scriptstyle \varepsilon(B)} \\ & \qquad (F \circ G)(B) \end{array} \tag{3.2}$$

universal property of the unit

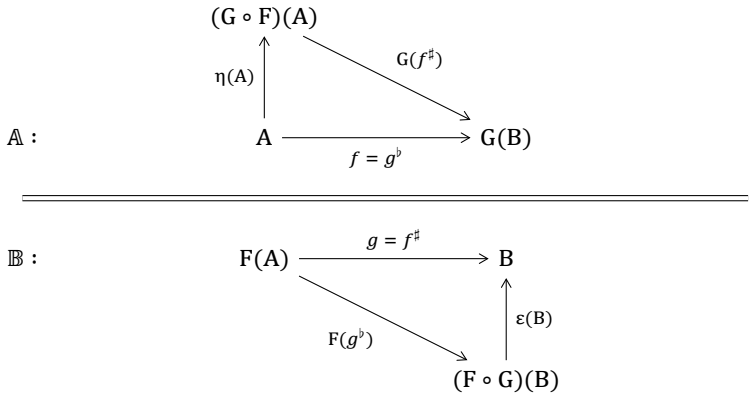
There exists a natural transformation $\eta : \mathrm{id}_{\mathbb{A}} \longrightarrow F \cdot G$ such that for any arrow $f : \mathbb{A}\,(A \to G(B))$, there is a unique arrow $f^\sharp : \mathbb{B}\,(F(A) \to B)$ such that $\eta(A) \cdot G(f^\sharp) = f$:

$$\begin{array}{ll} & (G \circ F)(A) \\ & \quad \uparrow{\scriptstyle \eta(A)} \quad \searrow{\scriptstyle G(f^\sharp)} \\ \mathbb{A}: & A \xrightarrow[f]{} G(B) \\ & \Vert \\ \mathbb{B}: & F(A) \overset{f^\sharp}{\dashrightarrow} B \end{array} \tag{3.3}$$

The bijection θ of the natural isomorphism of hom bifunctors characterization is the bijection $-^\sharp$ of the universal property of the unit characterization, and its inverse, θ^{-1}, is the bijection $-^\flat$ of the universal property of the counit characterization. In an adjunction $F \dashv G$, F is said to be **left adjoint** to G and G **right adjoint** to F. The natural transformations η and ε are called the **unit** and **counit** of the adjunction, respectively. When the intended adjunction is clear from context or irrelevant, we refer to the arrows of a pair $(f\,,\,f^\sharp)$ or $(g\,,\,g^\flat)$ as **adjoint complements** of one another. It is sometimes convenient to summarize an adjunction with the tuple, $(\mathbb{A}\,,\,\mathbb{B}\,,\,F\,,\,G\,,\,\eta\,,\,\varepsilon)$.

The reader may have noticed the *duality* between the two universal property characterizations of an adjunction. We will return to this point shortly. We now briefly mention the intuition behind the musical notation. Many familiar instances of the concept of adjunction arise in the form of algebraic structures, where the category $\mathbb{A}$ is Set and $\mathbb{B}$ is, to take one example, Grp, the category of groups and their homomorphisms. In this case, the free group functor is left adjoint to the underlying set functor and the natural bijection θ associates a "plain arrow" in the form of an ordinary function to a "fancy arrow" in the form of a group homomorphism. Thus mnemonically, $-^\sharp$ is "fancification" while $-^\flat$ is "plainification". Of course not all adjunctions fit this pattern, but we have found the mnemonic to be helpful.

Because the characterizations of an adjunction given above are equivalent, it will be convenient to summarize an adjunction diagrammatically as follows:



(3.4)

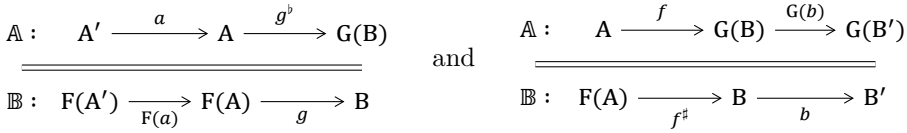
By setting g to $\varepsilon(\mathrm{B})$ or f to $\eta(\mathrm{A})$, it is easy to work out their adjoint complements.

Lemma 3.1.2 (adjoint complements of (co)unit components) The adjoint complements of the counit and unit of an adjunction are identity arrows on adjoint functor images; that is,

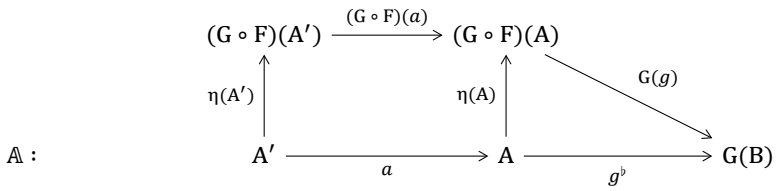
$$(\varepsilon(-))^\flat = \mathrm{id}_{\mathrm{G}(-)} \qquad\qquad (\eta(-))^\sharp = \mathrm{id}_{\mathrm{F}(-)}$$

In the sequel, it will be important for us to understand how arrows with adjoint complements behave under composition.

Lemma 3.1.3 (composition with adjoint complements) For adjunction $\mathrm{F} \dashv \mathrm{G}$, if arrow $g :: \mathbb{B}$ has adjoint complement $g^\flat :: \mathbb{A}$ then $a \cdot g^\flat = (\mathrm{F}(a) \cdot g)^\flat$; likewise, if arrow $f :: \mathbb{A}$ has adjoint complement $f^\sharp :: \mathbb{B}$ then $f^\sharp \cdot b = (f \cdot \mathrm{G}(b))^\sharp$, whenever the compositions are defined. That is,



Proof. Actually, these are exactly the conditions necessary for naturality in the "natural isomorphism of hom bifunctors" characterization of adjunction in definition 3.1.1. But we give here a proof using the universal properties characterizations to illustrate the connection. In the first case, pasting the diagrams for the naturality of η at a and the adjoint factorization of $g^\flat$ along $\eta(\mathrm{A})$ yields:



By the functoriality of G,

$$(\mathrm{G} \circ \mathrm{F})(a) \cdot \mathrm{G}(g) = \mathrm{G}(\mathrm{F}(a) \cdot g)$$

but by the uniqueness of adjoint complements,

$$\exists!(a \cdot g^\flat)^\sharp : \mathbb{B}\,(\mathrm{F}(\mathrm{A}') \to \mathrm{B})\,.\,\eta(\mathrm{A}') \cdot \mathrm{G}((a \cdot g^\flat)^\sharp) = a \cdot g^\flat$$

so it must be that

$$(a \cdot g^\flat)^\sharp = \mathrm{F}(a) \cdot g$$

The other case is dual to this one. □

By setting the arrows g and f in the preceding lemma to the respective identities, we obtain the following useful characterization of the of the action of adjoint functors on arrows.

Corollary 3.1.4 (adjoint functor image of arrows) For adjunction $\mathrm{F} \dashv \mathrm{G}$ with unit η and counit ε,

$$\mathrm{F}(-) = (- \cdot \eta(\mathrm{cod}(-)))^\sharp \qquad \text{and} \qquad \mathrm{G}(-) = (\varepsilon(\mathrm{dom}(-)) \cdot -)^\flat$$

A fundamental property of adjoint functors is their monogamy.

Lemma 3.1.5 (uniqueness of adjoints) Adjoint functors determine each other uniquely, up to natural isomorphism.

Proof. Given the functors $\mathrm{F}, \mathrm{F}' : \mathbb{A} \longrightarrow \mathbb{B}$ and $\mathrm{G}, \mathrm{G}' : \mathbb{B} \longrightarrow \mathbb{A}$, and the adjunctions $\mathrm{F} \dashv \mathrm{G}$ and $\mathrm{F}' \dashv \mathrm{G}'$, if $\mathrm{G} \cong \mathrm{G}'$ then for any $\mathrm{A} : \mathbb{A}$ and $\mathrm{B} : \mathbb{B}$,

$$\mathbb{B}\,(\mathrm{F}(\mathrm{A}) \to \mathrm{B}) \cong \mathbb{A}\,(\mathrm{A} \to \mathrm{G}(\mathrm{B})) \cong \mathbb{A}\,(\mathrm{A} \to \mathrm{G}'(\mathrm{B})) \cong \mathbb{B}\,(\mathrm{F}'(\mathrm{A}) \to \mathrm{B})$$

The isomorphisms are natural in B, so extend to the hom functors:

$$\mathbb{B}\,(\mathrm{F}(\mathrm{A}) \to -) \cong \mathbb{A}\,(\mathrm{A} \to \mathrm{G}(-)) \cong \mathbb{A}\,(\mathrm{A} \to \mathrm{G}'(-)) \cong \mathbb{B}\,(\mathrm{F}'(\mathrm{A}) \to -)$$

The transitive isomorphism, together with the **Yoneda principle**, which states that being full and faithful, the Yoneda embeddings reflect isomorphisms, implies that

$$\forall A : \mathbb{A} \,.\, F(A) \cong F'(A)$$

still natural in A, giving us the natural isomorphism $F \cong F'$. The case for right adjoints is dual. □

Another fundamental property of adjoint functors is their interaction with (co)limits.

Lemma 3.1.6 ((co)continuity of adjoint functors) Right adjoint functors preserve limits, dually, left adjoint functors preserve colimits.

Indeed, this is the idea behind yet another equivalent formulation of the concept of adjunction known as "Freyd's adjoint functor theorem", which is a partial converse to this lemma.

In some ways we might prefer a structure somewhat less strict than that of an adjunction. In particular, we might prefer that triangles (3.2) and (3.3) commute only up to a canonical 2-morphism. We will see that this would allow us to distinguish between $\beta\eta$-convertible derivations, but would also require a lax 2-categorical version of adjunction and add significant complexity (see for example, [See79]). Therefore we proceed with ordinary 1-dimensional categories and adjunctions, but will take the liberty of referring to the equalities short-cutting the triangles of (3.2) and (3.3) as "$\beta(g)$" and "$\beta'(f)$", respectively, for reasons that will soon become clear.

There is one good reason to prefer the 1-dimensional approach we are taking, at least when it comes to proof search. Since intuitionistic first-order natural deduction derivations have unique normal forms (i.e. the β-normal–η-long forms[1]), by identifying equivalence classes of derivations with their normal forms we may prune the search space of potential derivations without sacrificing completeness. To take a trivial example, there are many natural deduction proofs of $\top$, indeed, any valid proof whatsoever composed with the $\top+$ rule is one such. But by considering them all equivalent to the normal one, comprising just a single instance of the $\top+$ rule, we save ourselves from having to considering the infinity of others.

Despite the decision not to venture into the world of 2-categorical rewriting theory (cf. [Gha95a]), it will still prove useful in the sequel to understand adjunctions from a 2-categorical perspective. Fortunately, the 2-category in this case is Cat, so nothing too exotic

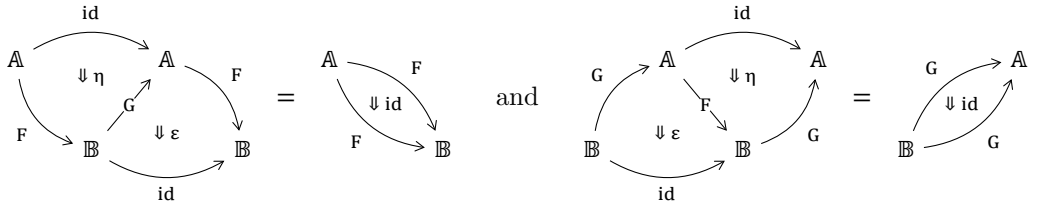
[1] We warn the unsuspecting reader not to confuse the "η" naming a conversion rule of the λ-calculus with the "η" naming the unit of an adjunction. Both uses are too entrenched for us to hope to change them, but anyway context should make abundantly clear which is intended.

will be involved. We begin by giving yet another equivalent characterization of adjunction, this one "external" or "global", describing *behavior*, in contrast to the "internal" or "local" characterizations of definition 3.1.1, that describe *structure*.

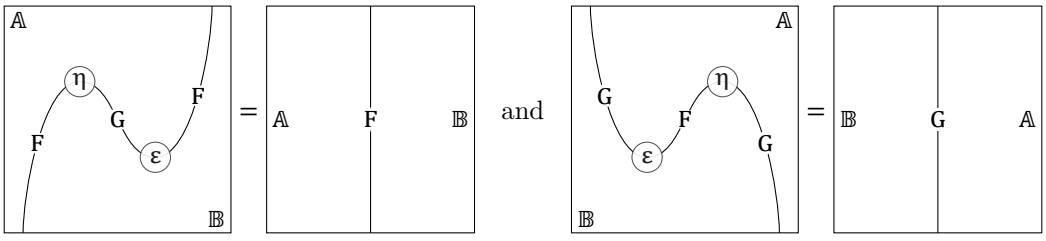
Proposition 3.1.7 (external characterization of adjunction) Given a pair of anti-parallel functors, $F : \mathbb{A} \longrightarrow \mathbb{B}$ and $G : \mathbb{B} \longrightarrow \mathbb{A}$, and pair of natural transformations $\eta : \mathrm{id}_{\mathbb{A}} \longrightarrow F \cdot G$ and $\varepsilon : G \cdot F \longrightarrow \mathrm{id}_{\mathbb{B}}$, there is an adjunction $F \dashv G$ with unit η and counit ε just in case the following two **adjunction laws** are satisfied:

$$(\eta \cdot \cdot F) \cdot (F \cdot \cdot \varepsilon) = \mathrm{id}_F \qquad\qquad (G \cdot \cdot \eta) \cdot (\varepsilon \cdot \cdot G) = \mathrm{id}_G \tag{3.5}$$

These equations surely seem rather opaque upon a first encounter. Fortunately, they have intuitive graphical representations. We can draw η and ε each as a triangular 2-cell. Any two 2-cells may be pasted together along a common boundary to form a composite 2-cell so long as their orientations are compatible; that is, so long as the common edge is part of the 1-dimensional codomain of the first 2-cell and part of the 1-dimensional domain of the second 2-cell. As *pasting diagrams* the adjunction laws become:



whence they get the name **triangle equations**. Another way to understand the adjunction laws is by using the dual graphs of pasting diagrams, known as *string diagrams*. In the graphical language of string diagrams it is customary to not annotate the 1- and 2-cells with their orientations, but rather to assign them prevailing orientations throughout a diagram. We will adopt the convention of reading 1-cells left-to-right and 2-cells top-to-bottom. It is also customary to suppress the drawing of identity cells. As string diagrams, the adjunction laws become:

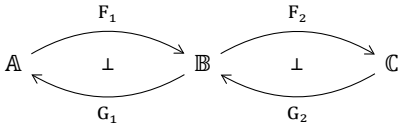


whence they get the alternative name **zig-zag equations** or "yankings", since one can very intuitively imagine grabbing the two ends of a zig-zag and yanking it straight.

One advantage of both of these graphical formulations is that they permit us to interpret the adjunction laws as saying essentially that the unit and counit of an adjunction are

mutually inverse. Another is to shed light on the deep symmetry inherent in the concept of adjunction. For example, by the zig-zag equations we can see that if η and ε are the unit and counit of an adjunction $F \dashv G$, then reversing their orientations to η° and ε° by reflecting the string diagrams vertically yields another adjunction with the roles of the natural transformations, as well as those of the functors in the adjunction, swapped. Since reversing a natural transformation reverses the orientation of its components, it is not hard to see that this dual adjunction must be $G^\circ \dashv F^\circ$ on the opposites of the original categories. Reversing the orientations of 2-cells corresponds to the categorical *duality* called "co-". So we see that a co-adjunction is again an adjunction. This explains the duality between the universal properties of the unit and counit: the counit of an adjunction is also the unit of the corresponding co-adjunction, and vice-versa.

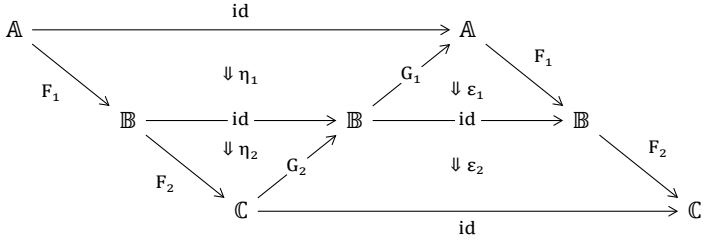
Lemma 3.1.8 (composition of adjunctions) Given a pair of "adjacent" adjunctions $F_1 \dashv G_1$ and $F_2 \dashv G_2$ as shown:



there is a "composite" adjunction $F_1 \cdot F_2 \dashv G_2 \cdot G_1$

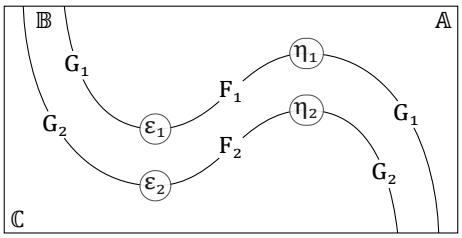
Proof. If the units and counits of the adjacent adjunctions are η_1, ε_1 and η_2, ε_2, respectively, then the the composite adjunction has unit $\eta_1 \cdot (F_1 \cdot \cdot \eta_2 \cdot \cdot G_1)$ and counit $(G_2 \cdot \cdot \varepsilon_1 \cdot \cdot F_2) \cdot \varepsilon_2$.

One way to see that the adjunction laws are then satisfied is by using a pasting diagram,



where each horizontal strip is equal to the identity natural transformation by the triangle equation for the respective adjunction, and so the whole diagram is the identity natural transformation on $F_1 \cdot F_2$.

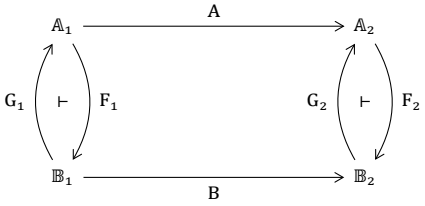
Another way is to use a string diagram,



and to yank both zig-zags simultaneously, leaving the identity natural transformation on $\mathsf{G}_2 \cdot \mathsf{G}_1$. □

In fact, by arbitrarily (but consistently) assigning to adjunctions the orientation of either their left or right adjoint functor, we can define a **category of adjunctions**, where an identity adjunction has identity functors as both its left and right adjoint.

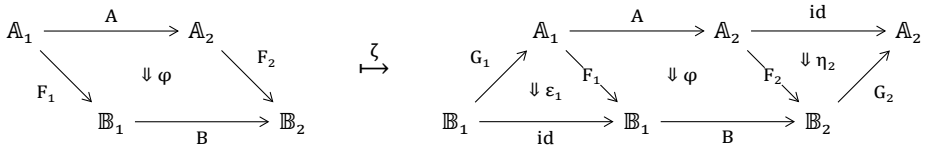
Lemma 3.1.9 Given adjunctions $\mathsf{F}_1 \dashv \mathsf{G}_1$ and $\mathsf{F}_2 \dashv \mathsf{G}_2$ and functors A and B as shown (without assuming that anything commutes):



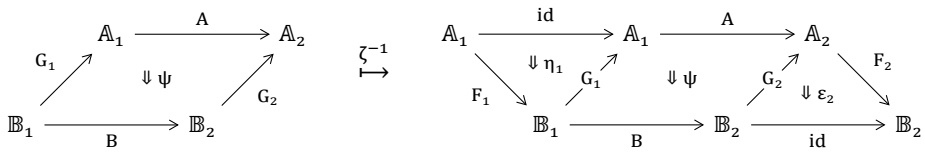
there is a bijection between sets of natural transformations:

$$\frac{\mathbb{A}_1 \supset \mathbb{B}_2\,(\mathsf{A} \cdot \mathsf{F}_2 \to \mathsf{F}_1 \cdot \mathsf{B})}{\mathbb{B}_1 \supset \mathbb{A}_2\,(\mathsf{G}_1 \cdot \mathsf{A} \to \mathsf{B} \cdot \mathsf{G}_2)}\ \zeta$$

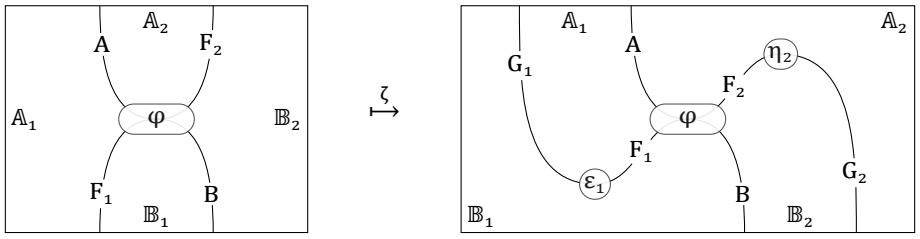
Proof. Such a bijection is given by pasting, equivalently concatenating, with the respective unit and counit of the two adjunctions:



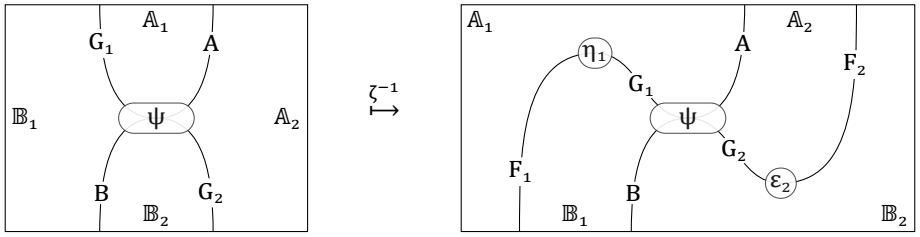
and



or equivalently,



and



The triangle equations, respectively, zig-zag equations, ensure that going back and forth in either order amounts to the identity. □

Definition 3.1.10 (mate) Natural transformations related by the bijection ζ are each called the other's **mate**. In case we need to be more specific, we will call φ the **left mate** and ψ the **right mate**, after the adjoint functors involved.

Definition 3.1.11 (Beck-Chevalley condition) The **Beck-Chevalley condition** is the requirement that the mate of a natural isomorphism is itself a natural isomorphism.

This condition could be separated into two distinct requirements, one for each direction across the bijection, but we will want both, so for us the condition states that a natural transformation with a (left or right) mate is an isomorphism just in case its mate is too.

3.2 Comonads

Definition 3.2.1 (comonad) Given a category $\mathbb{B}$, a **comonad** on $\mathbb{B}$ is a tuple $(\mathbb{B}, S, \varepsilon, \delta)$ where,

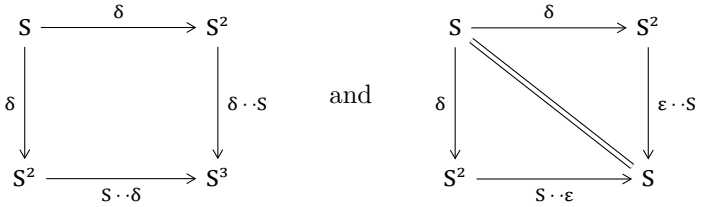
$$S : \mathbb{B} \longrightarrow \mathbb{B} \quad , \quad \varepsilon : S \longrightarrow \mathrm{id}_{\mathbb{B}} \quad , \quad \delta : S \longrightarrow S^2$$

such that the following equations are satisfied:

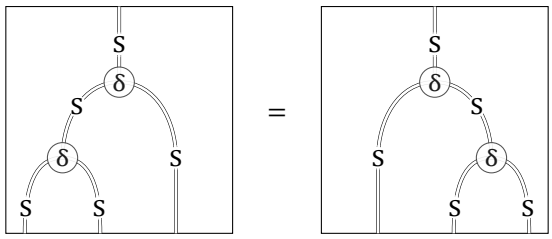
associative law: $\delta \cdot (\delta \cdot \cdot S) = \delta \cdot (S \cdot \cdot \delta)$

counit laws: $\delta \cdot (\varepsilon \cdot \cdot S) = \mathrm{id}_S = \delta \cdot (S \cdot \cdot \varepsilon)$

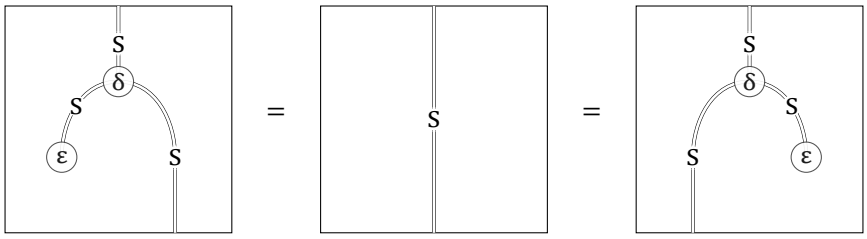
The natural transformations ε and δ called the **counit** and **comultiplication** of the comonad, respectively. When there is no danger of confusion, it is customary to refer to comonads by their endofunctors, as in "the comonad S". We can represent these equations as commuting diagrams in the monoidal category $\mathbb{B} \supset \mathbb{B}$ as:



or perhaps more helpfully, as equations in string diagrams as,



and

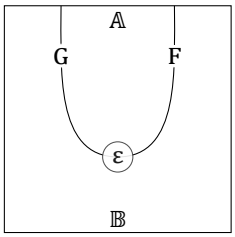


Comonads are closely related to adjunctions, as are their probably more well-known duals, **monads**, which have lately become a common abstraction in the field of functional programming. Readers already comfortable with monads will find the diagrams in this section familiar when turned upside-down (or rather, reflected vertically), as the duality "co-" corresponds to reversing the orientation of 2-cells. We begin by noting that to every adjunction there is associated a canonical comonad in the following sense:

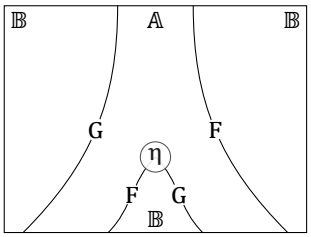
Lemma 3.2.2 (comonads from adjunctions) If $\mathcal{A} = (\mathbb{A}, \mathbb{B}, \mathsf{F}, \mathsf{G}, \eta, \varepsilon)$ is an adjunction, then $\mathit{com}(\mathcal{A}) \coloneqq (\mathbb{B}, \mathsf{G} \cdot \mathsf{F}, \varepsilon, \mathsf{G} \cdot \cdot \eta \cdot \cdot \mathsf{F})$ is a comonad.

Proof. Here is a picture-proof in string diagrams. Using the notation from the definition of comonad, the lemma asserts that the associative law and counit laws hold when S is $\mathsf{G} \cdot \mathsf{F}$ and δ is $\mathsf{G} \cdot \cdot \eta \cdot \cdot \mathsf{F}$. So if we were to "zoom in" on the counit and comultiplication, we would

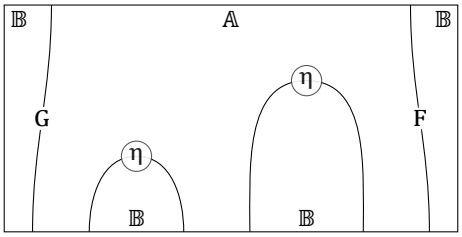
see, respectively,



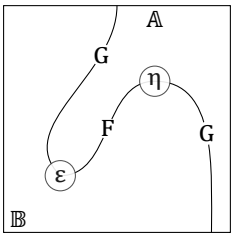
and



This is why we have suggestively drawn the functor S in the string diagrams for comonads as a ribbon: we imagine that there is a little category 𝔸 curled up inside. If we were to "fatten up" the inside of the ribbon in the left string diagram for associativity, it would then look like this:



The associative law holds because, being natural transformations and thus satisfying the interchange law, the two instances of η are free to slide up or down past one another within the G – F tube. The counit laws hold because any S-branch ending in ε looks like this (or its mirror-image with the 1-cells swapped):



This is a zig-zag and so may be yanked straight. □

The converse situation is more complex. Beginning from a comonad S, an adjunction $\mathcal{A}$ such that $com(\mathcal{A}) = \mathsf{S}$ is called an **adjoint resolution** (or just "resolution") of S. In general, there are many different ways to resolve a comonad into an adjunction, and in fact the collection of adjoint resolutions itself forms a category. Such a category of adjoint resolutions always has both initial and terminal objects, which form extremal canonical resolutions of the comonad. A terminal resolution, known as the **Eilenberg–Moore resolution**, involves the category of comonadic coalgebras of the functor S. Although a fascinating topic in its own right, we will not need to make use of it in the sequel, so we refer the interested reader to [Mac98] for further details. Embedded within the category of comonadic coalgebras as a

full subcategory lies the category of co-free comonadic coalgebras, which is equivalent to a category, called the Kleisli category, given by a simple syntactic construction. This category provides an initial resolution of the comonad, called the Kleisli resolution, which we now describe.

Lemma 3.2.3 (Kleisli category) Given a comonad $(\mathbb{B}, \mathsf{S}, \varepsilon, \delta)$ the following data define a category, called the **Kleisli category** of S and written "$\mathbb{B}_\mathsf{S}$".

objects: those of $\mathbb{B}$,

$$\mathsf{A} : \mathbb{B}_\mathsf{S} \quad \coloneqq \quad \mathsf{A} : \mathbb{B}$$

arrows: $\mathbb{B}$-arrows from S-images,

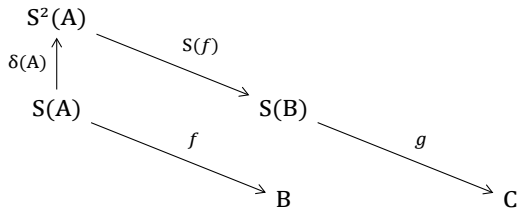
$$\mathbb{B}_\mathsf{S}(\mathsf{A} \to \mathsf{B}) \quad \coloneqq \quad \mathbb{B}(\mathsf{S}(\mathsf{A}) \to \mathsf{B})$$

composition: for arrows $f : \mathbb{B}_\mathsf{S}(\mathsf{A} \to \mathsf{B})$ and $g : \mathbb{B}_\mathsf{S}(\mathsf{B} \to \mathsf{C})$,

$$(f \cdot g) : \mathbb{B}_\mathsf{S}(\mathsf{A} \to \mathsf{C}) \quad \coloneqq \quad (\delta(\mathsf{A}) \cdot \mathsf{S}(f) \cdot g) : \mathbb{B}(\mathsf{S}(\mathsf{A}) \to \mathsf{C})$$

as shown:

$\mathbb{B}$:



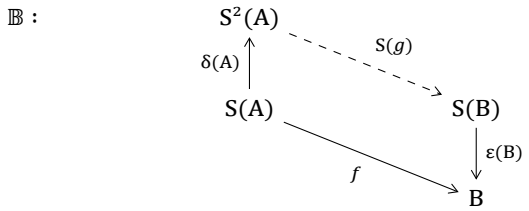
identities: counit components,

$$\mathsf{id}_\mathsf{B} : \mathbb{B}_\mathsf{S}(\mathsf{B} \to \mathsf{B}) \quad \coloneqq \quad \varepsilon(\mathsf{B}) : \mathbb{B}(\mathsf{S}(\mathsf{B}) \to \mathsf{B})$$

Lemma 3.2.4 (Kleisli resolution) Given a comonad $(\mathbb{B},\mathsf{S},\varepsilon,\delta)$, the tuple $(\mathbb{B}_\mathsf{S},\mathbb{B},\mathsf{F}_\mathsf{S},\mathsf{G}_\mathsf{S},\eta_\mathsf{S},\varepsilon)$, where $\mathbb{B}_\mathsf{S}$ is the Kleisli category of S and F_S, G_S, and η_S are as defined below, is an adjoint resolution of S, known as the **Kleisli resolution**.

$$\begin{array}{rcl} & \underline{\mathsf{F}_\mathsf{S}} & \\ \mathbb{B}_\mathsf{S} & \longrightarrow & \mathbb{B} \\ \mathsf{B} & \longmapsto & \mathsf{S}(\mathsf{B}) \\ f : \mathsf{A} \longrightarrow \mathsf{B} & \longmapsto & \delta(\mathsf{A}) \cdot \mathsf{S}(f) \end{array} \qquad \begin{array}{rcl} & \underline{\mathsf{G}_\mathsf{S}} & \\ \mathbb{B} & \longrightarrow & \mathbb{B}_\mathsf{S} \\ \mathsf{B} & \longmapsto & \mathsf{B} \\ f : \mathsf{A} \longrightarrow \mathsf{B} & \longmapsto & \varepsilon(\mathsf{A}) \cdot f :: \mathbb{B} \end{array} \qquad \begin{array}{rcl} & \underline{\eta_\mathsf{S}} & \\ \mathsf{id}_{\mathbb{B}_\mathsf{S}} & \longrightarrow & \mathsf{F}_\mathsf{S} \cdot \mathsf{G}_\mathsf{S} \\ \mathsf{B} & \longmapsto & \mathsf{id}_{\mathsf{S}(\mathsf{B})} :: \mathbb{B} \end{array}$$

Proof. We demonstrate the universal property of the counit. After unpacking the definitions involved, it must be shown that for any $f : \mathbb{B}\,(S(A) \to B)$ there is a unique $g : \mathbb{B}\,(S(A) \to B)$ making the following diagram commute:



But then we would have:

$$\begin{aligned} & \delta(A) \cdot S(g) \cdot \varepsilon(B) \\ = \quad & \text{[naturality of } \varepsilon\text{]} \\ & \delta(A) \cdot \varepsilon(S(A)) \cdot g \\ = \quad & \text{[counit law]} \\ & id_A \cdot g \\ = \quad & g \end{aligned}$$

So g must be just f itself. □

The arrow $F_S(f)$ is also known as the **Kleisli extension** of f and written "$\overline{f}$". The idea is that when composing a sequence of arrows in $\mathbb{B}_S$, each save the last has its F_S-image composed in $\mathbb{B}$. Thus, an equivalent way to think about composing a sequence of arrows in $\mathbb{B}_S$ is to compose all of their F_S-images in $\mathbb{B}$ and then postcompose the component of ε in $\mathbb{B}$, which is just postcomposing an identity arrow in $\mathbb{B}_S$. Haskell programmers may recognize this as the dual of a monad construction called "bind" and written as "$(>>= f)$".

3.3 Bicartesian Closed Categories

Definition 3.3.1 (bicartesian closed category) A **bicartesian closed category** is one having the following universal constructions:

- binary cartesian product $(- \times -)$
- terminal object (1)
- binary coproduct $(- + -)$
- initial object (0)
- exponential $(- \supset -)$

Together, a binary cartesian product and terminal object generate **finite products**, and dually, a binary coproduct and initial object generate **finite coproducts**. A category with just finite products and exponentials is a **cartesian closed category**, while one with only finite products is a **cartesian category**.

The functor determining each of these universal constructions has either a left or right adjoint, as follows. The binary cartesian product and coproduct bifunctors are right and left adjoints, respectively, to a **diagonal functor**, Δ, that simply splits its input into two identical copies: $+ \dashv \Delta \dashv \times$.

$$\begin{array}{ccc} & \underline{+} & \\ \mathbb{C} \times \mathbb{C} & \longrightarrow & \mathbb{C} \\ (A\,,B) & \longmapsto & A + B \\ (f\,,g) & \longmapsto & f + g \end{array} \qquad \begin{array}{ccc} & \underline{\Delta} & \\ \mathbb{C} & \longrightarrow & \mathbb{C} \times \mathbb{C} \\ A & \longmapsto & (A\,,A) \\ f & \longmapsto & (f\,,f) \end{array} \qquad \begin{array}{ccc} & \underline{\times} & \\ \mathbb{C} \times \mathbb{C} & \longrightarrow & \mathbb{C} \\ (A\,,B) & \longmapsto & A \times B \\ (f\,,g) & \longmapsto & f \times g \end{array}$$

Similarly, the constant functors that pick out terminal and initial objects of a category are right and left adjoints, respectively, to the necessarily unique functor, !, to a **terminal category**: $0 \dashv ! \dashv 1$.

$$\begin{array}{ccc} & \underline{0} & \\ \mathbb{1} & \longrightarrow & \mathbb{C} \\ * & \longmapsto & 0 \\ \mathrm{id}_* & \longmapsto & \mathrm{id}_0 \end{array} \qquad \begin{array}{ccc} & \underline{!} & \\ \mathbb{C} & \longrightarrow & \mathbb{1} \\ A & \longmapsto & * \\ f & \longmapsto & \mathrm{id}_* \end{array} \qquad \begin{array}{ccc} & \underline{1} & \\ \mathbb{1} & \longrightarrow & \mathbb{C} \\ * & \longmapsto & 1 \\ \mathrm{id}_* & \longmapsto & \mathrm{id}_1 \end{array}$$

Finally, the universal property of exponentials is determined by the well-known **curry adjunction**: for each $B : \mathbb{C}$ there is an adjunction, $- \times B \dashv B \supset -$.

$$\begin{array}{ccc} & \underline{- \times B} & \\ \mathbb{C} & \longrightarrow & \mathbb{C} \\ A & \longmapsto & A \times B \\ f & \longmapsto & f \times \mathrm{id}_B \end{array} \qquad \begin{array}{ccc} & \underline{B \supset -} & \\ \mathbb{C} & \longrightarrow & \mathbb{C} \\ C & \longmapsto & B \supset C \\ g & \longmapsto & \mathrm{id}_B \supset g \end{array}$$

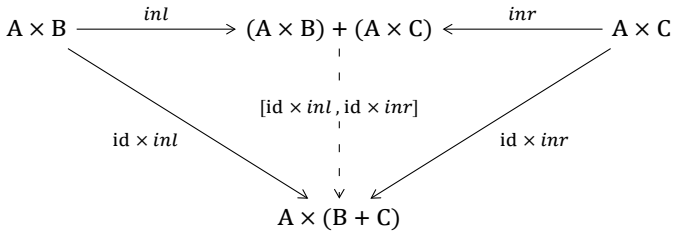
We will have occasion to examine each of these adjunctions in detail in the sequel.

One fact about bicartesian closed categories that we will make use of is that they necessarily satisfy a distributive property of products over coproducts, commonly known as the distributive law:

Lemma 3.3.2 (distributive law) In any bicartesian closed category $\mathbb{C}$,

$$A \times (B + C) \cong (A \times B) + (A \times C)$$

Proof. Note that in any category with products and coproducts there is a canonical arrow in the reverse direction:



so the content of the lemma is that the presence of exponentials makes this arrow invertible. For arbitrary X, we have:

$$
\begin{aligned}
& A \times (B + C) \longrightarrow X \\
\cong\ & \text{[symmetry of products]} \\
& (B + C) \times A \longrightarrow X \\
\cong\ & \text{[currying]} \\
& B + C \longrightarrow A \supset X \\
\cong\ & \text{[uncotupling]} \\
& (B \longrightarrow A \supset X) \times (C \longrightarrow A \supset X) \\
\cong\ & \text{[uncurrying]} \\
& (B \times A \longrightarrow X) \times (C \times A \longrightarrow X) \\
\cong\ & \text{[cotupling]} \\
& (B \times A) + (C \times A) \longrightarrow X \\
\cong\ & \text{[symmetry of products]} \\
& (A \times B) + (A \times C) \longrightarrow X
\end{aligned}
$$

So there is an isomorphism of hom functors in $\mathbb{C} \supset \textsc{Set}$,

$$A \times (B + C) \longrightarrow - \quad \cong \quad (A \times B) + (A \times C) \longrightarrow -$$

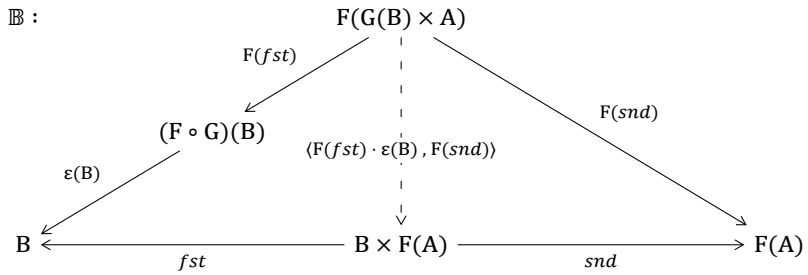
Applying the *Yoneda principle* reflects this isomorphism back into the category $\mathbb{C}$. □

The category of small bicartesian closed categories and structure-preserving functors between them is abbreviated "BCC". Every bicartesian closed category is already a cartesian closed category, and every bicartesian closed functor already a cartesian closed functor. One fact about cartesian closed functors that we will make use of is that when they have left adjoints, those left adjoints necessarily satisfy a distributive property of products over them, known as Frobenius reciprocity:

Lemma 3.3.3 (Frobenius reciprocity) For cartesian closed categories $\mathbb{A}$ and $\mathbb{B}$, exponential preserving functor $\mathsf{G} : \mathbb{B} \longrightarrow \mathbb{A}$ with left adjoint F, and objects $\mathsf{A} : \mathbb{A}$ and $\mathsf{B} : \mathbb{B}$,

$$\mathsf{B} \times \mathsf{F}(\mathsf{A}) \cong \mathsf{F}(\mathsf{G}(\mathsf{B}) \times \mathsf{A})$$

Proof. Note that the by virtue of the adjunction $\mathsf{F} \dashv \mathsf{G}$, the functor G must preserve products (lemma 3.1.6) and there is a canonical arrow in the reverse direction:



We could prove this arrow invertible by the Yoneda principle, as we did in the case of the distributive law, but instead we demonstrate a different technique.

The assumption that G preserves exponentials means that for any $\mathsf{B}, \mathsf{X} : \mathbb{B}$,

$$\mathsf{G}(\mathsf{B} \supset \mathsf{X}) \cong \mathsf{G}(\mathsf{B}) \supset \mathsf{G}(\mathsf{X})$$

So for any $\mathsf{B} : \mathbb{B}$, there is a natural isomorphism between functors,

$$(\mathsf{B} \supset -) \cdot \mathsf{G} \cong \mathsf{G} \cdot (\mathsf{G}(\mathsf{B}) \supset -)$$

Each of these functors has a left adjoint:

$$- \times \mathsf{B} \dashv \mathsf{B} \supset - \quad , \quad \mathsf{F} \dashv \mathsf{G} \quad , \quad - \times \mathsf{G}(\mathsf{B}) \dashv \mathsf{G}(\mathsf{B}) \supset -$$

By composition of adjunctions (lemma 3.1.8) we have:

$$\mathsf{F} \cdot (- \times \mathsf{B}) \dashv (\mathsf{B} \supset -) \cdot \mathsf{G} \quad \text{and} \quad (- \times \mathsf{G}(\mathsf{B})) \cdot \mathsf{F} \dashv \mathsf{G} \cdot (\mathsf{G}(\mathsf{B}) \supset -)$$

So by the uniqueness of adjoint functors up to natural isomorphism (lemma 3.1.5) we have:

$$\mathsf{F} \cdot (- \times \mathsf{B}) \cong (- \times \mathsf{G}(\mathsf{B})) \cdot \mathsf{F}$$

and so for any $\mathsf{A} : \mathbb{A}$,

$$\mathsf{F}(\mathsf{A}) \times \mathsf{B} \cong \mathsf{F}(\mathsf{A} \times \mathsf{G}(\mathsf{B}))$$

which is what we wanted to show, up to the symmetry of products. □

The reason that we have introduced bicartesian closed categories is that they provide suitable structure to interpret the propositional fragment of intuitionistic first-order logic. However, to interpret the rest of the first-order structure we will need to be a bit more creative.

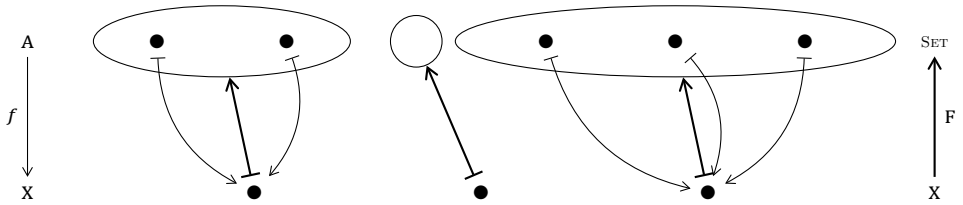


Figure 3.1: Two perspectives on dependence for indexed sets

3.4 Indexed Categories

A fundamental concept that recurs throughout mathematics is that of the **dependence** of one thing on another. A familiar instance is the idea of an indexed family of sets, known colloquially as an **indexed set**. It is well-known that any function $f : \mathsf{A} \longrightarrow \mathsf{X}$ determines a subset of its codomain in the form of its image. Dually, a function can be thought of as effecting a partition (or quotient) of its domain, where two elements are equivalent just in case they are mapped to the same value by the function. Each equivalence class, A_x, is a subset of A indexed by f from X:

$$\mathsf{A}_x \quad \coloneqq \quad \{a \in \mathsf{A} \mid f(a) = x\}$$

Given such a function f, we may always construct a function,

$$\begin{array}{ccc} & \underline{\mathsf{F}} & \\ \mathsf{X} & \longrightarrow & \wp(\mathsf{A}) \subseteq \textsc{Set} \\ x & \longmapsto & \mathsf{A}_x \end{array}$$

The image of F is the indexed set $\{\mathsf{A}_x\}_{x \in \mathsf{X}}$. We will write "$\int \mathsf{F}$" for the disjoint union of the image of F. Because f partitions its domain, $\int \mathsf{F} = \mathsf{A}$. On the other hand, if we begin with a function $\mathsf{F} : \mathsf{X} \longrightarrow \textsc{Set}$, we can "disjointify" the sets in its image by defining $\mathsf{F}'(x) \coloneqq (x, \mathsf{F}(x))$, so that $\int \mathsf{F} = \bigcup_{x \in \mathsf{X}} \mathsf{F}'(x)$. Then we may define a projection function,

$$\begin{array}{ccc} & \underline{f} & \\ \int \mathsf{F} & \longrightarrow & \mathsf{X} \\ (x\,, a) & \longmapsto & x \end{array}$$

The situation is summarized in figure 3.1. It should be clear that both perspectives describe the same relation of dependence on X.

A categorical generalization of the domain partitioning perspective of dependence is the concept of *fibration*, whereas generalizing the indexed set perspective results in the concept

of *indexed category*. In the sequel, we will adopt the indexed category perspective. As in the case for sets, the two perspectives are ultimately equivalent under suitable conditions (the magic word to utter here is "splitting"). As usually happens when trying to generalize a concept, there are several possible ways to proceed. Here is one that fits well to our purposes:

Definition 3.4.1 (indexed category) Let $\mathbb{X}$ be a category and $\mathcal{C}$ be a 2-category of categories. A (contravariant, strict) **$\mathbb{X}$-indexed $\mathcal{C}$-category** is a contravariant functor, $\mathsf{P} : \mathbb{X}^{\circ} \longrightarrow \mathcal{C}$.

When we wish to think of P as a functor, we will refer to it as the **indexing functor** of P (as indexed category). We call $\mathbb{X}$ the **base category** (or just "base") of P. For an object $\mathsf{X} : \mathbb{X}$, we call the category $\mathsf{P}(\mathsf{X})$ the **fiber** over X. For an arrow $s : \mathbb{X}(\mathsf{X} \to \mathsf{Y})$, we call $\mathsf{P}(s) : \mathsf{P}(\mathsf{Y}) \longrightarrow \mathsf{P}(\mathsf{X})$ the **reindexing functor** induced by s. When the indexed category under consideration is clear from context or irrelevant, it is customary to write "s^*" for $\mathsf{P}(s)$.

In the sequel, $\mathcal{C}$ will always be a subcategory of Cat (in fact, of BCC), so an indexing functor will map objects of the base to small categories of a given kind and arrows to contravariant structure-preserving functors between them. We will typically draw a diagram in an indexed category as diagrams in the fibers each positioned vertically above their respective indexing object in a diagram in the base category.

The categorical analogue of going from the indexed set perspective to the domain partitioning perspective of dependence involves building the *category of elements* of an indexed category by a process known as the "Grothendieck construction".[2] The category of elements generalizes the construction of $\int \mathsf{F}$ for sets, above.

Lemma 3.4.2 (category of elements) Let $\mathsf{P} : \mathbb{X}^{\circ} \longrightarrow \mathcal{C}$ be an indexed category. The following data define a category, called the **category of elements** of P, and written "$\int \mathsf{P}$".

objects: for objects $\mathsf{X} : \mathbb{X}$ and $\mathsf{A} : \mathsf{P}(\mathsf{X})$,

$$(\mathsf{X}, \mathsf{A}) \quad : \quad \int \mathsf{P}$$

arrows: for arrow $s : \mathbb{X}(\mathsf{X} \to \mathsf{Y})$, objects $\mathsf{A} : \mathsf{P}(\mathsf{X})$, $\mathsf{B} : \mathsf{P}(\mathsf{Y})$ and arrow $f : \mathsf{P}(\mathsf{X})(\mathsf{A} \to s^*(\mathsf{B}))$,

$$(s, f) \quad : \quad \int \mathsf{P}((\mathsf{X}, \mathsf{A}) \to (\mathsf{Y}, \mathsf{B}))$$

composition: for arrows $(s, f) : \int \mathsf{P}((\mathsf{X}, \mathsf{A}) \to (\mathsf{Y}, \mathsf{B}))$ and $(t, g) : \int \mathsf{P}((\mathsf{Y}, \mathsf{B}) \to (\mathsf{Z}, \mathsf{C}))$,

$$(s, f) \cdot (t, g) \quad \coloneqq \quad (s \cdot t, f \cdot s^*(g))$$

[2] Some authors restrict the term "category of elements" to the presheaf case ($\mathsf{P} : \mathbb{X}^{\circ} \longrightarrow$ Set), but since the term "Grothendieck construction" is almost completely uninformative, and in fact Grothendieck performed many constructions, we prefer the term with at least a bit of descriptive content.

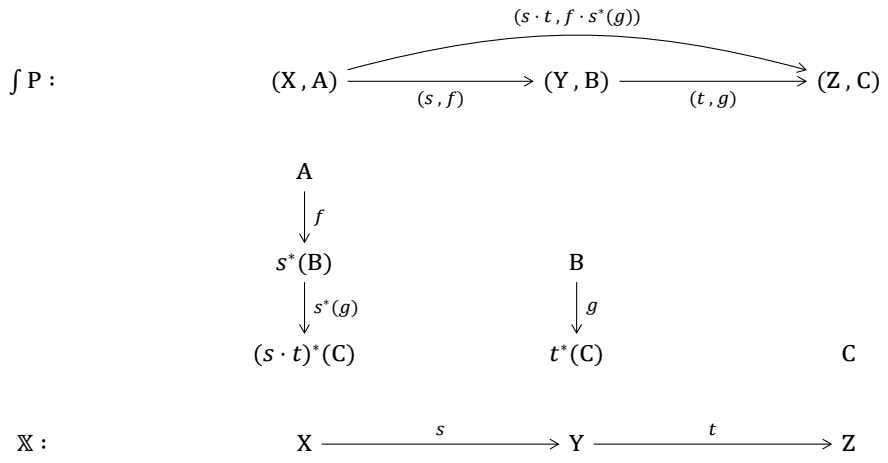


Figure 3.2: Composition in the category of elements of an indexed category

as shown in figure 3.2

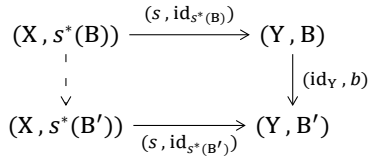
identities: for objects $\mathsf{X} : \mathbb{X}$ and $\mathsf{A} : \mathsf{P}(\mathsf{X})$,

$$\mathrm{id}_{(\mathsf{X},\mathsf{A})} \quad \coloneqq \quad (\mathrm{id}_{\mathsf{X}}, \mathrm{id}_{\mathsf{A}})$$

Constructing the category of elements allows us to gather an indexed category back into an ordinary category in a way that loses no information, thus we may recover an indexed category (up to isomorphism) from its category of elements. The objects and arrows of the base are given by first components of $\int \mathsf{P}$ objects and arrows. Objects within a fiber are given by second components of objects of $\int \mathsf{P}$. Arrows within a fiber are given by second components of arrows of $\int \mathsf{P}$ whose first components are identity arrows (so-called "vertical morphisms"). The action of reindexing functors on objects is given by the second components of the domains of $\int \mathsf{P}$-arrows whose second components are identity arrows; that is, if $(s, \mathrm{id}) :: \int \mathsf{P}$ has codomain (Y, B), then it is necessarily of the form,

$$(s, \mathrm{id}_{s^*(\mathsf{B})}) : (\mathsf{X}, s^*(\mathsf{B})) \longrightarrow (\mathsf{Y}, \mathsf{B})$$

Finally, given $b : \mathsf{P}(\mathsf{Y})\,(\mathsf{B} \to \mathsf{B}')$ and $s : \mathbb{X}\,(\mathsf{X} \to \mathsf{Y})$, we may recover $s^*(b)$ as the second component of the unique arrow making the following diagram in $\int \mathsf{P}$ commute:



This arrow must be $(\mathsf{id}_X , s^*(b))$ by the functoriality of s^*. The isomorphism between an indexed category and the indexed category recovered from its category of elements is a component of the (categorical) equivalence between the 2-category of indexed categories and that of split fibrations, which forms the basis of the (logical) equivalence of the two perspectives on dependence mentioned above.

Categories of elements can be used to apply constructions on categories to indexed categories. They may also be used to represent iterated dependence relationships. For example, they have been proposed in [FFL03] as interpretations for abstract notions of state in the course of proof search.

We now address the two adjectives that were snuck into the definition of indexed category above. The choice of contravariance may seem odd at first, but it is the natural choice for representing operations that act on *inputs* as opposed to on *outputs*. In our case, the dependence that we want to represent is that of logical propositions and derivations on their free term variables and the operation mediating between these is the substitution of a term for a free variable in another term.

The issue of strictness is more complex. Many naturally-occurring constructions that would otherwise be indexed categories fail to be because they are not strict. For example, a category $\mathbb{C}$ with pullbacks has a natural indexed structure over itself given by the slice construction (or "codomain fibration"): for $\mathtt{A} : \mathbb{C}$, the category $\mathbb{C}/\mathtt{A}$ has $\mathbb{C}$-arrows with codomain $\mathtt{A}$ for objects and commuting triangles for arrows. Since the pullback of a commuting triangle is again a commuting triangle, pullback along an arrow provides a contravariant reindexing operation. But it is not strict: pullback preserves composition and identity arrows only up to isomorphism. In order to accommodate this, we need the notion of a **pseudofunctor**, where composition and identity are not "on the nose", but rather "up to isomorphism". Indeed, a construction equivalent to that of pseudofunctors is adopted by default in the fibrational perspective mentioned above.

In general, the fibrational approach has two key advantages to the indexed approach. First, it avoids making potentially arbitrary choices since all definitions are purely in terms of universal properties. This is particularly pleasing to categorists for being "non-evil", that is, for respecting the *principle of equivalence* [nla]. Second, it extends naturally to systems involving more complex relations of dependence (which we do not consider here), either through incorporation of dependent type theories, or through allowing types to depend on propositions (e.g. $\{n : \mathbb{N} \mid \mathrm{prime}(n)\}$), in which case there is no longer any reason to consider propositions as separate from types. The indexed approach turns out to be less flexible in this setting.

On the other side of the ledger, the indexed-categorical approach is generally easier to understand and work with. Categories and functors have by now become familiar tools

in computer science, whereas cartesian morphisms and cleavages still remain rather less-so. Many of the results needed to ensure the preservation of the desired structure follow directly from structure-preserving functoriality in the indexed-categorical setting, but require non-trivial arguments in the fibrational setting. But convenience is not the only advantage of the indexed approach. The advantage that the fibrational approach has of not specifying canonical structures becomes a disadvantage for the interpretation of syntax, where we find that we need more, not less, *strictness*. For example, if A is a proposition and t a term, we will want the interpretation of $\mathsf{A}[x \mapsto t]$ to actually *be* the effect of the interpretation of the substitution on the interpretation of the proposition, and not just something isomorphic to it. It is precisely the strict *functoriality* of the indexed approach which permits this **soundness for syntax**. We will not pursue the fibrational approach further here and refer the interested reader to [Jac99] for details.

Although we also will not pursue here categorical notions of semantics, in the sense of model theory, we mention that the concept of indexed category supports a notion of functor, which may be used for the interpretation of such semantics:

Definition 3.4.3 (indexed functor) For indexed categories $\mathsf{P} : \mathbb{A}^{\circ} \longrightarrow \mathcal{C}$ and $\mathsf{Q} : \mathbb{B}^{\circ} \longrightarrow \mathcal{C}$ an **indexed functor** from P to Q is a base change functor $\mathsf{F} : \mathbb{A} \longrightarrow \mathbb{B}$ together with a natural transformation $\tau : \mathsf{P} \longrightarrow \mathsf{F}^{\circ} \cdot \mathsf{Q}$.

A notable special case is when F is an identity functor. Such an indexed functor over a fixed base is used in [FFL03] to give a categorical presheaf semantics for a logic programming language with implication in goals. Another special case occurs when τ is an identity natural transformation. Such an indexed functor, where the underlying base change functor has a left adjoint, is used in [Kri05] to define a Kan extension, which is then used to extend the categorical semantics of [FFL03] to accommodate universally quantified goals.

Chapter 4

Categorical Intuitionistic First-Order Logic

4.1 The Categorical Interpretation of Logic

The basic idea of categorical logic can be summarized like this. Beginning from a logical system that we are interested in, we endeavor to interpret its propositional formulas as objects in categories and its valid inferences as arrows between these objects. In order to identify a suitable sort of category in which to do this, we look for universal constructions from category theory that correspond to features of the logical system. These universal constructions generate collections of arrows, which we use to interpret the inference schemes of the logic. In addition, they generally impose certain relations, most notably equations on the arrows of the category. A candidate interpretation is **sound** if every valid inference in the logic is interpreted as a well-defined arrow in the category from the interpretation of its premises to that of its conclusion. Soundness is an essential feature of an interpretation, and when we speak of interpretations, we mean sound ones, although of course, the soundness is something that must be proved.

Once we have found a suitable sort of category in which to interpret the logical system, we may take any theory (i.e. collection of propositional formulas) from the logic and interpret its language in any category of the given sort. To do so, we must assign objects of the category to propositions and arrows to valid inferences in a consistent way. We may do this by providing a function that takes each atomic proposition in the language of the theory to an object of the category. This is because the universal constructions of the category that interpret the features of the logical system will then inductively provide interpretations for the rest of the structure. If M is an interpretation of language $\mathcal{L}$ in category $\mathbb{C}$, then we indicate the M interpretation in $\mathbb{C}$ of an element of $\mathcal{L}$, such as a term or proposition, by "$[\![-]\!]_{\mathsf{M}}$". And if J is some judgement regarding elements of $\mathcal{L}$, such as an inference or

logic	categories
logical system	category of categories
theory	free-interpretation category
model	structure-preserving functor
proposition	object
inference	arrow
connective	universal construction

Figure 4.1: the categorical interpretation of logic

equation, then we indicate that M satisfies J by "$\mathsf{M} \vDash \mathsf{J}$".

In particular, we can define an interpretation in the category of the given sort that is freely generated by the atomic propositions of the language. Such a category will contain only objects corresponding to these propositions and such other objects, arrows and relations among them as are required to exist by the universal constructions in the sort of category we have chosen. We call such an interpretation a **free interpretation**. A free interpretation will generally have the universal property that for any interpretation of the language in any category of the given sort there will be a unique structure-preserving functor from the free-interpretation category to the other category such that when the free interpretation is composed with this functor the result will be just the other interpretation. We usually omit the interpretations brackets, $[\![-]\!]$, when discussing free interpretations, as writing them quickly becomes tedious. We summarize this categorical interpretation of logic in figure 4.1.

We may consider the free-interpretation category to be the representation of a logical theory within the suitably chosen category of categories. Any inference that holds in the logic will also hold in the free-interpretation category (indeed, in any sound interpretation) in the sense that it will be interpreted by an arrow in the corresponding hom set. However, it is not necessarily the case that every arrow in the free-interpretation category between interpretations of propositions is the interpretation of a distinct logical inference. In order for this to hold, we must have chosen a sort of category with just the right structure, corresponding to universal constructions, needed to interpret the inference schemes of the given logical system. Such an interpretation is said to be **generic**. Generic interpretations are very useful because they allow us to forget about the logic completely and work entirely within category theory, safe in the knowledge that any results we obtain there can be translated back to the logic. Thus, given a logical system, the challenge is to identify a category of categories such that valid inferences for theories correspond precisely to arrows in their free interpretations.

A good overview of this perspective on categorical logic may be found in [AB09]. The case of intuitionistic logic is treated in more detail by Lambek and Scott [LS86] and Makkai [Mak93a; Mak93b]. We now proceed to describe a categorical interpretation of intuitionistic

first-order logic, called a "hyperdoctrine" interpretation, by building it up in stages.

4.2 Interpreting the Propositional Connectives

We begin with propositional logic. This is the language inductively constructed from atomic propositions, in the form of unexamined propositional variables, using the propositional connectives $\{\wedge, \vee, \top, \bot, \supset\}$. In the classic *Heyting algebra* interpretation, which models the behavior of the *consequence relation* ($\vdash$), these connectives are interpreted in a partially-ordered set by meet, join, top, bottom and Heyting implication, respectively. Using a common method of generalizing a preorder to a category, we may interpret them respectively as the cartesian product, coproduct, terminal object, initial object and exponential of a *bicartesian closed category*.

Definition 4.2.1 (interpretation of atomic propositions) For an arbitrary set of atomic propositions, $\mathcal{P}$, an interpretation of $\mathcal{P}$ in a category $\mathbb{C}$ is a function $[\![-]\!]$ sending atomic propositions to objects. That is, for $\mathrm{P} \in \mathcal{P}$,

$$[\![\mathrm{P}]\!] \quad : \quad \mathbb{C}$$

Definition 4.2.2 (interpretation of propositions) An interpretation of atomic propositions in a bicartesian closed category, $[\![-]\!]_{\mathcal{P}}$, extends inductively to an interpretation of propositions by the interpretations of the propositional connectives:

$$\begin{aligned}
[\![\mathrm{P}]\!] &\coloneqq [\![\mathrm{P}]\!]_{\mathcal{P}} \qquad \text{for } \mathrm{P} \in \mathcal{P} \\
[\![\mathrm{A} \wedge \mathrm{B}]\!] &\coloneqq [\![\mathrm{A}]\!] \times [\![\mathrm{B}]\!] \\
[\![\mathrm{A} \vee \mathrm{B}]\!] &\coloneqq [\![\mathrm{A}]\!] + [\![\mathrm{B}]\!] \\
[\![\mathrm{A} \supset \mathrm{B}]\!] &\coloneqq [\![\mathrm{A}]\!] \supset [\![\mathrm{B}]\!] \\
[\![\top]\!] &\coloneqq 1_{\mathbb{C}} \\
[\![\bot]\!] &\coloneqq 0_{\mathbb{C}}
\end{aligned}$$

In addition to interpretations for individual propositions, we will need interpretations for *propositional contexts*, which we met in the preceding discussions of proof theory. In the meta-theory of this logic, assumptions are unordered and may be appealed to any number of times, including possibly zero. We will see that this implies that a propositional context can be interpreted as a finite product of the interpretations of the propositions it contains.

Definition 4.2.3 (interpretation of propositional contexts) An interpretation of propositions in a bicartesian closed category, $[\![-]\!]_{\mathcal{P}}$, extends inductively to an interpretation of propositional contexts by finite products:

$$\begin{aligned}
[\![\emptyset]\!] &\coloneqq 1 \\
[\![\Gamma\,, \mathrm{P}]\!] &\coloneqq [\![\Gamma]\!] \times [\![\mathrm{P}]\!]_{\mathcal{P}}
\end{aligned}$$

We may form the free interpretation of any propositional language in the category of bicartesian closed categories.

Definition 4.2.4 (free interpretation of an intuitionistic propositional language) For $\mathcal{P}$ a set of atomic propositions, we define $\textsc{Prop}_{\mathcal{P}}$ to be the free bicartesian closed category generated by $\mathcal{P}$ (as objects) and the free interpretation of the propositional language over $\mathcal{P}$ to be the interpretation taking each atomic proposition to itself, regarded as an object of $\textsc{Prop}_{\mathcal{P}}$.

4.3 Interpreting the Term Language

In predicate logic an atomic proposition is not an unanalyzed propositional variable, rather, it is a *predicate* constructed from a relation symbol and terms. Before we can give a categorical interpretation for predicate logic, we must first interpret the language of terms. In order to concentrate on the logic we will use a very simple language of typed terms. Our type theory will have no type constructors and our term language will have no (object-level) term constructors. However, we will still have the meta-level operation of substituting a term for a variable in a term. The only (meta-level) relation on our terms is syntactic equality. Thus, we construct a so-called "language of uninterpreted terms".[1] This is what is typically desired in logic programming, although not in other contexts such as constraint logic programming or automated theorem proving.

Definition 4.3.1 (interpretation of atomic types) For an arbitrary set of atomic types, $\mathcal{T}$, an interpretation of $\mathcal{T}$ in a category $\mathbb{C}$ is a function $[\![-]\!]$ sending atomic types to objects. That is, for $\mathtt{X} \in \mathcal{T}$,

$$[\![\mathtt{X}]\!] \quad : \quad \mathbb{C}$$

A **type** over $\mathcal{T}$ is any expression that can be freely generated from the atomic types using the type-forming operations of a type theory. So for our purposes, the atomic types will be the only types.

As in the case for propositions, we will need to interpret not only individual types but also contexts of types. Refining our previous description, we will say that a *typing context* is a finite sequence of types:

$$\mathtt{X}_1 , \cdots , \mathtt{X}_n$$

Equivalently, it is an open α-equivalence class of finite sequences of distinct typed variables:

$$x_1 : \mathtt{X}_1 , \cdots , x_n : \mathtt{X}_n \qquad \text{such that } x_i = x_j \Longrightarrow i = j$$

[1] Note that the concept of interpretation in "uninterpreted terms" is distinct to that in "categorical interpretation". Due to an unfortunate coincidence of terminology, we will be defining the latter for the former.

The variables simply act as indices into the sequence. Although the latter description comports better with the syntactic presentation of languages, it is useful to recognize that the role of a variable in a language is simply to *refer*, a point of view eloquently advocated in [AM89] and [Tay99].

Definition 4.3.2 (interpretation of typing contexts) An interpretation of types in a cartesian category, $[\![-]\!]_{\mathcal{T}}$, extends inductively to an interpretation of typing contexts by finite products:

$$\begin{array}{rcl} [\![\emptyset]\!] & \coloneqq & 1 \\ [\![\Phi\,, x : \mathtt{X}]\!] & \coloneqq & [\![\Phi]\!] \times [\![\mathtt{X}]\!]_{\mathcal{T}} \end{array}$$

A fundamental operation on typing contexts is that of **weakening**, which allows us to forget about some of the variables in scope. For this we adopt terminology and notation introduced by Taylor in [Tay99]. A special case of context weakening is that of a **single omission**, which lets us forget about just the last element of the context:

$$\hat{x} \quad : \quad \Phi\,, x : \mathtt{X} \;\mapsto\; \Phi$$

Since our type system is not *dependent*, the well-formedness of each type in a context is independent of any other context members. Therefore, we may always safely permute typing contexts. When no confusion is likely to result, we will do this tacitly. Under permutation, all context weakenings are generated by the single omissions. Single omissions are interpreted by complement-projections:

$$[\![\hat{x}]\!] \quad \coloneqq \quad \pi : [\![\Phi\,, x : \mathtt{X}]\!] \longrightarrow [\![\Phi]\!]$$

A signature for a language of typed terms can be given by providing a set of atomic types, $\mathcal{T}$, together with a collection of typed **function symbols**, $\mathcal{F}$, each of the form:

$$f \quad \in \quad \mathcal{F}(\mathtt{Y}_1\,, \cdots, \mathtt{Y}_n\,;\, \mathtt{X})$$

where each $\mathtt{Y}_i$ is a type and $\mathtt{X}$ is an atomic type. The intended meaning is that when the function symbol f is applied to terms of types $\mathtt{Y}_1\,, \cdots, \mathtt{Y}_n$ the result is a term of type $\mathtt{X}$. We follow the usual convention of considering a constant symbol c of type $\mathtt{X}$ to be a function symbol $c \in \mathcal{F}(\emptyset\,;\, \mathtt{X})$.

Definition 4.3.3 (interpretation of function symbols) For $[\![-]\!]_{\mathcal{T}} : \mathcal{T} \longrightarrow \mathbb{C}$ an interpretation of typing contexts in a cartesian category and $\mathcal{F}$ a collection of function symbols over $\mathcal{T}$, an interpretation of $\mathcal{F}$ in $\mathbb{C}$ is a function $[\![-]\!]$ sending function symbols to arrows between the interpretations of their argument context and result type. That is, for $f \in \mathcal{F}(\mathtt{Y}_1\,, \cdots, \mathtt{Y}_n\,;\, \mathtt{X})$,

$$[\![f]\!] \quad : \quad \mathbb{C}\,([\![\mathtt{Y}_1\,, \cdots, \mathtt{Y}_n]\!]_{\mathcal{T}} \to [\![\mathtt{X}]\!]_{\mathcal{T}})$$

Mirroring the inductive construction of terms from function symbols and variables, we can inductively define their interpretations in a cartesian category. We want to interpret not only closed terms (those without free variables), but open ones as well. To do so, we associate to each term a typing context, which must contain at least the free variables that occur in the term. Thus we form **terms in context** and write "$\Phi \mid t : \mathrm{X}$" to indicate that term t has type X in context Φ. After having done so, we may subsequently refer to this term as just "t", unless we wish to discuss the same term in a different context as well. Given a syntactic term, we can always determine its native (minimal) context by consulting its set of free variables. However, it is also sensible to consider a term in any context extending its native context; the superfluous **dummy variables** simply play no role in the term.

A term is interpreted as an arrow from the interpretation of its context to that of its type.

Definition 4.3.4 (interpretation of terms) Let $\mathcal{L} \coloneqq (\mathcal{T}, \mathcal{F})$ be the signature of a language of typed terms, $[\![-]\!]_{\mathcal{T}}$ be an interpretation of typing contexts over $\mathcal{T}$ in a cartesian category $\mathbb{C}$ and $[\![-]\!]_{\mathcal{F}}$ be an interpretation of its function symbols in $\mathbb{C}$. Then $[\![-]\!]_{\mathcal{F}}$ extends inductively to an interpretation of terms in context by precomposition.

lifted variable: for variable $x_i \in \Phi$,

$$[\![\Phi \mid x_i : \mathrm{X}_i]\!] \quad \coloneqq \quad \pi_i \;:\; [\![\Phi]\!]_{\mathcal{T}} \longrightarrow [\![\mathrm{X}_i]\!]_{\mathcal{T}}$$

applied function symbol: for function symbol $f \in \mathcal{F}(\mathrm{Y}_1, \cdots, \mathrm{Y}_n ; \mathrm{X})$ and terms $\Phi \mid t_1 : \mathrm{Y}_1, \cdots, t_n : \mathrm{Y}_n$,

$$[\![\Phi \mid f(t_1, \cdots, t_n) : \mathrm{X}]\!] \quad \coloneqq \quad \langle [\![t_1]\!], \cdots, [\![t_n]\!] \rangle \cdot [\![f]\!]_{\mathcal{F}} \;:\; [\![\Phi]\!]_{\mathcal{T}} \longrightarrow [\![\mathrm{X}]\!]_{\mathcal{T}}$$

context extension: for term $\Phi \mid t : \mathrm{X}$ and variable $x \notin \Phi$,

$$[\![\Phi, x : \mathrm{X} \mid t : \mathrm{X}]\!] \quad \coloneqq \quad [\![\hat{x}]\!]_{\mathcal{T}} \cdot [\![t]\!] \;:\; [\![\Phi, x : \mathrm{X}]\!]_{\mathcal{T}} \longrightarrow [\![\mathrm{X}]\!]_{\mathcal{T}}$$

substitution: for terms $\Phi, y : \mathrm{Y} \mid t : \mathrm{X}$ and $\Phi \mid s : \mathrm{Y}$,

$$[\![\Phi \mid t[y \mapsto s] : \mathrm{X}]\!] \quad \coloneqq \quad \langle \mathrm{id}_{[\![\Phi]\!]_{\mathcal{T}}}, [\![s]\!] \rangle \cdot [\![t]\!] \;:\; [\![\Phi]\!]_{\mathcal{T}} \longrightarrow [\![\mathrm{X}]\!]_{\mathcal{T}}$$

A substitution of just one term for a variable is a **single substitution**. We call the precomposed arrow in the definition of a term under single substitution the interpretation of the single substitution:

$$[\![\, [y \mapsto s] \,]\!] \quad \coloneqq \quad \langle \mathrm{id}_{[\![\Phi]\!]}, [\![s]\!] \rangle$$

The reason that it tuples the interpretation of the substituting term with that of its context is that in the language, applying such a substitution to a term leaves undisturbed any other

variables that may occur in the term. We note in passing that the definition of substitution for terms is just the composition of their interpretations in the *Kleisli category* of the $\Phi \times -$ *comonad*:

$$[\![\Phi \mid t[y{\mapsto}s] : \mathrm{X}]\!] = [\![s]\!] \cdot [\![t]\!] : \mathbb{C}_{\Phi\times -}\,(1 \to \mathrm{X})$$

We will have more to say about this particular comonad in the context of logical derivations in chapter 6.

There is a standard result known as the "substitution lemma", which describes properties of the substitution operation. A corollary of this lemma is that the operation of simultaneous substitution is well-defined.

Lemma 4.3.5 (substitution lemma) For terms $a\,,\,b\,,\,c$ and distinct variables $x\,,\,y$, if $x\,,\,y \notin \mathrm{FV}(a)$ and $y \notin \mathrm{FV}(b)$ then,

- $a[y{\mapsto}b] = a$ and
- $c[y{\mapsto}b][x{\mapsto}a] = c[x{\mapsto}a][y{\mapsto}b[x{\mapsto}a]]$

Proposition 4.3.6 (categorical substitution and terms) The given interpretation of substitution respects the substitution lemma. That is, for terms as in the lemma and suitable contexts,

- $[\![a[y{\mapsto}b]]\!] = [\![a]\!]$ and
- $[\![c[y{\mapsto}b][x{\mapsto}a]]\!] = [\![c[x{\mapsto}a][y{\mapsto}b[x{\mapsto}a]]]\!]$

Proof.

- Let Φ be a context containing the free variables of both a and b. By hypothesis, $y \notin \Phi$. Then by definition, $[\![\Phi \mid a[y{\mapsto}b]]\!]$ is the composition:

$$[\![\Phi]\!] \xrightarrow{[\![\,[y\mapsto b]\,]\!]} [\![\Phi\,,\,y : \mathrm{Y}]\!] \xrightarrow{[\![\hat{y}]\!]} [\![\Phi]\!] \xrightarrow{[\![a]\!]} [\![\mathrm{X}]\!]$$

  but $[\![[y{\mapsto}b]]\!] \cdot [\![\hat{y}]\!] = \langle \mathrm{id}_{[\![\Phi]\!]}\,,\,[\![b]\!]\rangle \cdot \pi_{[\![\Phi]\!]} = \mathrm{id}_{[\![\Phi]\!]}$.

- Let Φ be a context such that the following terms are well-defined:

$$\Phi \mid a : \mathrm{X} \qquad \Phi\,,\,x : \mathrm{X} \mid b : \mathrm{Y} \qquad \Phi\,,\,x : \mathrm{X}\,,\,y : \mathrm{Y} \mid c : \mathrm{Z}$$

  By definition, $[\![\Phi \mid c[y{\mapsto}b][x{\mapsto}a]]\!]$ is the composition:

$$[\![\Phi]\!] \xrightarrow{[\![\,[x\mapsto a]\,]\!]} [\![\Phi\,,\,x : \mathrm{X}]\!] \xrightarrow{[\![\,[y\mapsto b]\,]\!]} [\![\Phi\,,\,x : \mathrm{X}\,,\,y : \mathrm{Y}]\!] \xrightarrow{[\![c]\!]} [\![\mathrm{Z}]\!]$$

And $[\![\Phi \mid c[x{\mapsto}a][y{\mapsto}b[x{\mapsto}a]]]\!]$ is the composition:

$$[\![\Phi]\!] \xrightarrow{[\![\,[y\mapsto b[x\mapsto a]]\,]\!]} [\![\Phi\,, y : \mathrm{Y}]\!] \xrightarrow{[\![\,[x\mapsto a]\,]\!]} [\![\Phi\,, y : \mathrm{Y}\,, x : \mathrm{X}]\!] \xrightarrow{[\![c]\!]} [\![\mathrm{Z}]\!]$$

up to a permutation on the context of c, where $[\![b[x{\mapsto}a]]\!]$ is itself the composition:

$$[\![\Phi]\!] \xrightarrow{[\![\,[x\mapsto a]\,]\!]} [\![\Phi\,, x : \mathrm{X}]\!] \xrightarrow{[\![b]\!]} [\![\mathrm{Y}]\!]$$

Since c is arbitrary, we check that $[\![\,[x{\mapsto}a]\,]\!] \cdot [\![\,[y{\mapsto}b]\,]\!]$ is equal to $[\![\,[y{\mapsto}b[x{\mapsto}a]]\,]\!] \cdot [\![\,[x{\mapsto}a]\,]\!]$ modulo the permutation. A routine computation shows that both compositions are equal to the tuple

$$\langle \mathrm{id}_{[\![\Phi]\!]}\,, [\![a]\!]\,, \langle \mathrm{id}_{[\![\Phi]\!]}\,, [\![a]\!]\rangle \cdot [\![b]\!]\rangle$$

□

We may form the free interpretation of any term language in the category of cartesian categories.

Definition 4.3.7 (free interpretation of a term language) If $\mathcal{L} \coloneqq (\mathcal{T}\,, \mathcal{F})$ is a signature for a typed term language then we define $\textsc{Type}_{\mathcal{L}}$ to be the free cartesian category generated by $\mathcal{F}$ (as arrows) and the *free interpretation* of $\mathcal{L}$ in $\textsc{Type}_{\mathcal{L}}$ to be the one taking each function symbol to itself, regarded as an arrow of $\textsc{Type}_{\mathcal{L}}$.

4.4 Interpreting Predicates

A signature for a language of typed predicate logic is formed by adding to the signature of a term language a collection of typed **relation symbols**, $\mathcal{R}$, each of the form:

$$\mathrm{R} \quad \in \quad \mathcal{R}(\mathrm{X}_i\,, \cdots, \mathrm{X}_n)$$

where each X_i is a type. The intended meaning is that when the relation symbol R is applied to terms of types $\mathrm{X}_i\,, \cdots, \mathrm{X}_n$ the result is an atomic proposition, or **predicate**.

We want to interpret not only closed predicates (those without free term variables), but open ones as well. Thus predicates are *dependent* on their free term variables. This suggests that we interpret predicates in an *indexed category* over a base category interpreting their typing contexts.

Definition 4.4.1 (interpretation of relation symbols) For $[\![-]\!]_{\mathcal{T}} : \mathcal{T} \longrightarrow \mathbb{C}$ an interpretation of typing contexts in a cartesian category and $\mathcal{R}$ a collection of relation symbols over $\mathcal{T}$, an interpretation of $\mathcal{R}$ in an indexed category $\mathrm{P} : \mathbb{C}^{\circ} \longrightarrow \mathcal{D}$ is a function $[\![-]\!]$ sending relation

symbols to objects in the fibers over the interpretations of their typing contexts. That is, for $\mathrm{R} \in \mathcal{R}(\mathrm{X}_1, \cdots, \mathrm{X}_n)$,

$$[\![\mathrm{R}]\!] \quad : \quad \mathrm{P}([\![\mathrm{X}_1, \cdots, \mathrm{X}_n]\!]_{\mathcal{T}})$$

Mirroring the inductive construction of predicates from relation symbols and terms, we can inductively define their interpretations in an indexed category. As in the case for terms, we want to keep track of the typing context in which a predicate, and in general, a proposition, occurs. Thus we form **propositions in context** and write "$\Phi \mid \mathrm{A}$ PROP" to indicate that A is a proposition, all of whose free variables are contained in the typing context Φ. Just as with terms, it is sensible to consider a proposition in any extension of its native context. The superfluous *dummy variables* simply play no role in the proposition.

A predicate is interpreted as an object in the fiber over the interpretation of its context.

Definition 4.4.2 (interpretation of predicates) Let $\mathcal{L} \coloneqq (\mathcal{T}, \mathcal{F}, \mathcal{R})$ be the signature of a typed predicate language, $[\![-]\!]_{\mathcal{T}}$ be an interpretation of typing contexts over $\mathcal{T}$ and $[\![-]\!]_{\mathcal{F}}$ be an interpretation of terms over $\mathcal{F}$ in a cartesian category $\mathbb{C}$. Let $[\![-]\!]_{\mathcal{R}}$ be an interpretation of the relation symbols in an indexed category $\mathrm{P} : \mathbb{C}^{\circ} \longrightarrow \mathcal{D}$. Then $[\![-]\!]_{\mathcal{R}}$ extends inductively to an interpretation of predicates in context by reindexing.

applied relation symbol: for relation symbol $\mathrm{R} \in \mathcal{R}(\mathrm{Y}_1, \cdots, \mathrm{Y}_n)$ and terms $\Phi \mid t_1 : \mathrm{Y}_1, \cdots, t_n : \mathrm{Y}_n$,

$$[\![\Phi \mid \mathrm{R}(t_1, \cdots, t_n) \text{ PROP}]\!] \quad \coloneqq \quad \langle [\![t_1]\!]_{\mathcal{F}}, \cdots, [\![t_n]\!]_{\mathcal{F}} \rangle^* \, ([\![\mathrm{R}]\!]_{\mathcal{R}}) \; : \; \mathrm{P}([\![\Phi]\!]_{\mathcal{T}})$$

context extension: for predicate $\Phi \mid \mathrm{A}$ PROP and variable $x \notin \Phi$,

$$[\![\Phi, x : \mathrm{X} \mid \mathrm{A} \text{ PROP}]\!] \quad \coloneqq \quad [\![\hat{x}]\!]^*_{\mathcal{T}} \, ([\![\mathrm{A}]\!]) \; : \; \mathrm{P}([\![\Phi, x : \mathrm{X}]\!]_{\mathcal{T}})$$

substitution: for predicate $\Phi, y : \mathrm{Y} \mid \mathrm{A}$ PROP and term $\Phi \mid s : \mathrm{Y}$,

$$[\![\Phi \mid \mathrm{A}[y \mapsto s] \text{ PROP}]\!] \quad \coloneqq \quad [\![\, [y \mapsto s] \,]\!]^*_{\mathcal{F}} \, ([\![\mathrm{A}]\!]) \; : \; \mathrm{P}([\![\Phi]\!]_{\mathcal{T}})$$

Note that in each case, the arrow in the base used to reindex is the same one that is precomposed in definition 4.3.4. We can form free interpretations of typed predicate languages as well.

Definition 4.4.3 (free interpretation of a predicate language) If $\mathcal{L} \coloneqq (\mathcal{T},\mathcal{F},\mathcal{R})$ is a signature for a typed predicate language and $\textsc{Type}_{\mathcal{L}}$ is the free cartesian category generated by $\mathcal{F}$, then we define $\textsc{Pred}_{\mathcal{L}} : \textsc{Type}_{\mathcal{L}}{}^{\circ} \longrightarrow \mathcal{D}$ to be the free indexed category of sort $\mathcal{D}$ generated by $\mathcal{R}$ and the *free interpretation* of $\mathcal{L}$ in $\textsc{Pred}_{\mathcal{L}}$ to be the one taking each relation symbol to itself, regarded as an object in the fiber over its typing context.

This is essentially the categorical interpretation of predicates used in [AM89] and [KP96] to model the operational semantics of logic programming for Horn logic, which can be formulated without connectives.

4.5 Interpreting Quantification

It was Lawvere who observed that the quantifiers act as adjoints to reindexing in indexed categories [Law69]. This characterization permits the quantifiers to be treated algebraically, rather than merely as infinitary conjunctions and disjunctions. Interestingly, it also permits for the definition of quantification over, not just bound variables, but rather a broader class of bound terms. The familiar case of quantification over a bound variable is that where the reindexing is induced by a *single omission*.

However, the adjoint characterization is not all there is to quantification. There is also the interaction between quantification and substitution. It turns out that this adds significant complexity to the categorical treatment of quantification, as was observed by Lawvere in [Law70] and investigated systematically by Seely in [See83]. Although there is much more that could be said about the matter, in order to interpret first-order logic we need consider only the familiar case of quantification over bound variables, where the situation is comparatively simple. First we formalize the conditions necessary for a class of arrows in a category to support quantifiers:

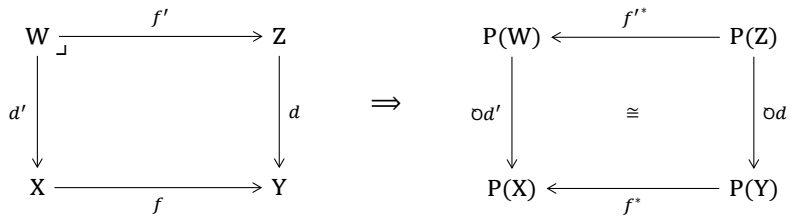
Definition 4.5.1 (having quantifiers) Let $\mathsf{P} : \mathbb{C}^{\circ} \longrightarrow \textsc{Cat}$ be an indexed category and $\mathcal{D}$ be a collection of $\mathbb{C}$-arrows. We will say that P has quantifiers for arrows in $\mathcal{D}$ if:

- P-images of $\mathcal{D}$-arrows have left and right adjoints:

$$d : \mathbb{C}\,(Z \to Y) \quad \in \quad \mathcal{D} \qquad \Longrightarrow \qquad \exists\, \Sigma d\,, \Pi d : \mathsf{P}(Z) \longrightarrow \mathsf{P}(Y) \;\;.\;\; \Sigma d \dashv d^{*} \dashv \Pi d$$

- the set $\mathcal{D}$ is closed under pullbacks with arbitrary $\mathbb{C}$-arrows; that is, for $d \in \mathcal{D}$ and coterminal $f :: \mathbb{C}$, there is a pullback of d and f, and the f pullback of d is again in $\mathcal{D}$.

- the P-image of a pullback with a $\mathcal{D}$-arrow satisfies the *Beck-Chevalley condition* (definition 3.1.11). This implies:



  for $d \in \mathcal{D}$ and $\mathfrak{O} \in \{\Sigma\,, \Pi\}$.

When P has quantifiers for an arrow d, we will write "$\exists d$" and "$\forall d$" for Σd and Πd, respectively.

Essentially, the first condition guarantees that functors Σd and Πd have the universal properties of the quantifiers, while the second and third conditions ensure that quantification is compatible with substitution, as we're about to see.

4.6 The Hyperdoctrine Interpretation

We have now assembled all the pieces necessary to define the type of category in which we want to interpret typed intuitionistic first-order logic.

Definition 4.6.1 (hyperdoctrine) A **hyperdoctrine** is an indexed bicartesian closed category over a cartesian base, that has quantifiers for projections.

The term "hyperdoctrine" was introduced by Lawvere in [Law69] to describe a slightly different indexed category. Makkai [Mak93a] used the term "Heyting fibration" for a similar concept. Sometimes the term "Heyting category" is also used, though often with the assumption that the fibers are posets. Anyway, we must pick a word to use here, and we have done so, but the choice has been fairly capricious. Because the purpose of a hyperdoctrine is to interpret a typed intuitionistic first-order logic, the precise definition must untimately depend on the particular type theory and logic involved. In the sequel, we will see that this definition is consistent with the choices we have made.

We are now nearly ready to give the definition of interpretation of a first-order language in a hyperdoctrine. There remains just one technical issue in our path, the issue of *strictness*. We would like to interpret substitution on propositions by reindexing between fibers. Indeed, for atomic propositions, we have already done so by definition 4.4.2. But there arises a technical complication. In logic it is expected that the propositional connectives commute "on the nose" with substitutions. That is, for propositions A and B and term t, we have:

$$\begin{aligned}
(\mathsf{A} \wedge \mathsf{B})[x{\mapsto}t] &= \mathsf{A}[x{\mapsto}t] \ \wedge \ \mathsf{B}[x{\mapsto}t] \\
\top[x{\mapsto}t] &= \top \\
(\mathsf{A} \vee \mathsf{B})[x{\mapsto}t] &= \mathsf{A}[x{\mapsto}t] \ \vee \ \mathsf{B}[x{\mapsto}t] \\
\bot[x{\mapsto}t] &= \bot \\
(\mathsf{A} \supset \mathsf{B})[x{\mapsto}t] &= \mathsf{A}[x{\mapsto}t] \ \supset \ \mathsf{B}[x{\mapsto}t]
\end{aligned} \tag{4.1}$$

Likewise for the quantifiers:

$$\begin{aligned}
(\forall x : \mathsf{X} \,.\, \mathsf{A})[y{\mapsto}t] &= \forall x : \mathsf{X} \,.\, (\mathsf{A}[y{\mapsto}t]) \\
(\exists x : \mathsf{X} \,.\, \mathsf{A})[y{\mapsto}t] &= \exists x : \mathsf{X} \,.\, (\mathsf{A}[y{\mapsto}t])
\end{aligned} \tag{4.2}$$

provided that $x \neq y$ and $x \notin \mathsf{FV}(t)$, which can always be arranged by α-conversion.

By virtue of the reindexing functors being bicartesian closed in the case of the propositional connectives, and satisfying the Beck-Chevalley condition in the case of the quantifiers, we will indeed get commutativity of reindexing with the interpretations of the connectives – but only up to a canonical isomorphism, which is not necessarily equality.

For example, for a term $\Phi \mid t : \mathsf{X}$ and proposition $\Phi, x : \mathsf{X} \mid \mathsf{A} \wedge \mathsf{B}$ PROP, the object $[\![[x \mapsto t]]\!]^*([\![\mathsf{A} \wedge \mathsf{B}]\!])$ will indeed be *a* cartesian product of $[\![[x \mapsto t]]\!]^*([\![\mathsf{A}]\!])$ and $[\![[x \mapsto t]]\!]^*([\![\mathsf{B}]\!])$ in $\mathsf{P}([\![\Phi]\!])$. But there is no *a priori* reason to assume that it will be *the same* cartesian product as $[\![\mathsf{A}[x \mapsto t] \wedge \mathsf{B}[x \mapsto t]]\!]$. In order for this to obtain, a choice of *which* cartesian product in each fiber interprets conjunction must be made, and the reindexing functors between the fibers must respect this choice.

This situation exemplifies the general distinction between giving categorical definition in terms of *universal properties* on the one hand, and in terms of *specified structure* on the other. While universal properties are much-preferred by categorists for respecting the *principle of equivalence* [nla], they come at the cost of requiring coherence isomorphisms (possibly up to even higher-dimensional equivalences) to mediate in any equations that we would like to impose on objects.

For our purposes, the shortest path to a categorical interpretation of intuitionistic first-order logic involves imposing specified structure. Thus we will require that the interpretation of substitution be strict with respect to that of the connectives, in other words, that it satisfy (4.1) and (4.2) on the nose. For the propositional connectives, we achieve this by insisting that the bicartesian closed functors between fibers respect the specified bicartesian closed structure within them. Thus from now on, "BCC" will refer to the 2-category of bicartesian closed categories *with specified structure*, and functors between them that respect this structure. For the quantifiers, we insist that the Beck-Chevalley condition hold strictly; that is, that the diagram in the fibers in the third condition of definition 4.5.1 commute on the nose. Having made this choice, we may finally define the interpretation of typed first-order language in a hyperdoctrine.

Definition 4.6.2 (interpretation of first-order logic) Let $\mathcal{L} \coloneqq (\mathcal{T}, \mathcal{F}, \mathcal{R})$ be the signature of a typed first-order language, $[\![-]\!]_{\mathcal{T}}$ be an interpretation of typing contexts over $\mathcal{T}$ and $[\![-]\!]_{\mathcal{F}}$ be an interpretation of terms over $\mathcal{F}$ in a cartesian category $\mathbb{C}$, and let $[\![-]\!]_{\mathcal{R}}$ be an interpretation of predicates in a hyperdoctrine $\mathsf{P} : \mathbb{C}^{\circ} \longrightarrow$ BCC. Then $[\![-]\!]_{\mathcal{R}}$ extends inductively to an interpretation of propositions in context by the interpretations of the connectives.

For any propositions $\Phi \mid \mathsf{A}, \mathsf{B}\ \textsc{Prop}$ and $\Phi, x : \mathsf{X} \mid \mathsf{C}\ \textsc{Prop}$,

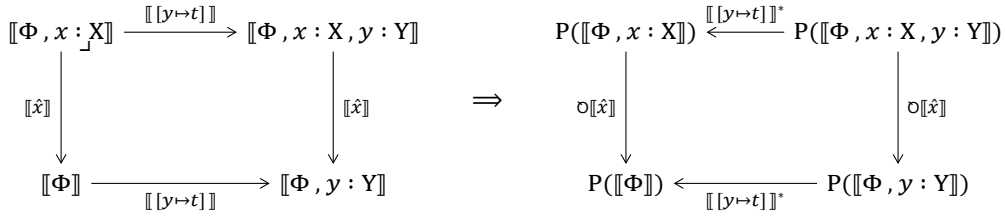
$$\begin{aligned}
[\![\Phi \mid \mathsf{A} \wedge \mathsf{B}\ \textsc{Prop}]\!] &\coloneqq [\![\mathsf{A}]\!] \times [\![\mathsf{B}]\!] : \mathsf{P}([\![\Phi]\!]) \\
[\![\Phi \mid \top\ \textsc{Prop}]\!] &\coloneqq 1 : \mathsf{P}([\![\Phi]\!]) \\
[\![\Phi \mid \mathsf{A} \vee \mathsf{B}\ \textsc{Prop}]\!] &\coloneqq [\![\mathsf{A}]\!] + [\![\mathsf{B}]\!] : \mathsf{P}([\![\Phi]\!]) \\
[\![\Phi \mid \bot\ \textsc{Prop}]\!] &\coloneqq 0 : \mathsf{P}([\![\Phi]\!]) \\
[\![\Phi \mid \mathsf{A} \supset \mathsf{B}\ \textsc{Prop}]\!] &\coloneqq [\![\mathsf{A}]\!] \supset [\![\mathsf{B}]\!] : \mathsf{P}([\![\Phi]\!]) \\
[\![\Phi \mid \forall x : \mathsf{X}.\, \mathsf{C}\ \textsc{prop}]\!] &\coloneqq \Pi[\![\hat{x}]\!]([\![\mathsf{C}]\!]) : \mathsf{P}([\![\Phi]\!]) \\
[\![\Phi \mid \exists x : \mathsf{X}.\, \mathsf{C}\ \textsc{prop}]\!] &\coloneqq \Sigma[\![\hat{x}]\!]([\![\mathsf{C}]\!]) : \mathsf{P}([\![\Phi]\!])
\end{aligned}$$

Lemma 4.6.3 (substitution and propositional connectives) For any interpretation of a first-order language in a hyperdoctrine, the interpretations of the propositional connectives commute with the interpretation of substitution.

Proof. Precisely because substitution is interpreted by a bicartesian closed functor that respects the interpretations of the propositional connectives. □

Lemma 4.6.4 (substitution and quantification) For any interpretation of a first-order language in a hyperdoctrine, the interpretations of the quantifiers commute with the interpretation of (capture-avoiding) substitution.

Proof. This property is ensured by the definition of *having quantifiers*. Let $\mathsf{P} : \mathbb{C}^{\circ} \longrightarrow \mathrm{BCC}$ be a hyperdoctrine and $[\![-]\!]$ be an interpretation of $\mathcal{L}$ in P. Given term $\Phi \mid t : \mathsf{Y}$ of $\mathcal{L}$, the diagram on the left is a pullback in $\mathbb{C}$ where the two vertical arrows are projections. Thus for $\mathfrak{O} \in \{\Sigma, \Pi\}$ the diagram on the right commutes as well. For any proposition $\Phi, x : \mathsf{X}, y : \mathsf{Y} \mid \mathsf{A}\ \textsc{prop}$ of $\mathcal{L}$, chasing $[\![\mathsf{A}]\!]$ around that diagram gives the desired result.



□

Definition 4.6.5 (free interpretation of an intuitionistic first-order language) If $\mathcal{L} \coloneqq (\mathcal{T}, \mathcal{F}, \mathcal{R})$ is a signature for a first-order language and $\textsc{Type}_{\mathcal{L}}$ is the free cartesian category generated by $\mathcal{F}$, then we define $\textsc{Prop}_{\mathcal{L}} : \textsc{Type}_{\mathcal{L}} \longrightarrow \mathrm{BCC}$ to be the free hyperdoctrine over $\textsc{Type}_{\mathcal{L}}$ generated by $\mathcal{R}$ and the *free interpretation* of $\mathcal{L}$ in $\textsc{Prop}_{\mathcal{L}}$ to be the one taking each relation symbol to itself, regarded as an object in the fiber of the interpretation of its context.

When the language under discussion is clear or irrelevant, we will drop the subscript $\mathcal{L}$. We should note that using the name "Prop" both for a free bicartesian closed category of a propositional language and for the functor of a free hyperdoctrine of first-order language should cause no confusion. When we write "Prop" in the former sense, we simply mean $\textsc{Prop}(\llbracket\Phi\rrbracket)$ for some unspecified typing context Φ.

If we were to define the hyperdoctrine interpretation for intuitionistic first-order logic by universal properties rather than specified structure, it would provide an interpretation naturally suited to logic with explicit substitution. There, we would have coherence isomorphisms such as $\llbracket(\mathsf{A} \wedge \mathsf{B})[x{\mapsto}t]\rrbracket \xrightarrow{\cong} \llbracket\mathsf{A}[x{\mapsto}t] \ \wedge \ \mathsf{B}[x{\mapsto}t]\rrbracket$ playing the role of the explicit substitutions. We could also consider these morphisms to be non-invertible, yielding a co-lax version of a hyperdoctrine with directed rewritings for applying substitutions.

4.7 Posetal Hyperdoctrines

We have explained how the concept of a hyperdoctrine is designed to interpret typed intuitionistic first-order logic. But of course the definition of a hyperdoctrine does not make any reference to a type theory or logic. We will see in chapter 5 that there is a remarkable relationship between normal natural deduction derivations and arrows in the fibers of a free hyperdoctrine. But first we attempt to provide some familiarity with the concept of hyperdoctine by considering two "naturally-occurring" instances, in which the fibers are posets.

Perhaps the most familiar hyperdoctrine is the **subset hyperdoctrine**, where the base category is Set, the category of (small) sets and the indexing functor is Sub, which takes each set X to the partial order of its subsets, $\wp(\mathsf{X})$, regarded as a category, and each function $f : \mathsf{X} \longrightarrow \mathsf{Y}$ to the induced **inverse image** map:

$$\begin{array}{rcl} & \overline{f^{-1}} & \\ \wp(\mathsf{Y}) & \longrightarrow & \wp(\mathsf{X}) \\ \mathsf{V} & \longmapsto & \{x \in \mathsf{X} \mid f(x) \in \mathsf{V}\} \end{array}$$

Each fiber is a bounded complete distributive lattice under union and intersection, and thus a **Heyting algebra**; in other words, a bicartesian closed poset. Implication is definable as:

$$\mathsf{A} \supset \mathsf{B} \coloneqq \bigvee \mathsf{C} \,.\, \mathsf{C} \wedge \mathsf{A} \leq \mathsf{B}$$

Inverse image is a continuous map and thus preserves the Heyting algebra structure.

Inverse image has a left adjoint, given by the **existential image** map:

$$\begin{array}{rcl} & \underline{\exists f} & \\ \wp(X) & \longrightarrow & \wp(Y) \\ U & \longmapsto & \{y \in Y \mid \exists x \in f^{-1}(\{y\}) \,.\, x \in U\} \end{array}$$

which may be written in terms of f rather than f^{-1} using *guarded quantification* as,

$$\{y \in Y \mid \exists x \in X \,.\, f(x) = y \wedge x \in U\}$$

It also has a right adjoint, given by the **universal image** map:

$$\begin{array}{rcl} & \underline{\forall f} & \\ \wp(X) & \longrightarrow & \wp(Y) \\ U & \longmapsto & \{y \in Y \mid \forall x \in f^{-1}(\{y\}) \,.\, x \in U\} \end{array}$$

which may also be written using guarded quantification as,

$$\{y \in Y \mid \forall x \in X \,.\, f(x) = y \supset x \in U\}$$

When f is a projection function, $\pi : Y \times Z \longrightarrow Y$, its inverse image defines a cylinder:

$$\pi^{-1}(V) = V \times Z$$

and the quantifiers correspond to the projection of the smallest cylinder containing, respectively the largest cylinder contained in, the given set:

$$\exists \pi(U) = \{y \in Y \mid \exists z \in Z \,.\, (y\,, z) \in U\} \qquad \forall\, \pi(U) = \{y \in Y \mid \forall z \in Z \,.\, (y\,, z) \in U\}$$

which we abbreviate as $\exists z \in Z \,.\, U$ and $\forall z \in Z \,.\, U$, respectively. This has a particularly intuitive graphical representation, shown in figure 4.2.

In this case the adjunctions governing the quantifiers stipulate,

$$\frac{\;U \subseteq V \times Z\;}{\exists z \in Z \,.\, U \subseteq V} \qquad \text{and} \qquad \frac{\;V \subseteq \forall z \in Z \,.\, U\;}{V \times Z \subseteq U}$$

and the unit and counit of the two adjunctions tell us that,

$$\begin{array}{l} \eta_{\exists} : U \subseteq (\exists z \in Z \,.\, U) \times Z \\ \varepsilon_{\exists} : \exists z \in Z \,.\, (V \times Z) \subseteq V \end{array} \qquad \text{and} \qquad \begin{array}{l} \eta_{\forall} : V \subseteq \forall z \in Z \,.\, (V \times Z) \\ \varepsilon_{\forall} : (\forall z \in Z \,.\, U) \times Z \subseteq U \end{array}$$

all of which we can easily verify from the figure by inspection. Incidentally, the counit of the existential adjunction and the unit of the universal one are in fact equalities. It is a

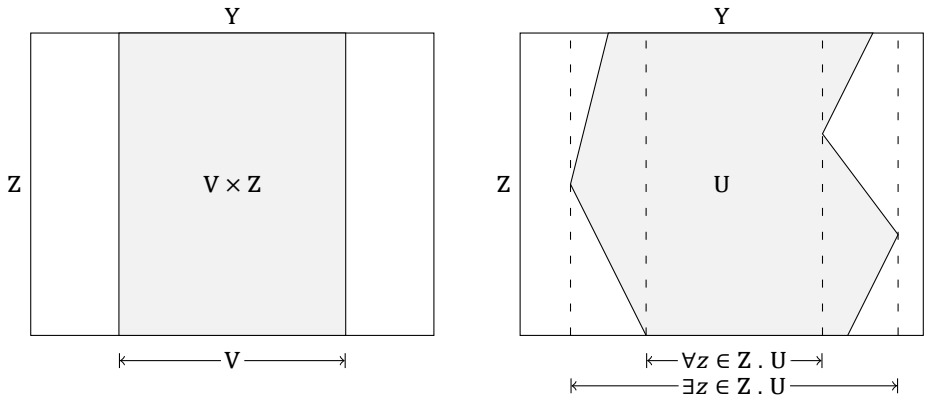
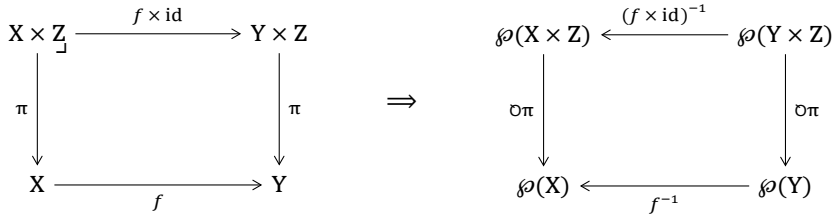


Figure 4.2: Quantification over projections in Set

theorem that the unit, respectively counit, of an adjunction is an isomorphism just in case the left, respectively right, adjoint functor is full and faithful. Thus we may rest assured that $V \subseteq V'$ in $\mathrm{Sub}(Y)$ iff $V \times Z \subseteq V' \times Z$ in $\mathrm{Sub}(Y \times Z)$.

Having quantifiers for projections requires for any pullback of a projection, necessarily of the form on the left, the corresponding diagram for each quantifier, shown on the right, commutes as well – and in any case strictly, because equality is the only form of isomorphism available in a poset.



There, for any $U \subseteq Y \times Z$ and $x \in X$, we have:

$$
\begin{aligned}
& x \in \forall\pi((f \times \mathrm{id}_Z)^{-1}(U)) \\
\Leftrightarrow\ & \text{[definition } \forall\pi\text{]} \\
& \forall z \in Z\,.\,(x\,,z) \in (f \times \mathrm{id}_Z)^{-1}(U) \\
\Leftrightarrow\ & \text{[definition } (-)^{-1}\text{]} \\
& \forall z \in Z\,.\,(f(x)\,,z) \in U \\
\Leftrightarrow\ & \text{[definition } \forall\pi\text{]} \\
& f(x) \in \forall\pi(U) \\
\Leftrightarrow\ & \text{[definition } (-)^{-1}\text{]} \\
& x \in f^{-1}(\forall\pi(U))
\end{aligned}
$$

We can interpret a language of typed first-order logic in this hyperdoctrine by assigning each atomic type to a set, each function symbol to a function and each relation symbol to a subset of the interpretation of its type. This is the approach taken in (set-based) model theory.

Another important example is that of a **sieve hyperdoctrine**. Given any category $\mathbb{C}$, a **presheaf** on $\mathbb{C}$ is a functor $F : \mathbb{C}^\circ \longrightarrow \textsc{Set}$. The **category of presheaves** on $\mathbb{C}$, $\mathbb{C}^\circ \supset \textsc{Set}$, is the functor category having presheaves on $\mathbb{C}$ for objects and natural transformations between them for arrows. The covariant **Yoneda embedding**, $Y : \mathbb{C} \longrightarrow \mathbb{C}^\circ \supset \textsc{Set}$ is the currying of the hom bifunctor (swapping argument order), which we know to be full and faithful by the *Yoneda lemma*. A presheaf in the image of the Yoneda embedding is called **representable**.

A **subfunctor** is a just subobject in a functor category. In more detail, a natural transformation is monic just in case all of its components are. Given parallel functors F and G, we call F (a representative of) a subfunctor of G if there is a monic natural transformation $\varphi : F \longrightarrow G$. Just as in the case of ordinary subobjects, this induces a partial order on coterminal monic natural transformations, and by extension, on their domains: given another monic natural transformation $\psi : E \longrightarrow G$, if ψ factors through φ as $\psi = \theta \cdot \varphi$, then θ is unique because φ is monic and it is monic because ψ is monic. Then we say $E \leq F \leq G$ (since id_G is also monic).

An equivalent (and intuitive) characterization of a subfunctor of a representable functor in a presheaf category is that of a sieve. A **sieve** on an object $X : \mathbb{C}$ is a set U of arrows with codomain X that is closed under precomposition by $\mathbb{C}$-arrows:

$$f : \mathbb{C}(Y \to X) \in U \quad \wedge \quad g : \mathbb{C}(Z \to Y) \qquad \Longrightarrow \qquad g \cdot f \in U$$

This closure under precomposition is sometimes called "saturation". Of course any set of coterminal arrows may be saturated to a sieve on their common codomain. We will write "$\langle - \rangle$" for the sieve generated from a set by saturation. A sieve on X turns out to be the same thing as a subfunctor of $\mathbb{C}(- \to X)$.

In a sieve hyperdoctrine, the base category is the category of presheaves on some underlying category, which being a topos, has finite products. The indexing functor takes each presheaf to the partial order of its subfunctors:

$$\textsc{Sub} : \mathbb{C}^\circ \supset \textsc{Set} \longrightarrow \textsc{Poset}$$

Because a presheaf category is the free cocompletion of its underlying category, it suffices to consider just the representables. Therefore we may equivalently say that the indexing functor takes each object of the underlying category to its poset of sieves, where the partial order is just inclusion ($U \leq U' \coloneqq U \subseteq U'$). From this perspective, given an arrow $f : \mathbb{C}(X \to$

Y), the reindexing functor it generates is:

$$\begin{array}{rcl} & \overline{f^*} & \\ \textsc{Sub}(\mathbb{C}\,(- \to \mathsf{Y})) & \longrightarrow & \textsc{Sub}(\mathbb{C}\,(- \to \mathsf{X})) \\ \mathsf{V} & \longmapsto & \{x :: \mathbb{C} \mid x \cdot f \in \mathsf{V}\} \end{array}$$

As in the case of subsets, the fibers are Heyting algebras whose structure is preserved by reindexing. Reindexing has a left adjoint:

$$\begin{array}{rcl} & \overline{\Sigma f} & \\ \textsc{Sub}(\mathbb{C}\,(- \to \mathsf{X})) & \longrightarrow & \textsc{Sub}(\mathbb{C}\,(- \to \mathsf{Y})) \\ \mathsf{U} & \longmapsto & \{y :: \mathbb{C} \mid \exists x \in f^*(\langle\{y\}\rangle)\,.\, x \in \mathsf{U}\} \end{array}$$

Said more simply, this takes a sieve U on X to the sieve on Y containing the composition of each arrow in U with f. Reindexing also has a right adjoint:

$$\begin{array}{rcl} & \overline{\Pi f} & \\ \textsc{Sub}(\mathbb{C}\,(- \to \mathsf{X})) & \longrightarrow & \textsc{Sub}(\mathbb{C}\,(- \to \mathsf{Y})) \\ \mathsf{U} & \longmapsto & \{y :: \mathbb{C} \mid \forall x \in f^*(\langle\{y\}\rangle)\,.\, x \in \mathsf{U}\} \end{array}$$

This takes a sieve U on X to the sieve on Y containing those arrows, all of whose factorizations through f are by arrows in U.

We can interpret a language of typed first-order logic in this hyperdoctrine by assigning each atomic type to an object of $\mathbb{C}$, each function symbol to an arrow of $\mathbb{C}$ and each relation symbol to a sieve on the interpretation of its type. In this way each proposition can be regarded as its set of satisfying substitutions. This provides a useful generalization of a **Herbrand interpretation**, in which predicates are represented by the set of tuples of closed terms that satisfy them. In contrast, in a sieve hyperdoctrine closed terms play no special role, providing for a **non-ground semantics**. This is the categorical semantics used by Lipton et al. [FFL03] and Krishnan [Kri05] in their categorical investigations of logic programming.

These two hyperdoctrines are examples of a more general phenomenon. Given any topos $\mathbb{C}$, it is a theorem that the indexed category of subobjects, $\textsc{Sub} : \mathbb{C}^\circ \longrightarrow \textsc{Poset}$ forms a **Heyting category**, that is, an indexed bicartesian closed category with posetal fibers. Furthermore, the Beck-Chevalley condition holds for the $\textsc{Sub}$-image of any pullback square in the base. Since a topos has all limits, such a hyperdoctrine has quantifiers for every $\mathbb{C}$-arrow. The details may be found in [MM92].

Chapter 5

Natural Deduction by Adjunction

The use of adjoint functors to provide universal constructions for interpreting logical connectives goes back to at least the work of Lawvere in the late 1960s. Although Lawvere clearly intended the structure of a hyperdoctrine to interpret intuitionistic first-order logic, there is no mention of any particular system of formal derivation in [Law69] or [Law70]. These present examples involving the consequence relation, resulting in hyperdoctrines with preorder fibers, like our examples in chapter 4. In the latter, Lawvere writes, "honest proof theory would presumably also yield a hyperdoctrine with nontrivial $\mathsf{P}(\mathsf{X})$, but a syntactically-presented one".

The connection to "honest proof theory" was first made in the propositional case in a series of articles by Lambek [Lam68; Lam69; Lam72] culminating in his book with Scott [LS86], and in the first-order case by Seely in [See83]. In [HM92] Harnik and Makkai write,

> Prawitz advanced the thesis that two deductions in natural deduction represent the same proof iff they are inter-reducible in a suitable lambda-calculus. We believe that, after a suitable and natural link is established between natural deduction and the Lambek calculus, Prawitz's inter-reducibility will turn out to be equivalent to equality deducible under the [axioms of cartesian closed categories with coproducts (interpreting minimal propositional logic)].

We will show, in fact, that convertibility of natural deduction derivations in intuitionistic first-order logic is equivalent to equality of arrows in the fibers of a free hyperdoctrine.

In this chapter we explain how the version of hyperdoctrine that we have constructed captures the concept of normal derivation in Gentzen's system of natural deduction. We show that the adjunctions characterizing the interpretations of the connectives determine inference rules and conversion relations for a system of derivation in a completely uniform way, and that these rules and conversions are equivalent to those of natural deduction. Thus

we show how the derivation system of natural deduction could have been "discovered" from the categorical semantics of intuitionistic first-order logic. This can be seen as a contribution to Melliès's program (mentioned in the introduction) of providing algebraic semantics for proof theory.

By viewing natural deduction as a categorical graphical language, we see that *hypothetical derivations* play the role of adjoint complements. We see also that the connectives are naturally partitioned into two sets depending upon their **chirality**; that is, on whether they are characterized by a right or a left adjoint functor. We believe that this chirality-based perspective provides an important complement to the introduction rule-based perspective (*verificationism*) and elimination rule-based perspective (*pragmatism*) found in proof theory. We will call the connectives that are characterized by right adjoint functors (those in the set $\{\top, \wedge, \supset, \forall\}$) **right connectives**, and likewise, the connectives that are characterized by left adjoint functors (those in the set $\{\bot, \vee, \exists\}$) **left connectives**.

Our main result in this chapter is that by viewing derivations as a categorical graphical language, inference rules of *introduction* and *elimination*, as well as derivation conversions of *local reduction*, *local expansion* and *permutation*, can be derived uniformly for the connectives from their characterizations by adjunctions; and that the rules and conversions obtained in this way are equivalent to those of natural deduction. The demonstration of this result will occupy the next few sections.

Theorem 5.0.1 (natural deduction by adjunction) The adjunction-theoretic interpretation of the connectives of intuitionistic first-order logic extends to an interpretation of the inference rules and derivation conversions of a derivation system of natural deduction in the following uniform way. For each class of connectives the adjoint-theoretic concept on the right provides as interpretation for the proof-theoretic concept on the left.

- For right connectives,

  introduction rules: the adjoint complement operation $(-^\flat)$,

  elimination rules: the component of the counit (ε),

  local reductions: the factorization of arrows from a left adjoint image in the *universal property of the counit* $(\mathsf{F}(\mathcal{D}^\flat) \cdot \varepsilon = \mathcal{D})$,

  permutation conversions: (which are implicit in Gentzen's syntax) the naturality of the hom set isomorphism in the domain coordinate $(\mathcal{E} \cdot \mathcal{D}^\flat = (\mathsf{F}(\mathcal{E}) \cdot \mathcal{D})^\flat)$,

  local expansions: the fact that identity maps on right adjoint images are adjoint complements of counit components $(\mathsf{id}_{\mathsf{G}} = \varepsilon^\flat)$.

- For left connectives,

introduction rules: the component of the unit (η),

elimination rules: the adjoint complement operation $(-^\sharp)$,

local reductions: the factorization of arrows to a right adjoint image in the *universal property of the unit* $(\eta \cdot \mathsf{G}(\mathcal{D}^\sharp) = \mathcal{D})$,

permutation conversions: the naturality of the hom set isomorphism in the codomain coordinate $(\mathcal{D}^\sharp \cdot \mathcal{E} = (\mathcal{D} \cdot \mathsf{G}(\mathcal{E}))^\sharp)$,

local expansions: the fact that identity maps on left adjoint images are adjoint complements of unit components $(\mathsf{id}_\mathsf{F} = \eta^\sharp)$.

In the presence of the *distributive law* (lemma 3.3.2) and *Frobenius reciprocity* (lemma 3.3.3), both of which hold in a hyperdoctrine, the inference rules obtained in this way are interchangeable for those in figure 2.1, in the sense that they have the the same premises, conclusion and hypothetical subderivations – except in the case of the non-invertible quantifier rules ($\forall-$ and $\exists+$), where the derived rules decompose their traditional versions into a strictly logical rule and a substitution. This fact will prove very useful in our considerations of proof search in the sequel.

The equations interpreting the relations on derivations are described in definition 3.1.1, lemma 3.1.3 and lemma 3.1.2, respectively. The derived permutation conversions for right connectives do not appear in Gentzen's syntax because there the respective operations on precomposed derivations are implicit. Making these operations explicit sheds light on the algebraic principles governing the meta-theory of natural deduction. It also allows us to see that the traditional local expansions for right connectives may be decomposed into a permutation and a "hyper-local" expansion.

We wish to interpret natural deduction derivations as arrows in categories whose objects interpret propositional contexts. In order to do this we must make explicit two things that are left tacit in presentations of natural deduction like that in figure 2.1. The first is the *source* of a derivation, that is, its collection of logical assumptions. In a category, every arrow has a well-defined domain as well as a codomain, yet in the inference rules for natural deduction the sources of subderivations are left implicit. If a derivation is a *proof*, then its source is the empty context. However, we need to be able to compose derivations, so we must be able to accommodate arbitrary sources.

The second thing we must make explicit is the category in which the interpretation of a derivation resides. We must keep track of the category we are working in because some inference rules have subderivations interpreted in different categories. Specifically, rules corresponding to adjoint complement operations (i.e. introduction rules of right connectives and elimination rules of left connectives) generally have their minor subderivation (which we will always write on the right) in a category different from that of their major premise and

conclusion, which are always in the same category. In these rules, the primitive inference from major premise to conclusion is the *adjoint complement* of the minor subderivation, schematically:

$$\dfrac{\mathtt{A} \qquad \dfrac{\overline{\mathtt{F(A)}}}{\dfrac{\mathcal{D}}{\mathtt{B}}}}{\mathtt{G(B)}}\ *+ \qquad \text{and} \qquad \dfrac{\mathtt{F(A)} \qquad \dfrac{\dfrac{\overline{\mathtt{A}}}{\mathcal{D}}}{\mathtt{G(B)}}}{\mathtt{B}}\ *-$$

for adjunction $\mathtt{F} \dashv \mathtt{G}$. Intuitively, such rules assert that the minor subderivation may be traded for a primitive inference from the major premise to the conclusion. The composition of inferences in a derivation is interpreted by the composition of the arrows interpreting those inferences in a category. The leftmost branch, or **trunk** of a derivation will be interpreted as a composition of arrows from the interpretation of the *assumptions* to that of the *end-formula*.

When thinking about the fibers of a hyperdoctrine, it is safe to intuitively regard objects as interpretations of individual propositions. This is because the logic being interpreted includes the connectives truth and conjunction, which, like the empty context and context extension operation, are interpreted by finite products. A propositional context is equivalent to a proposition comprising the conjunction of its contents, but strictly speaking, in natural deduction a context is a meta-level thing while an equivalent conjunction is an object-level thing. We say that a conjunction **internalizes** a propositional context by representing it as a proposition.

In the following discussion we will assume that our interpretations are *free*, that is, in the hyperdoctrine Prop; and we will mercifully stop writing the interpretation brackets. But the result holds for any hyperdoctrine interpretation.

In order to view natural deduction derivations as arrows, we need a natural deduction notation for derivations in the categories we will find ourselves in. Recall that an arrow in a product of categories is an ordered pair consisting of an arrow from each. Therefore, we will represent a derivation in the category Prop × Prop as a pair of derivations in Prop placed side by side:

$$\dfrac{\dfrac{(\Gamma_1\,,\Gamma_2)}{(\mathcal{D}_1\,,\mathcal{D}_2)}}{(\mathtt{A}_1\,,\mathtt{A}_2)} \quad = \quad \dfrac{\dfrac{\Gamma_1}{\mathcal{D}_1}}{\mathtt{A}_1} \quad \dfrac{\dfrac{\Gamma_2}{\mathcal{D}_2}}{\mathtt{A}_2}$$

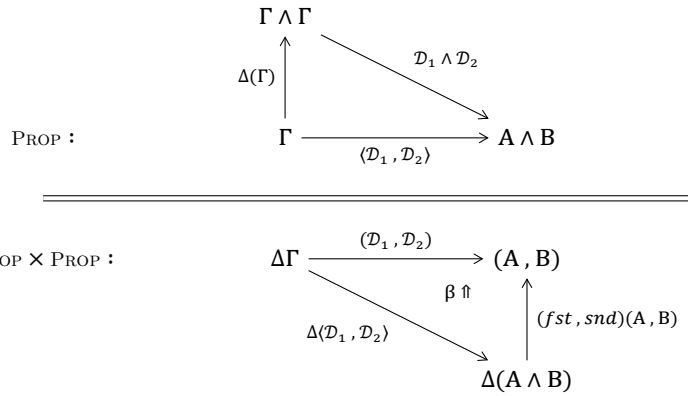
As for derivations in the terminal category $\mathbb{1}$, there is only one, namely the *identity derivation* on $*$:

$$\dfrac{\dfrac{*}{\mathrm{id}_*}}{*} \quad = \quad *$$

We now establish the claims of the theorem for each connective in turn. In each case, we will distinguish the adjoint-theoretic "moral rule" described in the theorem with the annotation "*" whenever it differs from the corresponding rule in figure 2.1.

5.1 Conjunction

Recall that we interpret conjunction as the cartesian product and that this bifunctor is right adjoint to a *diagonal functor* ($\Delta \dashv \wedge$). We can summarize this adjunction with the following instance of diagram (3.4):



(5.1)

Here $\langle -\,,-\rangle$ is the tupling operation, fst and snd are the projections and $\Delta \coloneqq \langle \mathsf{id}\,,\mathsf{id}\rangle$ is the internal diagonal. The 2-cell β is just equality, but it is instructive to give it a name and orientation, none the less. We will address the meaning of a conjunction of derivations $(\mathcal{D}_1 \wedge \mathcal{D}_2)$ momentarily.

In this adjunction, the adjoint complement operation $-^\flat$ takes a pair of derivations with common assumptions to a single derivation of the conjunction of their respective goals:

$$(\mathcal{D}_1\,,\mathcal{D}_2) \;:\; \Delta\Gamma \longrightarrow (\mathsf{A}\,,\mathsf{B}) \qquad \overset{-^\flat}{\longmapsto} \qquad \langle \mathcal{D}_1\,,\mathcal{D}_2\rangle \;:\; \Gamma \longrightarrow \mathsf{A}\wedge\mathsf{B}$$

This allows us to trade $(\mathcal{D}_1\,,\mathcal{D}_2)$ for $\langle \mathcal{D}_1\,,\mathcal{D}_2\rangle$. We can express this trade with the following inference rule of natural deduction:

$$\frac{\dfrac{\Gamma}{\mathcal{D}_1}\,\mathsf{A} \qquad \dfrac{\Gamma}{\mathcal{D}_2}\,\mathsf{B}}{\mathsf{A}\wedge\mathsf{B}}\wedge\!+^\star \qquad \overset{-^\flat}{\longmapsto} \qquad \frac{\Gamma \qquad \dfrac{\overline{\Gamma}}{\mathcal{D}_1}\,\mathsf{A} \qquad \dfrac{\overline{\Gamma}}{\mathcal{D}_2}\,\mathsf{B}}{\mathsf{A}\wedge\mathsf{B}}$$

The assumptions of the surrendered derivations are discharged, and in exchange we receive a new one-step derivation of a conjunction. We would like to call this the introduction rule for conjunction. It is not exactly the rule given by Gentzen, but it is interchangeable for it

once the implicit sources of the subderivations are made explicit:

$$\dfrac{\dfrac{\Gamma}{\dfrac{\mathcal{D}_1}{\mathrm{A}}} \quad \dfrac{\Gamma}{\dfrac{\mathcal{D}_2}{\mathrm{B}}}}{\mathrm{A} \wedge \mathrm{B}}\ \wedge{+}$$

The counit of this adjunction is the ordered pair of projections (fst, snd). When written as inference rules, they form exactly the ordered pair of elimination rules for conjunction:

$$\mathrm{A} \quad \mathrm{B} \quad \overset{\varepsilon}{\longmapsto} \quad \dfrac{\mathrm{A} \wedge \mathrm{B}}{\mathrm{A}}\ \wedge{-}_1 \quad \dfrac{\mathrm{A} \wedge \mathrm{B}}{\mathrm{B}}\ \wedge{-}_2$$

We can now see how the factorization in the universal property of the counit,

$$\Delta\langle \mathcal{D}_1 \,,\, \mathcal{D}_2 \rangle \cdot (fst \,,\, snd) \quad \overset{\beta}{=} \quad (\mathcal{D}_1 \,,\, \mathcal{D}_2)$$

expresses the local reduction for conjunction in the category Prop × Prop:

$$\dfrac{\Gamma \quad \dfrac{\dfrac{\overline{\Gamma}}{\mathcal{D}_1}}{\mathrm{A}} \quad \dfrac{\dfrac{\overline{\Gamma}}{\mathcal{D}_2}}{\mathrm{B}}}{\dfrac{\mathrm{A} \wedge \mathrm{B}}{\mathrm{A}}\ \wedge{-}_1}\ \wedge{+}^{\star} \qquad \dfrac{\Gamma \quad \dfrac{\dfrac{\overline{\Gamma}}{\mathcal{D}_1}}{\mathrm{A}} \quad \dfrac{\dfrac{\overline{\Gamma}}{\mathcal{D}_2}}{\mathrm{B}}}{\dfrac{\mathrm{A} \wedge \mathrm{B}}{\mathrm{B}}\ \wedge{-}_2}\ \wedge{+}^{\star} \qquad \overset{\wedge{>}^{\star}}{\longmapsto} \qquad \dfrac{\dfrac{\Gamma}{\mathcal{D}_1}}{\mathrm{A}} \quad \dfrac{\dfrac{\Gamma}{\mathcal{D}_2}}{\mathrm{B}}$$

This explains why we call the commuting triangle of the universal property of the counit "β": short-cutting it performs β-reduction.

The naturality of the hom set bijection of this adjunction in the domain coordinate,

$$\mathcal{E} \cdot \langle \mathcal{D}_1 \,,\, \mathcal{D}_2 \rangle \quad = \quad \langle \mathcal{E} \cdot \mathcal{D}_1 \,,\, \mathcal{E} \cdot \mathcal{D}_2 \rangle$$

says that any derivation precomposed to a $\wedge{+}^{\star}$ rule may be moved into the minor branch by duplication:

$$\dfrac{\dfrac{\Gamma}{\dfrac{\mathcal{E}}{\mathrm{C}}} \quad \dfrac{\dfrac{\overline{\mathrm{C}}}{\mathcal{D}_1}}{\mathrm{A}} \quad \dfrac{\dfrac{\overline{\mathrm{C}}}{\mathcal{D}_2}}{\mathrm{B}}}{\mathrm{A} \wedge \mathrm{B}}\ \wedge{+}^{\star} \qquad \overset{\wedge{\rightleftarrows}^{\star}}{\longmapsto} \qquad \dfrac{\Gamma \quad \dfrac{\dfrac{\dfrac{\overline{\Gamma}}{\mathcal{E}}}{\mathrm{C}}}{\mathcal{D}_1}}{\mathrm{A}} \quad \dfrac{\dfrac{\dfrac{\overline{\Gamma}}{\mathcal{E}}}{\mathrm{C}}}{\mathcal{D}_2}}{\mathrm{B}}}{\mathrm{A} \wedge \mathrm{B}}\ \wedge{+}^{\star}$$

Here C may internalize a context. This transformation is necessary with our formulation of the $\wedge{+}^{\star}$ rule in order to unite a β-redex that may be split between the the major and minor branches of the rule. The situation is dual to that for disjunction permutation with the $\vee{-}$ rule, as described in (2.1).

The equation for the adjoint complement of a counit component,

$$\mathrm{id}_{-\wedge-} \quad = \quad \langle fst\,, snd \rangle$$

expresses the local expansion for conjunction:

$$\mathrm{A} \wedge \mathrm{B} \quad \overset{\wedge <^{\star}}{\longmapsto} \quad \dfrac{\mathrm{A} \wedge \mathrm{B} \qquad \dfrac{\dfrac{\overline{\mathrm{A} \wedge \mathrm{B}}}{\mathrm{A}}\,\wedge -_1 \qquad \dfrac{\overline{\mathrm{A} \wedge \mathrm{B}}}{\mathrm{B}}\,\wedge -_2}{\mathrm{A} \wedge \mathrm{B}}\,\wedge +^{\star}$$

We recover the formulation of table 2.3 by precomposing an arbitrary derivation and applying the permutation conversion.

Finally, we explain the arrow $\mathcal{D}_1 \wedge \mathcal{D}_2$ that appears in diagram (5.1). In natural deduction, connectives such as conjunction are defined only on propositions and not on derivations as well. In contrast, the cartesian product is a functor, acting on the arrows, as well as on the objects of a category. The categorical description of conjunction allows us to extend its definition functorially to derivations by specifying what the conjunction of derivations would have to be if it were defined. In particular, the characterization of the adjoint functor image of arrows (lemma 3.1.4) tells us that for any derivations $\mathcal{D}_1$ and $\mathcal{D}_2$,

$$\mathcal{D}_1 \wedge \mathcal{D}_2 \quad = \quad \langle fst \cdot \mathcal{D}_1\,, snd \cdot \mathcal{D}_2 \rangle$$

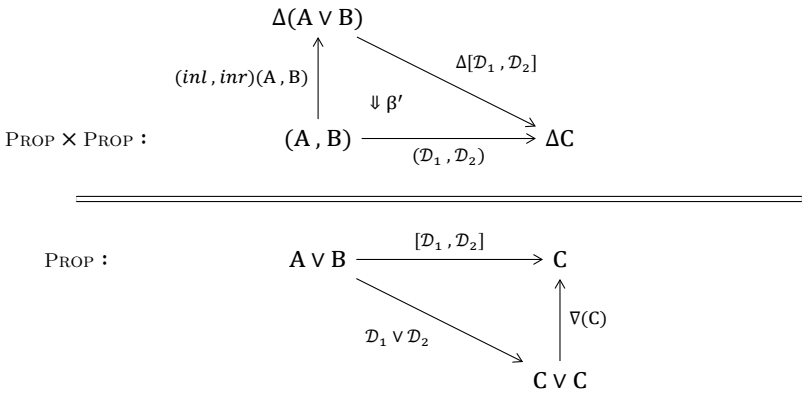
In natural deduction notation this yields the definition:

$$\dfrac{\Gamma_1 \wedge \Gamma_2}{\dfrac{\mathcal{D}_1 \wedge \mathcal{D}_2}{\mathrm{A} \wedge \mathrm{B}}} \quad \overset{\wedge \rightarrow}{\coloneqq} \quad \dfrac{\Gamma_1 \wedge \Gamma_2 \qquad \dfrac{\dfrac{\overline{\Gamma_1 \wedge \Gamma_2}}{\Gamma_1}\,\wedge -_1}{\dfrac{\mathcal{D}_1}{\mathrm{A}}} \qquad \dfrac{\dfrac{\overline{\Gamma_1 \wedge \Gamma_2}}{\Gamma_2}\,\wedge -_2}{\dfrac{\mathcal{D}_2}{\mathrm{B}}}}{\mathrm{A} \wedge \mathrm{B}}\,\wedge +^{\star}$$

5.2 Disjunction

Recall that we interpret disjunction as the coproduct and that this bifunctor is left adjoint to a *diagonal functor* ($\vee \dashv \Delta$). We can summarize this adjunction with the following instance

of diagram (3.4):



(5.2)

Here $[-\,,-]$ is the cotupling operation, *inl* and *inr* are the coprojections, $\nabla \coloneqq [\mathsf{id}\,,\mathsf{id}]$ is the internal codiagonal and β' is a notable equality. As in the case for conjunction, we may conservatively extend the definition of disjunction to derivations.

In this adjunction, the adjoint complement operation $-^\sharp$ takes a pair of derivations with common goal to a single derivation of that goal from the disjunction of the respective assumptions:

$$(\mathcal{D}_1\,,\mathcal{D}_2)\ :\ (\mathsf{A}\,,\mathsf{B}) \longrightarrow \Delta\mathsf{C} \qquad \overset{-^\sharp}{\longmapsto} \qquad [\mathcal{D}_1\,,\mathcal{D}_2]\ :\ \mathsf{A}\vee\mathsf{B} \longrightarrow \mathsf{C}$$

It allows us to trade $(\mathcal{D}_1\,,\mathcal{D}_2)$ for $[\mathcal{D}_1\,,\mathcal{D}_2]$. We can express this trade with the following inference rule of natural deduction:

$$\dfrac{\dfrac{\mathsf{A}}{\mathcal{D}_1}\mathsf{C} \qquad \dfrac{\mathsf{B}}{\mathcal{D}_2}\mathsf{C}}{\mathsf{C}} \quad \overset{-^\sharp}{\longmapsto} \quad \dfrac{\mathsf{A}\vee\mathsf{B} \qquad \dfrac{\overline{\mathsf{A}}}{\mathcal{D}_1}\mathsf{C} \qquad \dfrac{\overline{\mathsf{B}}}{\mathcal{D}_2}\mathsf{C}}{\mathsf{C}}\ \vee\!-^\star$$

This "moral" elimination rule for disjunction looks superficially like Gentzen's rule, but there is a subtle complication involving contexts. Recall that in Gentzen's rule there is an implicit ambient propositional context. If we write it explicitly, then the rule looks like this:

$$\dfrac{\dfrac{\Gamma}{\mathcal{E}}\mathsf{A}\vee\mathsf{B} \qquad \dfrac{\Gamma\,,\overline{\mathsf{A}}}{\mathcal{D}_1}\mathsf{C} \qquad \dfrac{\Gamma\,,\overline{\mathsf{B}}}{\mathcal{D}_2}\mathsf{C}}{\mathsf{C}}\ \vee\!-$$

Because it occurs in the trunk of the derivation, Γ must be a valid context in the category Prop. So we must explain how it gets into the minor subderivation in Prop × Prop. We may suspect that it is transported there by the right adjoint functor Δ, but we must ensure that this belief is justified by the properties of our categorical interpretation.

Using a conjunction to internalize the context, we have the natural bijection:

$$(\mathcal{D}_1\,,\mathcal{D}_2)\ :\ ((\Gamma\,,A)\,,(\Gamma\,,B))\longrightarrow \Delta C \qquad \overset{-^\sharp}{\longmapsto} \qquad [\mathcal{D}_1\,,\mathcal{D}_2]\ :\ (\Gamma\wedge A)\vee(\Gamma\wedge B)\longrightarrow C$$

We could recover Gentzen's elimination rule for disjunction in the presence of an ambient context Γ from our adjunction-based moral rule if we had an arrow $dist : \Gamma\,,(A\vee B)\longrightarrow(\Gamma\wedge A)\vee(\Gamma\wedge B)$, since then we could construct:

$$\Gamma \xrightarrow{\langle \mathrm{id}\,,\mathcal{E}\rangle} \Gamma\,,(A\vee B) \xrightarrow{\ dist\ } (\Gamma\wedge A)\vee(\Gamma\wedge B) \xrightarrow{[\mathcal{D}_1\,,\mathcal{D}_2]} C$$

Fortunately, the *distributive law* of products over coproducts in the presence of exponentials (lemma 3.3.2) provides just such an arrow. What's more, since this arrow is an isomorphism, the Gentzen rule is equivalent to the adjoint-theoretic one in any bicartesian closed category.

We derive a rule interchangeable for Gentzen's rule of disjunction elimination by distributing the ambient context across the disjunction. In (pseudo)natural deduction notation:

$$\dfrac{\dfrac{\Gamma \quad \dfrac{\overline{\Gamma}}{\mathcal{E}}}{(\Gamma\wedge A)\vee(\Gamma\wedge B)}\ dist \quad \dfrac{\overline{\Gamma\,,A}}{\dfrac{\mathcal{D}_1}{C}} \quad \dfrac{\overline{\Gamma\,,B}}{\dfrac{\mathcal{D}_2}{C}}}{C}\ \vee-^\star$$

The distributive law is of course admissible in Gentzen's system as:

$$\dfrac{A\vee B \quad \dfrac{\Gamma \quad \overline{A}}{\dfrac{\Gamma\wedge A}{(\Gamma\wedge A)\vee(\Gamma\wedge B)}\ \vee+_1} \quad \dfrac{\Gamma \quad \overline{B}}{\dfrac{\Gamma\wedge B}{(\Gamma\wedge A)\vee(\Gamma\wedge B)}\ \vee+_2}}{(\Gamma\wedge A)\vee(\Gamma\wedge B)}\ \vee-$$

The unit of this adjunction is the ordered pair of coprojections $(inl\,,inr)$. When written as inference rules, they form exactly the ordered pair of introduction rules for disjunction:

$$A \qquad B \qquad \overset{\eta}{\longmapsto} \qquad \dfrac{A}{A\vee B}\ \vee+_1 \quad \dfrac{B}{A\vee B}\ \vee+_2$$

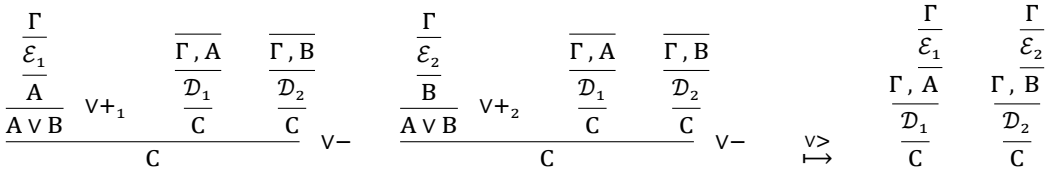
We can now see how the factorization in the universal property of the unit,

$$(inl\,,inr)\cdot\Delta[\mathcal{D}_1\,,\mathcal{D}_2] \quad \overset{\beta'}{=} \quad (\mathcal{D}_1\,,\mathcal{D}_2)$$

expresses the local reduction for disjunction in the category Prop × Prop using the adjoint-theoretic moral elimination rule:

$$\dfrac{\dfrac{A}{A\vee B}\ \vee+_1 \quad \dfrac{\overline{A}}{\dfrac{\mathcal{D}_1}{C}} \quad \dfrac{\overline{B}}{\dfrac{\mathcal{D}_2}{C}}}{C}\ \vee-^\star \qquad \dfrac{\dfrac{B}{A\vee B}\ \vee+_2 \quad \dfrac{\overline{A}}{\dfrac{\mathcal{D}_1}{C}} \quad \dfrac{\overline{B}}{\dfrac{\mathcal{D}_2}{C}}}{C}\ \vee-^\star \qquad \overset{\vee>^\star}{\longmapsto} \qquad \dfrac{\dfrac{A}{\dfrac{\mathcal{D}_1}{C}} \quad \dfrac{B}{\dfrac{\mathcal{D}_2}{C}}}{}$$

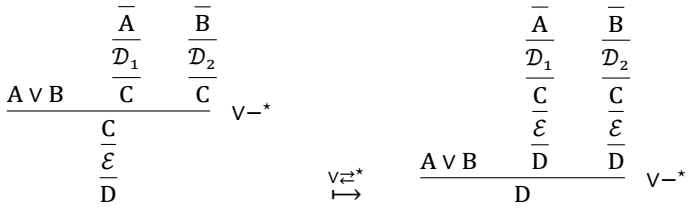
Adding an ambient context poses no problems since it just means precomposing the distributive law. In this case, we get the local reduction:



The naturality of the hom set bijection of this adjunction in the codomain coordinate,

$$[\mathcal{D}_1\,,\mathcal{D}_2]\cdot\mathcal{E} \quad = \quad [\mathcal{D}_1\cdot\mathcal{E}\,,\mathcal{D}_2\cdot\mathcal{E}]$$

says that any derivation postcomposed to a $\vee-^\star$ rule may be moved into the minor branch by duplication:

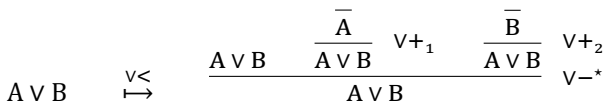


This allows us to unite a β-redex that may be split between the the minor branch and the conclusion of the rule. Adding an ambient context again just means precomposing the distributive law.

The equation for the adjoint complement of a unit component,

$$\mathrm{id}_{-\vee-} \quad = \quad [inl\,,inr]$$

expresses the local expansion for disjunction:



Precomposing an arbitrary derivation yields the version in table 2.3.

The arrow $\mathcal{D}_1 \vee \mathcal{D}_2$ appearing in in diagram (5.2) reminds us that we may extend the definition of disjunction to derivations. In particular, the characterization of the adjoint functor image of arrows (lemma 3.1.4) tells us that for any derivations, $\mathcal{D}_1$ and $\mathcal{D}_2$,

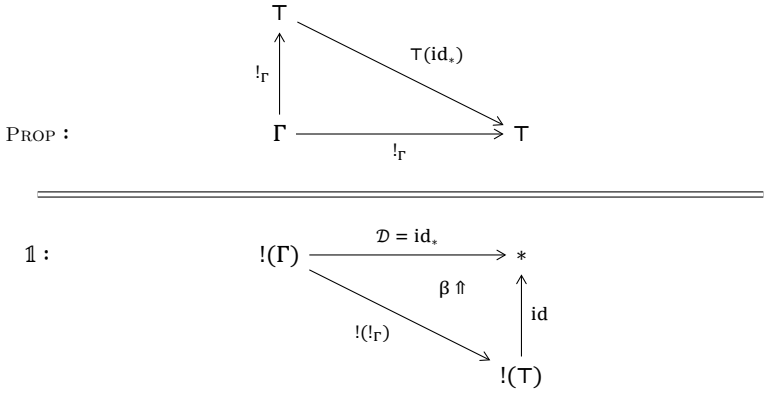
$$\mathcal{D}_1 \vee \mathcal{D}_2 \quad = \quad [\mathcal{D}_1\cdot inl\,,\mathcal{D}_2\cdot inr]$$

In natural deduction notation this yields the definition:

$$\dfrac{\dfrac{A \vee B}{\mathcal{D}_1 \vee \mathcal{D}_2}}{C \vee D} \quad \overset{\vee\to}{\coloneqq} \quad \dfrac{A \vee B \qquad \dfrac{\dfrac{\overline{A}}{\mathcal{D}_1}}{\dfrac{C}{C \vee D}}\vee+_1 \qquad \dfrac{\dfrac{\overline{B}}{\mathcal{D}_2}}{\dfrac{D}{C \vee D}}\vee+_2}{C \vee D}\vee-^\star$$

5.3 Truth

Recall that we interpret truth as a terminal object and that the constant functor picking it out is right adjoint to the only functor to a *terminal category* ($! \dashv \top$). We can summarize this adjunction with the following instance of diagram (3.4):



(5.3)

Here the triangle in $\mathbb{1}$ is degenerate since id_* is the only arrow in the whole category. Thus $\eta(\Gamma) = !_\Gamma$ is the unique arrow from Γ to $\top$.

In this adjunction, the adjoint complement operation $-^\flat$ takes an identity derivation on $*$ to a derivation of $\top$ from arbitrary assumptions:

$$\mathcal{D} = \mathsf{id}_* \;:\; !(\Gamma) \longrightarrow * \qquad \overset{-^\flat}{\longmapsto} \qquad !_\Gamma \;:\; \Gamma \longrightarrow \top$$

We can express this with the following inference rule of natural deduction:

$$* \quad \overset{-^\flat}{\longmapsto} \quad \dfrac{\Gamma \qquad \overline{*}}{\top}\top+^\star$$

Of course, the minor subderivation of this rule is trivial. Eliding it gives us Gentzen's rule.

The counit of this adjunction has only one component, id_*. By rights, this should provide the elimination rules for truth, but it tells us nothing about the category Prop, or any other category for that matter. The absence of derived elimination rules for truth comes as a welcome relief.

The universal property of the counit is just an equation between id_* and itself in the category $\mathbb{1}$, which tells us nothing useful. This comports with the Prawitz interpretation, where having no elimination rules implies that there are no introduction-elimination detours to short-cut.

The naturality of the hom set bijection of this adjunction in the domain coordinate,

$$\mathcal{E} \cdot !_{\mathsf{cod}(\mathcal{E})} \quad = \quad !_{\mathsf{dom}(\mathcal{E})}$$

says that any derivation precomposed to a $\top{+}^\star$ rule may be unceremoniously tossed into the black hole of the minor branch:

$$\dfrac{\dfrac{\dfrac{\Gamma}{\mathcal{E}}}{A} \quad \overline{*}}{\top}\ \top{+}^\star \qquad \overset{\top \rightleftarrows^\star}{\longmapsto} \qquad \dfrac{\Gamma \quad \overline{*}}{\top}\ \top{+}^\star$$

The equation for the adjoint complement of a counit component,

$$\mathsf{id}_\top \quad = \quad !_\top$$

expresses the local expansion for truth:

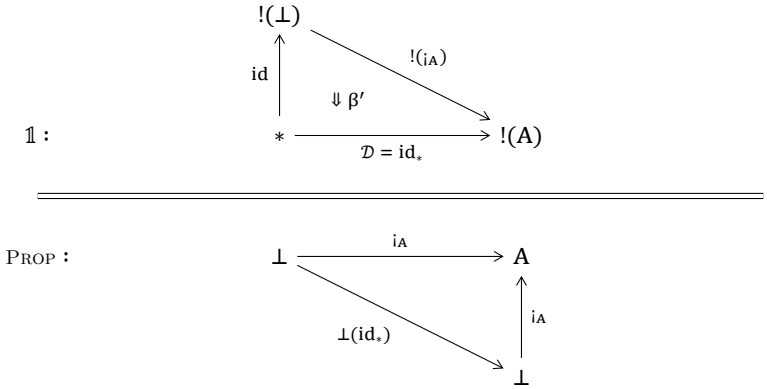
$$\top \qquad \overset{\top <^\star}{\longmapsto} \qquad \dfrac{\top \quad \overline{*}}{\top}\ \top{+}^\star$$

We recover the formulation of table 2.3 by precomposing an arbitrary derivation and applying the permutation conversion.

As a functor, $\top$ has no choice but to take id_* to $\mathsf{id}_\top$.

5.4 Falsehood

Recall that we interpret falsehood as an initial object and that the constant functor picking it out is left adjoint to the only functor to a *terminal category* ($\bot \dashv !$). We can summarize this adjunction with the following instance of diagram (3.4):



(5.4)

Again, the triangle in $\mathbb{1}$ is degenerate, making $\varepsilon(A) = \text{¡}_A$ the unique arrow from $\bot$ to A.

In this adjunction, the adjoint complement operation $-^\sharp$ takes an identity derivation on $*$ to a derivation of an arbitrary goal from $\bot$:

$$\mathcal{D} = \mathrm{id}_* \ : \ * \longrightarrow !(A) \qquad \overset{-^\sharp}{\longmapsto} \qquad \text{¡}_A \ : \ \bot \longrightarrow A$$

We can express this with the following inference rule of natural deduction:

$$* \quad \overset{-^\sharp}{\longmapsto} \quad \dfrac{\bot \quad \overline{*}}{A}\ \bot\!-^\star$$

As with $\top\!+^\star$, eliding the trivial minor subderivation gives us Gentzen's rule. Although $\bot$ is a left connective like $\vee$, there is no issue with ambient contexts here because all information about the context gets destroyed in the category $\mathbb{1}$ anyway.

The unit of this adjunction has only the component, id_*. Like the counit for $\top$, this tells us nothing about the category Prop, leaving us without an introduction rule for $\bot$, as we would expect. Since the universal property of the unit tells us nothing, there is no local reduction for $\bot$ either.

The naturality of the hom set bijection of this adjunction in the codomain coordinate,

$$\text{¡}_{\mathrm{dom}(\mathcal{E})} \cdot \mathcal{E} \quad = \quad \text{¡}_{\mathrm{cod}(\mathcal{E})}$$

says that any derivation postcomposed to a $\bot\!-^\star$ rule may be unceremoniously tossed into the black hole of the minor branch:

$$\dfrac{\dfrac{\bot \quad \overline{*}}{A}\ \bot\!-^\star}{\dfrac{\mathcal{E}}{B}} \qquad \overset{\bot\rightleftarrows^\star}{\longmapsto} \qquad \dfrac{\bot \quad \overline{*}}{B}\ \bot\!-^\star$$

The equation for the adjoint complement of a unit component,

$$\mathrm{id}_\bot \quad = \quad \text{¡}_\bot$$

expresses the local expansion for falsehood:

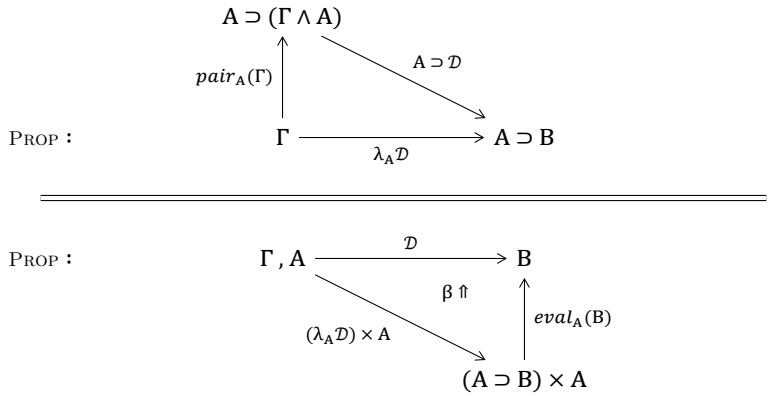
$$\bot \quad \overset{\bot<^\star}{\longmapsto} \quad \dfrac{\bot \quad \overline{*}}{\bot}\ \bot\!-^\star$$

Precomposing an arbitrary derivation yields the version in table 2.3.

As a functor, $\bot$ has no choice but to take id_* to $\mathrm{id}_\bot$.

5.5 Implication

Recall that we interpret implication as the exponential and that exponentiation is right adjoint to the cartesian product by the *curry adjunction* $(- \times \mathsf{A} \dashv \mathsf{A} \supset -)$. We can summarize this adjunction with the following instance of diagram (3.4):



(5.5)

Here λ is the abstraction operation (currying), eval is evaluation (modus ponens) and $\mathit{pair}_{\mathsf{A}} \coloneqq \lambda_{\mathsf{A}}(\mathsf{id}_{- \times \mathsf{A}})$ is the pairing operation that is the currying of the identity map on $- \times \mathsf{A}$.

In this adjunction, the adjoint complement operation $-^\flat$ takes a derivation of B under the assumption A to a derivation of $\mathsf{A} \supset \mathsf{B}$ without that assumption:

$$\mathcal{D} \ : \ \Gamma \, , \mathsf{A} \longrightarrow \mathsf{B} \qquad \overset{-^\flat}{\longmapsto} \qquad \lambda_{\mathsf{A}} \mathcal{D} \ : \ \Gamma \longrightarrow \mathsf{A} \supset \mathsf{B}$$

We express this with the following inference rule of natural deduction:

$$\dfrac{\dfrac{\Gamma \, , \mathsf{A}}{\vdots \, \mathcal{D}}}{\mathsf{B}} \quad \overset{-^\flat}{\longmapsto} \quad \dfrac{\Gamma \qquad \dfrac{\overline{\Gamma \, , \mathsf{A}}}{\vdots \, \mathcal{D}}}{\mathsf{A} \supset \mathsf{B}} \ \supset\!+^\star$$

This rule is interchangeable for Gentzen's rule with an explicit ambient context.

The counit of this adjunction is the evaluation map $\mathit{eval}_{\mathsf{A}}$. When written as an inference rule, this forms a moral elimination rule for implication:

$$\mathsf{B} \quad \overset{\varepsilon}{\longmapsto} \quad \dfrac{(\mathsf{A} \supset \mathsf{B}) \times \mathsf{A}}{\mathsf{B}} \ \supset\!-^\star$$

But this leads to a small difficulty. In Gentzen's system, the rule $\supset -$ has two subderivations. If we interpret them as a derivation in the product category $\textsc{Prop} \times \textsc{Prop}$, then $\supset -$ cannot

be an arrow in the category Prop. One solution is to interpret the Gentzen $\supset -$ rule as the composition $(\wedge +^\star) \cdot (\supset -^\star)$:

$$\dfrac{\Gamma \quad \dfrac{\overline{\Gamma}}{\dfrac{}{\mathcal{D}_1}} \quad \dfrac{\overline{\Gamma}}{\dfrac{}{\mathcal{D}_2}}}{\dfrac{(\mathrm{A} \supset \mathrm{B}) \wedge \mathrm{A}}{\mathrm{B}}\ \supset -^\star}\ \wedge +^\star$$

Alternatively, we may think of the two subderivations not as a derivation in a product category but rather as a product of derivations in the category Prop.

In any case, the factorization in the universal property of the counit,

$$(\lambda_{\mathrm{A}} \mathcal{D}) \times \mathrm{A} \cdot \mathit{eval}_{\mathrm{A}} \quad \overset{\beta}{=} \quad \mathcal{D}$$

expresses the local reduction for implication:

$$\dfrac{\Gamma \quad \dfrac{\dfrac{\overline{\Gamma, \mathrm{A}}}{\dfrac{}{\mathcal{D}}}}{\mathrm{B}}}{\dfrac{\mathrm{A} \supset \mathrm{B} \qquad \mathrm{A}}{\mathrm{B}}\ \supset -}\ \supset +^\star \qquad \overset{\supset >^\star}{\mapsto} \qquad \dfrac{\dfrac{\Gamma, \mathrm{A}}{\dfrac{}{\mathcal{D}}}}{\mathrm{B}}$$

The naturality of the hom set bijection of this adjunction in the domain coordinate,

$$\mathcal{E} \cdot \lambda_{\mathrm{A}} \mathcal{D} \quad = \quad \lambda_{\mathrm{A}} (\mathcal{E} \times \mathrm{A} \cdot \mathcal{D})$$

says that any derivation precomposed to a $\supset +^\star$ rule may be moved into the minor branch by forming the product with the identity derivation on A:

$$\dfrac{\dfrac{\Gamma}{\dfrac{}{\mathcal{E}}} \quad \dfrac{\overline{\mathrm{C}, \mathrm{A}}}{\dfrac{}{\mathcal{D}}}}{\dfrac{\mathrm{C}}{\mathrm{A} \supset \mathrm{B}}\ \supset +^\star} \qquad \overset{\supset \rightleftarrows^\star}{\mapsto} \qquad \dfrac{\Gamma \quad \dfrac{\dfrac{\overline{\Gamma}}{\dfrac{}{\mathcal{E}}}}{\dfrac{\mathrm{C}, \overline{\mathrm{A}}}{\dfrac{}{\mathcal{D}}}}}{\mathrm{A} \supset \mathrm{B}}\ \supset +^\star$$

Here C may internalize a context.

The equation for the adjoint complement of a counit component,

$$\mathrm{id}_{\mathrm{A} \supset -} \quad = \quad \lambda_{\mathrm{A}} (\mathit{eval}_{\mathrm{A}}(-))$$

expresses the local expansion for implication:

$$\mathrm{A} \supset \mathrm{B} \qquad \overset{\supset <^\star}{\mapsto} \qquad \dfrac{\mathrm{A} \supset \mathrm{B} \qquad \dfrac{\overline{(\mathrm{A} \supset \mathrm{B}) \times \mathrm{A}}}{\mathrm{B}}\ \supset -^\star}{\mathrm{A} \supset \mathrm{B}}\ \supset +^\star$$

We recover the formulation of table 2.3 by precomposing an arbitrary derivation and applying the permutation conversion.

Finally, implication may be extended to a functor in its covariant coordinate by lemma 3.1.4:

$$\mathrm{A} \supset \mathcal{D} \quad = \quad \lambda_{\mathrm{A}}(eval_{\mathrm{A}} \cdot \mathcal{D})$$

In natural deduction notation this yields the definition:

$$\dfrac{\dfrac{\mathrm{A} \supset \mathrm{B}}{\mathrm{A} \supset \mathcal{D}}}{\mathrm{A} \supset \mathrm{C}} \quad \overset{\supset\rightarrow}{:=} \quad \dfrac{\mathrm{A} \supset \mathrm{B} \qquad \dfrac{\dfrac{\dfrac{\overline{(\mathrm{A} \supset \mathrm{B}) \times \mathrm{A}}}{\mathrm{B}}\ \supset\!-^{\star}}{\mathcal{D}}}{\mathrm{C}}}{\mathrm{A} \supset \mathrm{C}}\ \supset\!+^{\star}$$

5.6 Universal Quantification

We interpret universal quantification as the right adjoint to reindexing by the interpretation of a *single omission* ($\hat{x}^* \dashv \forall x$). We can summarize this adjunction with the following instance of diagram (3.4):

$$\begin{array}{lc}
\textsc{Prop}(\Phi): & \begin{array}{ccc} \forall x : \mathrm{X} \,.\, \hat{x}^*\Gamma & & \\ {\scriptstyle \eta_{\forall x}(\Gamma)} \uparrow & \searrow^{\forall x : \mathrm{X} \,.\, \mathcal{D}} & \\ \Gamma & \xrightarrow[gen_x \mathcal{D}]{} & \forall x : \mathrm{X} \,.\, \mathrm{A} \end{array} \\
\hline\hline
\textsc{Prop}(\Phi, x : \mathrm{X}): & \begin{array}{ccc} \hat{x}^*\Gamma & \xrightarrow{\mathcal{D}} & \mathrm{A} \\ & {\scriptstyle \beta \Uparrow} & \\ {\scriptstyle \hat{x}^*(gen_x \mathcal{D})} \searrow & & \uparrow {\scriptstyle \varepsilon_{\forall x}(\mathrm{A})} \\ & \hat{x}^*(\forall x : \mathrm{X} \,.\, \mathrm{A}) & \end{array}
\end{array} \tag{5.6}$$

Here gen is the operation of universal generalization. We do not know of any commonly used names for the unit and counit of this adjunction. Reindexing by a single omission just adds a dummy variable and is invisible in the concrete syntax of propositions and derivations. However, we will sometimes write these reindexings explicitly to emphasize the adjoint structure we are dealing with.

In this adjunction, the adjoint complement operation $-^\flat$ takes a derivation from assumptions in which a given variable does not occur free to a derivation of the universal

quantification over that variable of the goal:

$$\mathcal{D} \;:\; \hat{x}^*\Gamma \longrightarrow \mathrm{A} \qquad \overset{\_^\flat}{\longmapsto} \qquad gen_x\mathcal{D} \;:\; \Gamma \longrightarrow \forall x : \mathrm{X} \,.\, \mathrm{A}$$

We can express this with the following inference rule of natural deduction:

$$\dfrac{\dfrac{\hat{x}^*\Gamma}{\mathcal{D}}}{\mathrm{A}} \quad \overset{\_^\flat}{\longmapsto} \quad \dfrac{\Gamma \qquad \dfrac{\overline{\hat{x}^*\Gamma}}{\mathcal{D}}}{\forall x : \mathrm{X} \,.\, \mathrm{A}}\ \forall{+}^\star$$

This is our moral introduction rule for universal quantification. It is equivalent to the Gentzen rule with the context variable x acting as the eigenvariable. Observe that the side condition of the Gentzen rule is automatically enforced since the trunk of the derivation is in the category $\textsc{Prop}(\Phi)$, where the variable x is not in scope.

The counit of this adjunction when written as an inference rule becomes our moral elimination rule:

$$\mathrm{A} \quad \overset{\varepsilon}{\longmapsto} \quad \dfrac{\hat{x}^*(\forall x : \mathrm{X} \,.\, \mathrm{A})}{\mathrm{A}}\ \forall{-}^\star$$

The corresponding Gentzen rule allows us to conclude any instance of the formula A in which a type-appropriate term t is substituted for the variable x, but any remaining free variables of A are left undisturbed. Categorically, this corresponds to reindexing by the interpretation of the *single substitution* induced by the term in context $\Phi \mid t : \mathrm{X}$. Thus the Gentzen elimination rule corresponds to $(\varepsilon_{\forall x}(\mathrm{A}))[x{\mapsto}t]$, as shown:

$$\begin{array}{ccc} \forall x : \mathrm{X} \,.\, \mathrm{A} & \hat{x}^*(\forall x : \mathrm{X} \,.\, \mathrm{A}) & \forall x : \mathrm{X} \,.\, \mathrm{A} \\ & \Big\downarrow{\scriptstyle \varepsilon_{\forall x}(\mathrm{A})} & \Big\downarrow{\scriptstyle (\varepsilon_{\forall x}(\mathrm{A}))[x\mapsto t]} \\ & \mathrm{A} & \mathrm{A}[x{\mapsto}t] \end{array} \tag{5.7}$$

$$\Phi \underset{\mathrm{id}}{\overset{\hat{x}}{\rightleftarrows}} \Phi \,,\, x : \mathrm{X} \xleftarrow{[x\mapsto t]} \Phi$$

The Gentzen rule gives us a result that is in some sense stronger by yielding a derivation in the typing context Φ. But in exchange it requires us to immediately choose a term of type X in that context to act as the *representative* of its type under consideration. The adjoint-theoretic moral rule takes a more relaxed approach. It allows us the option, but not the obligation, of supplying a representative term at some point in the future, while initially selecting the context variable x. To wit, the context variable x is a term of type X, but only in the extended context $\Phi \,,\, x : \mathrm{X}$, and not in the original context Φ. At any point we may

choose a representative term $\Phi \mid t : \mathrm{X}$ to use in place of this context variable by performing the substitution $[x \mapsto t]$. Because substitution commutes with the propositional connectives (lemma 4.6.3) and quantifiers (lemma 4.6.4), it does not matter when we eventually supply this term. But until we do so our derivation is valid only in the extended context, a world in which a term of the specified type exists by fiat, in the form of the context variable. The moral version has the advantage that we need not decide *which* term t to substitute for x in A at the time we apply the rule. We will use this property to good effect when we turn our attention to proof search in the sequel.

The factorization in the universal property of the counit,

$$\hat{x}^*(gen_x \mathcal{D}) \cdot \varepsilon_{\forall x} \quad \overset{\beta}{=} \quad \mathcal{D}$$

expresses the local reduction for universal quantification in the category $\textsc{Prop}(\Phi, x : \mathrm{X})$:

$$\dfrac{\dfrac{\hat{x}^*\Gamma \qquad \dfrac{\overline{\hat{x}^*\Gamma}}{\dfrac{\mathcal{D}}{\mathrm{A}}}}{\hat{x}^*(\forall x : \mathrm{X} \,.\, \mathrm{A})}\ \hat{x}^*(\forall{+}^\star)}{\mathrm{A}}\ \forall{-}^\star \qquad \overset{\forall{>}^\star}{\longmapsto} \qquad \dfrac{\dfrac{\hat{x}^*\Gamma}{\mathcal{D}}}{\mathrm{A}}$$

Applying a single substitution for x to both sides corresponds to the version in table 2.2.

The naturality of the hom set bijection of this adjunction in the domain coordinate,

$$\mathcal{E} \cdot gen_x \mathcal{D} \quad = \quad gen_x(\hat{x}^*\mathcal{E} \cdot \mathcal{D})$$

says that any derivation precomposed to a $\forall{+}^\star$ rule may be moved into the minor branch by adding the dummy variable x:

$$\dfrac{\dfrac{\Gamma}{\dfrac{\mathcal{E}}{\mathrm{C}}} \qquad \dfrac{\overline{\hat{x}^*\mathrm{C}}}{\dfrac{\mathcal{D}}{\mathrm{A}}}}{\forall x : \mathrm{X} \,.\, \mathrm{A}}\ \forall{+}^\star \qquad \overset{\forall{\rightleftarrows}^\star}{\longmapsto} \qquad \dfrac{\Gamma \qquad \dfrac{\dfrac{\dfrac{\overline{\hat{x}^*\Gamma}}{\hat{x}^*\mathcal{E}}}{\hat{x}^*\mathrm{C}}}{\dfrac{\mathcal{D}}{\mathrm{A}}}}{\forall x : \mathrm{X} \,.\, \mathrm{A}}\ \forall{+}^\star$$

The equation for the adjoint complement of a counit component,

$$\mathrm{id}_{\forall x : \mathrm{X} . -} \quad = \quad gen_x(\varepsilon_{\forall x} -)$$

expresses the local expansion for universal quantification:

$$\forall x : \mathrm{X} \,.\, \mathrm{A} \qquad \overset{\forall{<}^\star}{\longmapsto} \qquad \dfrac{\forall x : \mathrm{X} \,.\, \mathrm{A} \qquad \dfrac{\overline{\hat{x}^*(\forall x : \mathrm{X} \,.\, \mathrm{A})}}{\mathrm{A}}\ \forall{-}^\star}{\forall x : \mathrm{X} \,.\, \mathrm{A}}\ \forall{+}^\star$$

We recover a formulation equivalent to that of table 2.3 by precomposing an arbitrary derivation and applying the permutation conversion.

Universal quantification may be extended to a functor by using lemma 3.1.4 to determine its action on arrows:

$$\forall x : \mathrm{X} \,.\, \mathcal{D} \quad = \quad gen_x(\varepsilon_{\forall x} \cdot \mathcal{D})$$

In natural deduction notation this yields the definition:

$$\dfrac{\dfrac{\forall x : \mathrm{X} \,.\, \Gamma}{\forall x : \mathrm{X} \,.\, \mathcal{D}}}{\forall x : \mathrm{X} \,.\, \mathrm{A}} \quad \overset{\forall\to}{:=} \quad \dfrac{\forall x : \mathrm{X} \,.\, \Gamma \qquad \dfrac{\dfrac{\overline{\hat{x}^*(\forall x : \mathrm{X} \,.\, \Gamma)}}{\Gamma}}{\mathcal{D}}}{\forall x : \mathrm{X} \,.\, \mathrm{A}}\ \forall{+}^\star \ \forall{-}^\star$$

5.7 Existential Quantification

Finally, we interpret existential quantification as the left adjoint to reindexing by the interpretation of a *single omission* ($\exists x \dashv \hat{x}^*$). We can summarize this adjunction with the following instance of diagram (3.4):

$$\begin{array}{ccc} & \hat{x}^*(\exists x : \mathrm{X} \,.\, \mathrm{A}) & \\ {\scriptstyle \eta_{\exists x}(\mathrm{A})} \uparrow & \Downarrow \beta' & \searrow {\scriptstyle \hat{x}^*(ind_x\mathcal{D})} \\ \textsc{Prop}(\Phi, x : \mathrm{X}) : & \mathrm{A} & \xrightarrow[\mathcal{D}]{} \hat{x}^*\mathrm{B} \end{array} \tag{5.8}$$

$$\textsc{Prop}(\Phi) : \quad \exists x : \mathrm{X} \,.\, \mathrm{A} \xrightarrow{ind_x\mathcal{D}} \mathrm{B}, \quad \exists x : \mathrm{X} \,.\, \mathcal{D} : \exists x : \mathrm{X} \,.\, \mathrm{A} \to \exists x : \mathrm{X} \,.\, \hat{x}^*\mathrm{B}, \quad \varepsilon_{\exists x}(\mathrm{B}) : \exists x : \mathrm{X} \,.\, \hat{x}^*\mathrm{B} \to \mathrm{B}$$

Here *ind* is meant to convey indifference about the bound variable.[1]

In this adjunction, the adjoint complement operation $-^\sharp$ takes a derivation with a goal in which a given variable does not occur free to a derivation from the existential quantification over that variable of the assumption:

$$\mathcal{D} \ : \ \mathrm{A} \longrightarrow \hat{x}^*\mathrm{B} \qquad \overset{-^\sharp}{\longmapsto} \qquad ind_x\mathcal{D} \ : \ \exists x : \mathrm{X} \,.\, \mathrm{A} \longrightarrow \mathrm{B}$$

[1] We feel tempted to use the phrase "existential indifference" here, but fear that it may already have an altogether different meaning.

We can express this with the following inference rule of natural deduction:

$$\dfrac{\dfrac{\dfrac{\mathrm{A}}{\mathcal{D}}}{\hat{x}^*\mathrm{B}}}{} \quad \overset{-^\sharp}{\longmapsto} \quad \dfrac{\exists x : \mathrm{X} \,.\, \mathrm{A} \qquad \dfrac{\dfrac{\overline{\mathrm{A}}}{\mathcal{D}}}{\hat{x}^*\mathrm{B}}}{\mathrm{B}}\ \exists -^\star$$

This is our moral elimination rule for existential quantification. As with the moral rule for universal introduction, the context variable x plays the role of the eigenvariable and Gentzen's side condition is automatically enforced because the trunk of the derivation is in the category $\textsc{Prop}(\Phi)$, where x is not in scope.

As in the case for disjunction elimination, however, this rule is at least not obviously equivalent to Gentzen's rule because it does not account for an ambient propositional context. If we write out Gentzen's rule with explicit global assumptions, it becomes:

$$\dfrac{\dfrac{\Gamma}{\dfrac{\mathcal{E}}{\exists x : \mathrm{X} \,.\, \mathrm{A}}} \qquad \dfrac{\Gamma \,,\, \overline{e : \mathrm{X}} \,,\, \overline{\mathrm{A}[x \mapsto e]}}{\dfrac{\mathcal{D}}{\mathrm{B}}}}{\mathrm{B}}\ \exists -^\dagger$$

As in the case of disjunction, we reason that the global context must be valid in the fiber of the trunk of the derivation, $\textsc{Prop}(\Phi)$. Therefore, it is the syntactically identical $\hat{x}^*$-image of Γ that appears in the minor subderivation in $\textsc{Prop}(\Phi \,,\, x : \mathrm{X})$.

Using a conjunction to internalize the context, we have the natural bijection:

$$\mathcal{D} \;:\; \hat{x}^*\Gamma \,,\, \mathrm{A} \longrightarrow \hat{x}^*(\mathrm{B}) \qquad \overset{-^\sharp}{\longmapsto} \qquad ind_x\mathcal{D} \;:\; \exists x : \mathrm{X} \,.\, (\hat{x}^*\Gamma \wedge \mathrm{A}) \longrightarrow \mathrm{B}$$

We could recover Gentzen's elimination rule for existential quantification in the presence of an ambient context Γ from our adjunction-based moral rule if we had an arrow $frob : \Gamma \,,\, \exists x : \mathrm{X} \,.\, \mathrm{A} \longrightarrow \exists x : \mathrm{X} \,.\, (\hat{x}^*\Gamma \wedge \mathrm{A})$, since then we could construct:

$$\Gamma \xrightarrow{\langle \mathrm{id} \,,\, \mathcal{E} \rangle} \Gamma \,,\, \exists x : \mathrm{X} \,.\, \mathrm{A} \xrightarrow{frob} \exists x : \mathrm{X} \,.\, (\hat{x}^*\Gamma \wedge \mathrm{A}) \xrightarrow{ind_x\mathcal{D}} \mathrm{B}$$

This time we are saved by *Frobenius reciprocity* (lemma 3.3.3), which provides just such an arrow. Because this arrow is invertible, the Gentzen rule is equivalent to adjoint-theoretic one in a hyperdoctrine. Like the distributive law for disjunction, Frobenius reciprocity allows a left connective to be compatible with contexts by providing a means of distributing an ambient context across the connective.

We derive a rule interchangeable for Gentzen's rule of existential elimination by precomposing Frobenius reciprocity with the moral rule. In (pseudo)natural deduction notation:

$$\dfrac{\dfrac{\Gamma}{\mathcal{E}}}{\dfrac{\Gamma \qquad \exists x : \mathrm{X} \,.\, \mathrm{A}}{\exists x : \mathrm{X} \,.\, \hat{x}^*\Gamma \wedge \mathrm{A}}\ frob \qquad \dfrac{\dfrac{\hat{x}^*\Gamma \,,\, \mathrm{A}}{\mathcal{D}}}{\hat{x}^*\mathrm{B}}}{\mathrm{B}}\ \exists{-}^\star$$

The admissibility of the rule $frob$ in Gentzen's system is analogous to that of $dist$.

The unit of this adjunction when written as an inference rule becomes our moral introduction rule:

$$\mathrm{A} \quad \overset{\eta}{\longmapsto} \quad \dfrac{\mathrm{A}}{\hat{x}^*(\exists x : \mathrm{X} \,.\, \mathrm{A})}\ \exists{+}^\star$$

Dual to the case of $\forall{-}^\star$, we recover the Gentzen rule by reindexing along a *single substitution*:

$$\begin{array}{ccc} & \mathrm{A} & \mathrm{A}[x \mapsto t] \\ & \Big\downarrow{\scriptstyle \eta_{\exists x}(\mathrm{A})} & \Big\downarrow{\scriptstyle (\eta_{\exists x}(\mathrm{A}))[x \mapsto t]} \\ \exists x : \mathrm{X} \,.\, \mathrm{A} & \hat{x}^*(\exists x : \mathrm{X} \,.\, \mathrm{A}) & \exists x : \mathrm{X} \,.\, \mathrm{A} \end{array} \tag{5.9}$$

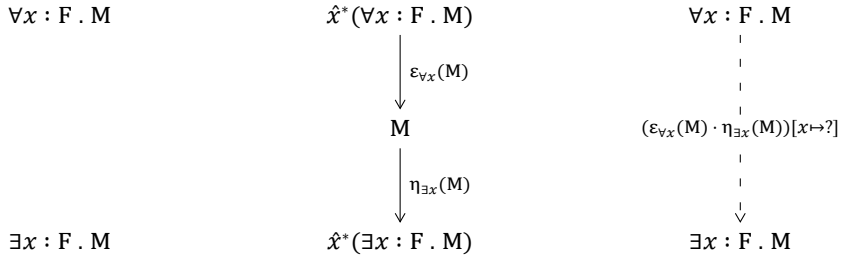
$$\Phi \xleftarrow{\ \hat{x}\ } \Phi \,,\, x : \mathrm{X} \xleftarrow{\ [x \mapsto t]\ } \Phi \qquad \text{id} : \Phi \to \Phi$$

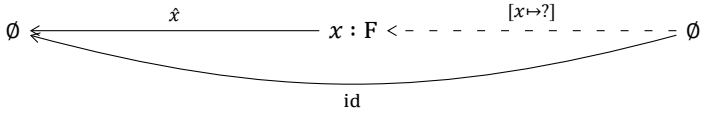
This decomposition of the rule into a purely logical rule and a substitution has important implications for the operational semantics of proof search. In particular, it means that we need not select a *witness* term at the time we apply the rule, but may instead temporarily use the context variable x. This has the effect of leaving x as a free variable (i.e. metavariable or *logic variable*) to be instantiated later, for example through *unification*. But it also means that in order to obtain a derivation in the world in which we started, the typing context Φ, we must eventually choose a witness term $\Phi \mid t : \mathrm{X}$ to substitute for the context variable x. Otherwise, we have a derivation that is valid only in a world where such a term is assumed to exist.

The importance of knowing which fiber we are in is illustrated by the attempt to prove the existence of a magical fairy from the well-known fact that all fairies are indeed magical:

Example 5.7.1 (magical fairies) If we allow F to represent the type of fairies and M the

predicate that x is magical, then we may derive $x : \mathrm{F} \mid \forall x : \mathrm{F} \,.\, \mathrm{M} \vdash \exists x : \mathrm{F} \,.\, \mathrm{M}$ as shown.





Thus the fulfillment of childhood dreams awaits us in our own world – just as soon as we are able to produce any fairy whatsoever.

The factorization in the universal property of the unit,

$$\eta_{\exists x} \cdot \hat{x}^*(ind_x \mathcal{D}) \quad \overset{\beta'}{=} \quad \mathcal{D}$$

expresses the local reduction for existential quantification in the category $\textsc{Prop}(\Phi, x : \mathrm{X})$:

$$\dfrac{\dfrac{\mathrm{A}}{\hat{x}^*(\exists x : \mathrm{X} \,.\, \mathrm{A})}\ \exists{+}^\star \quad \dfrac{\dfrac{\overline{\mathrm{A}}}{\mathcal{D}}}{\hat{x}^*\mathrm{B}}}{\hat{x}^*\mathrm{B}}\ \hat{x}^*(\exists{-}^\star) \qquad \overset{\exists{>}^\star}{\longmapsto} \qquad \dfrac{\dfrac{\mathrm{A}}{\mathcal{D}}}{\hat{x}^*\mathrm{B}}$$

Precomposing Frobenius reciprocity and applying a single substitution for x to both sides corresponds to the version in table 2.2.

The naturality of the hom set bijection of this adjunction in the codomain coordinate,

$$ind_x(\mathcal{D}) \cdot \mathcal{E} \quad = \quad ind_x(\mathcal{D} \cdot \hat{x}^* \mathcal{E})$$

says that any derivation postcomposed to a $\exists{-}^\star$ rule may be moved into the minor branch by adding the dummy variable x:

$$\dfrac{\dfrac{\exists x : \mathrm{X} \,.\, \mathrm{A} \qquad \dfrac{\dfrac{[\mathrm{A}]}{\mathcal{D}}}{\hat{x}^*\mathrm{B}}}{\mathrm{B}}}{\dfrac{\mathcal{E}}{\mathrm{C}}}\ \exists{-}^\star \qquad \overset{\exists{\rightleftarrows}^\star}{\longmapsto} \qquad \dfrac{\exists x : \mathrm{X} \,.\, \mathrm{A} \qquad \dfrac{\dfrac{\dfrac{[\mathrm{A}]}{\mathcal{D}}}{\hat{x}^*\mathrm{B}}}{\dfrac{\hat{x}^*\mathcal{E}}{\hat{x}^*\mathrm{C}}}}{\mathrm{C}}\ \exists{-}^\star$$

Adding an ambient context again means precomposing Frobenius reciprocity.

The equation for the adjoint complement of a unit component,

$$\mathrm{id}_{\exists x:\mathrm{X}.-} \quad = \quad ind_x(\eta_{\exists x} -)$$

expresses the local expansion for existential quantification:

$$\exists x : \mathrm{X} \,.\, \mathrm{A} \quad \overset{\exists <^\star}{\longmapsto} \quad \dfrac{\exists x : \mathrm{X} \,.\, \mathrm{A} \qquad \dfrac{\overline{\mathrm{A}}}{\hat{x}^*(\exists x : \mathrm{X} \,.\, \mathrm{A})}\ \exists +^\star}{\exists x : \mathrm{X} \,.\, \mathrm{A}}\ \exists -^\star$$

Finally, existential quantification may be extended to a functor by using lemma 3.1.4 to determine its action on arrows:

$$\exists x : \mathrm{X} \,.\, \mathcal{D} \quad = \quad ind_x(\mathcal{D} \cdot \eta_{\exists x})$$

In natural deduction notation this yields the definition:

$$\dfrac{\dfrac{\exists x : \mathrm{X} \,.\, \mathrm{A}}{\exists x : \mathrm{X} \,.\, \mathcal{D}}}{\exists x : \mathrm{X} \,.\, \mathrm{B}} \quad \overset{\exists \to}{\coloneqq} \quad \dfrac{\exists x : \mathrm{X} \,.\, \mathrm{A} \qquad \dfrac{\dfrac{\overline{\mathrm{A}}}{\mathcal{D}}}{\hat{x}^*(\exists x : \mathrm{X} \,.\, \mathrm{B})}\ \exists +^\star}{\exists x : \mathrm{X} \,.\, \mathrm{B}}\ \exists -^\star$$

This completes the proof of theorem 5.0.1.

5.8 Genericity of Free Hyperdoctrines

Using the interpretations of natural deduction inference rules and derivation conversions described in theorem 5.0.1, we can see that *hyperdoctrines interpretations* of natural deduction derivations are *sound*.

Corollary 5.8.1 (soundness of hyperdoctrine interpretations for natural deduction) To any natural deduction derivation $\mathcal{D}$ of $\Phi \mid \Gamma \vdash \mathrm{A}$ in intuitionistic first-order logic and any interpretation $[\![-]\!]$ of the language of $\mathcal{D}$ in a hyperdoctrine P, there corresponds an arrow $[\![\mathcal{D}]\!] : \mathrm{P}([\![\Phi]\!])\,([\![\Gamma]\!] \to [\![\mathrm{A}]\!])$. Furthermore, the extension of interpretations to natural deduction derivations is well-defined, in the sense that derivations that are equivalent under the conversion relations are interpreted by the same arrow.

Proof. We have just seen how each inference rule of Gentzen's natural deduction can be built from an adjoint-theoretic "moral" rule that is interpreted by an arrow in the fibers of a hyperdoctrine. The composition of primitive inferences is interpreted by the composition

of the corresponding arrows and the substitution of a term for a free variable in a proposition or derivation is interpreted by reindexing between fibers. Thus the composition of the arrows in the *trunk* of the moral version of $\mathcal{D}$ is an arrow in the specified fiber and hom set.

Derivation interpretations are well-defined since the interpretation of each of the conversion relations of local reduction, permutation and local expansion is a consequence of the properties of adjunctions, as described in definition 3.1.1, lemma 3.1.3 and lemma 3.1.2, respectively. □

In addition, due to the coincidence between relations on arrows generated by the conversion relations for natural deduction derivations and those determined by adjunctions, we are able to see that *free hyperdoctrine interpretations* of natural deduction derivations are *generic*.

Corollary 5.8.2 (genericity of free hyperdoctrine interpretations for natural deduction) Let $\mathcal{L}$ be a language of intuitionistic first-order logic and $\textsc{Prop}_{\mathcal{L}}$ be its free hyperdoctrine interpretation. For any propositional context Γ and proposition A of $\mathcal{L}$ in typing context Φ, the set of equivalence classes of natural deduction derivations of $\Phi \mid \Gamma \vdash A$ is in bijection with the hom set $\textsc{Prop}_{\mathcal{L}}(\Phi)\,(\Gamma \to A)$, where the equivalence is that generated by the local reductions, local expansions and permutation conversions.

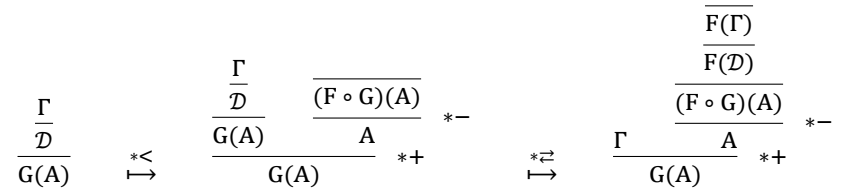
Proof. As described by Barr and Wells in [BW98], generic interpretations are characterized by the properties of having "no junk" and "no confusion". "No junk" means that every arrow of the interpreting category is the interpretation of an inference in the logical system from the interpretation of its premises to that of its conclusion. "No confusion" means that equations between arrows that hold in the interpreting category hold between the interpreted inferences the in the logical system as well.

no junk: Since $\textsc{Prop}_{\mathcal{L}}$ is the free hyperdoctrine generated by $\mathcal{L}$, the only arrows that exist in the fibers are those generated from the universal constructions interpreting the connectives by identity, composition and reindexing. As we have seen, these universal constructions may all be characterized by adjunctions whose units and counits interpret the moral introduction and elimination rules of left and right connectives respectively, and whose natural bijections interpret the dual inference rules.

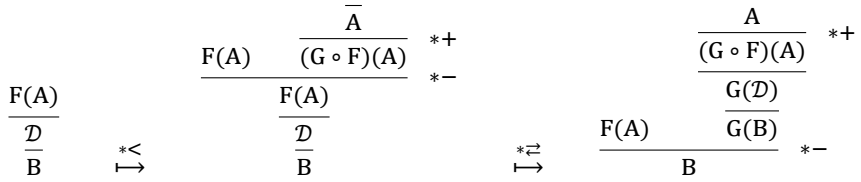
no confusion: The only equations that hold between arrows in a free hyperdoctrine are those imposed by the adjunctions interpreting the connectives. These are generated by the *universal property of the counit* for right connectives and the *universal property of the unit* for left connectives.

The interpretations of the moral elimination rules of right connectives and introduction rules of left connectives are natural transformations because these rules are schematic in their propositional variables. The introduction rules of right connectives and elimination rules of left connectives provide adjoint complements to arrows from left adjoint images, and to right adjoint images, respectively. The *local expansions* together with the *permutation conversions* ensure that adjoint complements are unique by guaranteeing that all arrows to right adjoint images, respectively, from left adjoint images, can be formed in this way as follows:

- for right connectives,



- for left connectives,



Finally, the *local reductions* require arrows with adjoint complements to satisfy the universal properties of the counit, respectively, unit. These universal properties characterize adjunctions, thus all equations generated by the adjunctions are forced by the derivation conversions.

□

In summary, we have that normal equivalence classes of natural deduction derivations correspond precisely to arrows in the fibers of free hyperdoctrines. This ensures the soundness and completeness of natural deduction with respect to free hyperdoctrine categorical semantics.

Chapter 6

Categories for Cartesian Logics

The free hyperdoctrine Prop provides an ideal categorical structure in which to interpret the rules of natural deduction for typed intuitionistic first-order logic. It is also perfectly adequate for interpreting natural deduction derivations. This is in large part due to the fact that both the meta-level operation of context extension and the object-level connective conjunction are interpreted by the cartesian product, which permits us to treat propositional contexts as though they were propositions and to construct tuples of derivations. Although we may interpret derivations directly in Prop, the fit here is not ideal. Part of the reason has to do with the fact that within a derivation, members of the global context may be used any number of times. For example, in the derivation,

$$\dfrac{\dfrac{\dfrac{\Gamma}{\mathcal{D}_1}}{\vdots}}{\dfrac{\mathcal{D}_2}{\mathrm{B}}}$$

assumptions from Γ may be used in $\mathcal{D}_2$ as well as in $\mathcal{D}_1$. Indeed, the global assumptions may be used anywhere throughout a derivation, so each of the arrows composed together to form a derivation should "pass along" the global context in case it is needed again further down the chain; except at the very end where it may finally be discarded. It is certainly possible to do this "by hand" using tupling and projection, but it is not a very good fit. To speak evocatively, in Gentzen's orchard, Γs rain down onto derivation trees from heaven, while in the category Prop we try to account for this phenomenon by describing it as a complex irrigation system. So it is worth thinking about how this could be described more naturally, both for convenience, and so as not to obscure the elegant adjoint-theoretic structure we have been investigating.

6.1 Meta-Theoretic Considerations

It is worth pausing to consider the meta-theoretic assumptions at work here. The fact that we may appeal to a derivation's resources (in the form of the assumptions) as many times as we like, including possibly none at all, without having to keep an account means that we are not treating them in a **conservative** fashion. Intuitively, the contents of contexts act like abstract ideas or chemical catalysts, in that they are eternal, as opposed to like physical resources, which may be transformed or consumed by the interactions in which they participate. Furthermore, there is a tacit assumption that the context members may always be disentangled from the consequences derived from them. Such assumptions may seem uncontroversial to mathematicians, but probably less-so to chemists or economists. They are wildly as odds with our understanding of the laws governing physical reality, where conservativity seems to rule. This observation has led to investigations of **linear logics** [Gir87].

We shall call these kinds of nonconservative contexts **cartesian** and logics in which contexts behave this way **cartesian logics**. The rationale is that one way to characterize the cartesian product is as a symmetric monoidal product with duplication and deletion maps. Thus our intuitionistic logic is cartesian, as opposed to linear.

6.2 The Kleisi Category $\textsc{Prop}_\Gamma$

Fortunately, we have already encountered a categorical construction that will allow us to interpret the cartesian nature of contexts. We can construct a category that automatically propagates the global context, and what's more, Prop can be embedded into this category so that the construction doesn't interfere with anything we've done so far.

Let Γ be a propositional context and consider the endofunctor

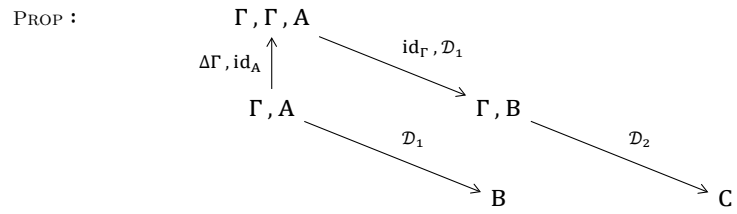
$$\begin{array}{rcl} & \overline{\Gamma\,,-} & \\ \textsc{Prop} & \longrightarrow & \textsc{Prop} \\ A & \longmapsto & \Gamma\,,A \\ \mathcal{D} & \longmapsto & \mathsf{id}_\Gamma\,,\mathcal{D} \end{array} \tag{6.1}$$

We will make a slight notational shorthand and refer to this functor as "Γ". Γ determines a *comonad* on Prop by:

$$\varepsilon_\Gamma \coloneqq \pi' : \Gamma\,,- \longrightarrow - \qquad\qquad \delta_\Gamma \coloneqq \Delta\Gamma\,,\mathsf{id} : \Gamma\,,- \longrightarrow \Gamma\,,\Gamma\,,-$$

It is easy to check that the comonad laws are satisfied. We will call this the **context comonad**.

In the *Kleisli category* of this comonad, $\textsc{Prop}_\Gamma$, an arrow of type $A \longrightarrow B$ is actually a derivation in $\textsc{Prop}$ of type $\Gamma, A \longrightarrow B$. And the composition of $\mathcal{D}_1 : \textsc{Prop}_\Gamma(A \to B)$ and $\mathcal{D}_2 : \textsc{Prop}_\Gamma(B \to C)$ becomes:



For the Kleisli resolution of this comonad, we have:

$$\begin{array}{rcl} & \underline{F_\Gamma} & \\ \textsc{Prop}_\Gamma & \longrightarrow & \textsc{Prop} \\ A & \longmapsto & \Gamma, A \\ \mathcal{D} : A \longrightarrow B & \longmapsto & \langle \pi_\Gamma , \mathcal{D} \rangle : \Gamma, A \longrightarrow \Gamma, B \end{array} \qquad \begin{array}{rcl} & \underline{G_\Gamma} & \\ \textsc{Prop} & \longrightarrow & \textsc{Prop}_\Gamma \\ A & \longmapsto & A \\ \mathcal{D} : A \longrightarrow B & \longmapsto & \pi_A \cdot \mathcal{D} : \textsc{Prop}(\Gamma, A \to B) \end{array}$$

Where $F_\Gamma(\mathcal{D})$ is the *Kleisli extension* of the derivation $\mathcal{D}$:

$$\overline{\mathcal{D}} \quad := \quad (\Delta\Gamma , id) \cdot (\mathrm{id}_\Gamma , \mathcal{D}) \quad = \quad \langle \pi_\Gamma , \mathcal{D} \rangle$$

which is exactly what is needed to propagate the global context throughout a derivation.

Thus we may take a **pure derivation** $\mathcal{E} : \textsc{Prop}(A \to B)$, which does not depend on the global context, and create the **contextual derivation** $G_\Gamma(\mathcal{E}) : \textsc{Prop}(\Gamma, A \to B)$. And from any contextual derivation $\mathcal{D} : \textsc{Prop}(\Gamma, A \to B)$ we may create the **context-propagating derivation** $\overline{\mathcal{D}} \coloneqq F_\Gamma(\mathcal{D}) : \textsc{Prop}(\Gamma, A \to \Gamma, B)$. Finally, we may always discard the context to recover the **pure proposition** B from $(\Gamma, -)(B)$ by postcomposing the component of the counit ε_Γ. And the naturality of the counit together with the *comonad counit law* guarantee that it is harmless to propagate and then later discard a context: $\overline{\mathcal{D}} \cdot \varepsilon(B) = \mathcal{D}$.

6.3 The Polynomial Category $\textsc{Prop}[x]$

The category $\textsc{Prop}_\Gamma$ is isomorphic to a perhaps more intuitive category constructed in [LS86] called the **polynomial category** over $\textsc{Prop}$ and written "$\textsc{Prop}[x]$". The construction proceeds by freely adjoining an arrow $x : \textsc{Prop}(1 \to \Gamma)$, thereby providing a means to obtain a "free copy" of the global context to each object of the category via its component of the natural transformation,

$$\begin{array}{rcl} & \underline{\eta_{[x]}} & \\ \mathrm{id}_{\textsc{Prop}[x]} & \longrightarrow & \Gamma, - \\ A & \longmapsto & \langle !_A \cdot x , \mathrm{id}_A \rangle : A \longrightarrow \Gamma, A \end{array}$$

which is the unit of the adjoint resolution of Γ by $\textsc{Prop}[x]$. The right adjoint functor $\mathsf{G}_x : \textsc{Prop} \longrightarrow \textsc{Prop}[x]$ is a surjective embedding, resulting in the natural isomorphism:

$$\frac{\textsc{Prop}[x]\,(\mathsf{A} \to \mathsf{B})}{\textsc{Prop}\,(\Gamma\,, \mathsf{A} \to \mathsf{B})}$$

This functor preserves the the interpretations of $\wedge$ and $\top$ because they are limits, and preserves the interpretations of the other propositional connectives as well:

$$\begin{array}{ll} & \textsc{Prop}[x]\,(\mathsf{A}\,, \mathsf{B} \to \mathsf{C}) \\ \cong & \textsc{Prop}\,(\Gamma\,, \mathsf{A}\,, \mathsf{B} \to \mathsf{C}) \\ \cong & \textsc{Prop}\,(\Gamma\,, \mathsf{A} \to \mathsf{B} \supset \mathsf{C}) \\ \cong & \textsc{Prop}[x]\,(\mathsf{A} \to \mathsf{B} \supset \mathsf{C}) \end{array} \qquad \begin{array}{ll} & \textsc{Prop}[x]\,(\mathsf{A} \to \mathsf{C}) \times \textsc{Prop}[x]\,(\mathsf{B} \to \mathsf{C}) \\ \cong & \textsc{Prop}\,(\Gamma\,, \mathsf{A} \to \mathsf{C}) \times \textsc{Prop}\,(\Gamma\,, \mathsf{B} \to \mathsf{C}) \\ \cong & \textsc{Prop}\,((\Gamma \wedge \mathsf{A}) \vee (\Gamma \wedge \mathsf{B}) \to \mathsf{C}) \\ \cong & \textsc{Prop}\,(\Gamma\,, \mathsf{A} \vee \mathsf{B} \to \mathsf{C}) \\ \cong & \textsc{Prop}[x]\,(\mathsf{A} \vee \mathsf{B} \to \mathsf{C}) \end{array} \qquad \begin{array}{ll} & \textsc{Prop}[x]\,(0 \to \mathsf{A}) \\ \cong & \textsc{Prop}\,(\Gamma\,, 0 \to \mathsf{A}) \\ \cong & \textsc{Prop}\,(0\,, \Gamma \to \mathsf{A}) \\ \cong & \textsc{Prop}\,(0 \to \Gamma \supset \mathsf{A}) \\ = & \{\mathsf{i}_{\Gamma \supset \mathsf{A}}\} \end{array}$$

Moving to the polynomial category $\textsc{Prop}[x]$ effectively turns a context Γ into an axiom of the logical system and turns derivations with Γ as global context into proofs.

Chapter 7

Categorical Sequent Calculus

We continue the examination of intuitionistic first-order proof theory from a categorical perspective based on the adjunctions governing the connectives. In this chapter we interpret a system of sequent calculus in hyperdoctrines, much as we did with natural deduction in chapter 5. Having a categorical description of sequent calculus will allow us to apply search strategies for constructing sequent proofs in the categorical setting of a hyperdoctrine to search the free interpretation of a theory for a derivation of a goal. As in the case of natural deduction, the adjoint-theoretic descriptions of the quantifiers will allow us to decompose their non-invertible sequent rules into a purely logical part and a substitution. Based on this decomposition, we present a formulation of sequent calculus in which derivations are indexed by typing contexts and where these typing contexts and reindexings between them have first-class status. This permits the interpretation of the logic variables and generic parameters found in logic programming and allows us to defer the choice of quantifier instantiations during proof search.

An issue that arises in choosing a system of sequent calculus to which to try to give a categorical interpretation is exemplified by the engineer's old gripe about technical standards: the nice thing about them is that there are so many to choose from. Numerous sequent calculi have been developed, even just for intuitionistic first-order logic, each designed to have some particular desirable property; for example, close relationship to natural deduction, freedom from certain structural rules, freedom from certain adjacent inference permutations, and so on. As the sophistication of such a calculus increases, it becomes more difficult to relate it back to natural deduction, and thus to show that it captures the familiar notion of intuitionistic inference, or to give it constructive meaning through a calculus of realizers. Here we will investigate a particularly simple system of intuitionistic first-order sequent calculus, close to Gentzen's original system, "NJ". In chapter 8 we will impose further constraints on the system through the additional layer of a search strategy, which limits where and how the inference rules may be applied.

7.1 Intuitionistic Sequent Calculus

An intuitionistic logical **sequent** is an expression of the form, "$\Gamma \implies \mathsf{A}$", where A is a logical proposition called the "goal" or **succedent**, and Γ is a collection of logical propositions called the "context", "program", or **antecedent**. Sequent calculus was originally developed by Gentzen as a meta-calculus for the theory of natural deduction. In this regard, the symbol "$\Longrightarrow$" can be understood to represent inference. Thus the sequent $\Gamma \implies \mathsf{A}$ expresses the judgement that A is a logical consequence of Γ, and a sequent calculus proof of such a sequent represents a natural deduction derivation witnessing the claim. As described in chapter 2, for typed logics a context consists of a *propositional context*, containing logical assumptions, as well as a *typing context*, containing the typed free term variables in scope. We represent this with the notation, "$\Phi \mid \Gamma \implies \mathsf{A}$", where Φ is the typing context.

The inference rules of **sequent calculus** are like those of natural deduction in that they may have multiple (including possibly zero) *premises*, but always a single *conclusion*. However, the premises and conclusion are now sequents rather than propositions.[1] These rules come in two sorts, structural and logical, which determine the properties of logical contexts and connectives, respectively. These rules are depicted in figure 7.1.

If we consider contexts to be multisets, that is, unordered collections with multiplicity, then we have four **structural rules**:

contraction ($c\mathsf{L}$) , which says that the (non-zero) multiplicity of assumptions is irrelevant,

weakening ($w\mathsf{L}$) , which says that an inference remains valid under added assumptions,

cut (*cut*) , which describes how inferences may be composed,

initial sequent axiom (*init*) , which says that we may infer that which we assume.

If contexts are instead taken to be sets, then contraction of course becomes superfluous. In the case of sequent *proofs*, weakening and cut are admissible. For weakening this is quite obvious since we may push weakenings up to the leaves of a derivation, where they are absorbed by the axioms. The admissibility of cut is much more subtle, its proof constituting Gentzen's celebrated *Hauptsatz* [Gen35]. However, for sequent derivations with assumptions, weakening and cut must be retained. A sequent that contains its succedent within its antecedent is called an **initial sequent**.

When read from premises to conclusion, the **logical rules** of sequent calculus are all introduction rules, which act either on the right (succedent) or on the left (antecedent) of a sequent. Each logical rule acts on a single proposition in its conclusion, called the

[1] There remain also *typing judgement* premises in quantifier rules.

Structural Rules:

$$\frac{\Gamma, A, A \Rightarrow B}{\Gamma, A \Rightarrow B}\ cL \qquad \frac{\Gamma \Rightarrow B}{\Gamma, A \Rightarrow B}\ wL$$

$$\frac{\Gamma \Rightarrow A \qquad \Gamma, A \Rightarrow B}{\Gamma \Rightarrow B}\ cut \qquad \frac{}{\Gamma, A \Rightarrow A}\ init$$

Logical Rules:

	right	left
	$\frac{}{\Gamma \Rightarrow \top}\ \top R$	no rule for $\top L$
	no rule for $\bot R$	$\frac{}{\Gamma, \bot \Rightarrow A}\ \bot L$
	$\frac{\Gamma \Rightarrow A \qquad \Gamma \Rightarrow B}{\Gamma \Rightarrow A \wedge B}\ \wedge R$	$\frac{\Gamma, A, B \Rightarrow C}{\Gamma, A \wedge B \Rightarrow C}\ \wedge L$
	$\frac{\Gamma \Rightarrow A}{\Gamma \Rightarrow A \vee B}\ \vee R_1 \qquad \frac{\Gamma \Rightarrow B}{\Gamma \Rightarrow A \vee B}\ \vee R_2$	$\frac{\Gamma, A \Rightarrow C \qquad \Gamma, B \Rightarrow C}{\Gamma, A \vee B \Rightarrow C}\ \vee L$
	$\frac{\Gamma, A \Rightarrow B}{\Gamma \Rightarrow A \supset B}\ \supset R$	$\frac{\Gamma \Rightarrow A \qquad \Gamma, B \Rightarrow C}{\Gamma, A \supset B \Rightarrow C}\ \supset L$
	$\frac{\Gamma \Rightarrow A[x \mapsto e]}{\Gamma \Rightarrow \forall x : X \,.\, A}\ \forall R^{\dagger}$	$\frac{t : X \qquad \Gamma, A[x \mapsto t] \Rightarrow B}{\Gamma, \forall x : X \,.\, A \Rightarrow B}\ \forall L$
	$\frac{t : X \qquad \Gamma \Rightarrow A[x \mapsto t]}{\Gamma \Rightarrow \exists x : X \,.\, A}\ \exists R$	$\frac{\Gamma, A[x \mapsto e] \Rightarrow B}{\Gamma, \exists x : X \,.\, A \Rightarrow B}\ \exists L^{\dagger}$

† e not free in the conclusion

Figure 7.1: Inference rules for intuitionistic first-order sequent calculus

principal formula, and one or more propositions in each premise, called **active formulas**. The remaining propositions that occur in a rule are **passive formulas**, and are parametric. Like in natural deduction, the term "e" occurring in the quantifier rules $\forall\mathsf{R}$ and $\exists\mathsf{L}$ is an *eigenvariable*, and the term "t" in $\forall\mathsf{L}$ and $\exists\mathsf{R}$ is a *representative*, respectively, *witness* of its type.

There is a close relationship between the system of sequent calculus presented here and that of natural deduction in figure 2.1. The correspondence is given the **Prawitz translation**, which describes a function from cut-free sequent proofs to normal natural deduction derivations [Pra65]. This function is surjective (though not injective) and has been extended to sequent proofs with cuts, where cut elimination in sequent calculus is related to normalization in natural deduction [Zuc74; Pot77].

The basic idea of the translation is as follows. A sequent calculus logical right rule corresponds to appending an instance of the corresponding natural deduction introduction rule at the root of a derivation subtree. The sequent calculus logical left rules are more nuanced. Left rules for most right connectives correspond to appending instances of the corresponding natural deduction elimination rules at the leaves of a natural deduction subtree. Left rules for left connectives correspond to appending instances of the corresponding natural deduction elimination rules at the root of a natural deduction subtree; this is due to the single-conclusion nature of the inference rules. The left rule for implication corresponds to taking the derivation corresponding to an instance of $\supset -$ and both precomposing it with a derivation from the context and postcomposing it with a derivation to the goal. This has the effect of splitting a natural deduction subtree into two sequential parts. For a more detailed treatment of the translation, we refer the reader to [Gal93], [TS00], or the original [Pra65].

7.2 Sequent Calculus in Hyperdoctrines

Rather than detailing the interpretation of sequent calculus in natural deduction via the Prawitz translation, we present an interpretation of sequent calculus in the categorical semantics of hyperdoctrines. This allows us to see a sequent calculus proof as instructions for discovering – or constructing, depending on your point of view – an arrow of a given hom set in the fibers. Recall that propositions and propositional contexts are interpreted by objects in the fibers of a hyperdoctrine over the interpretation of their typing contexts.

Definition 7.2.1 (interpretation of a sequent) The **interpretation of a sequent**, $\Phi \mid \Gamma \Longrightarrow \mathsf{A}$, in a hyperdoctrine P is the hom set $\mathsf{P}(\llbracket\Phi\rrbracket)\,(\llbracket\Gamma\rrbracket \to \llbracket\mathsf{A}\rrbracket)$.

The interpretations of the structural rules follow from the interpretation of contexts as finite products.

Definition 7.2.2 (interpretation of the structural rules)

- The structural rule of *contraction* is interpreted as the precomposition of a diagonal arrow in context:

$$[\![\Gamma\,,\,A]\!] \xrightarrow{\mathrm{id}\times\Delta} [\![\Gamma\,,\,A\,,\,A]\!] \xrightarrow{[\![\mathcal{D}]\!]} [\![B]\!]$$

- The structural rule of *weakening* is interpreted as the precomposition of a complement-projection:

$$[\![\Gamma\,,\,A]\!] \xrightarrow{\pi} [\![\Gamma]\!] \xrightarrow{[\![\mathcal{D}]\!]} [\![B]\!]$$

- The structural rule of *cut* is interpreted as the composition:

$$[\![\Gamma]\!] \xrightarrow{\langle \mathrm{id}\,,\,[\![\mathcal{D}_1]\!]\rangle} [\![\Gamma\,,\,A]\!] \xrightarrow{[\![\mathcal{D}_2]\!]} [\![B]\!]$$

- The *initial sequent axiom* is interpreted as the projection:

$$[\![\Gamma\,,\,A]\!] \xrightarrow{\pi} [\![A]\!]$$

Notice that in the *context comonad*, these are simply the arrows Δ and !, and the Kleisli composition and identity, respectively. Because propositional contexts are considered to be unordered, we will leave tacit the associating and commuting isomorphisms on their interpretations.

We now turn our attention to the logical rules of sequent calculus. We can use the adjunctions interpreting the connectives to interpret these rules in a manner similar to the case for the inference rules of natural deduction described in theorem 5.0.1. As we did there, we will assume that the interpretations are free and suppress the interpretation brackets for readability; though again, the result holds in any hyperdoctrine interpretation.

- The right rules of right connectives and left rules of left connectives correspond exactly to their introduction, respectively, elimination rules in natural deduction. For right connectives, they are interpreted as the adjoint complement operation $-^\flat$, and for left connectives, as the adjoint complement operation $-^\sharp$ precomposed with a context distributing isomorphism. We call these rules **eigenrules**.
- The right rules of left connectives are interpreted as the postcomposition of the unit component of the corresponding adjunction. Dually, we obtain moral left rules of right connectives as the precomposition of the counit component "in context" – that

is, using its context comonad image in order to propagate the ambient propositional context. However, we will see that proof-theoretic considerations in the cases of conjunction and implication lead to deviation from this simple symmetry. We call these rules **anderrules**. Just as in natural deduction, the adjoint-theoretic anderrules for quantifiers decompose their Gentzen counterparts into a purely logical part and a substitution.

We now explain this in the case of each connective.

Conjunction

The adjunction for conjunction is summarized in diagram 5.1. It has the bijection of hom sets:

$$\frac{\textsc{Prop} \times \textsc{Prop}\,(\Delta\Gamma \to (A\,,B))}{\textsc{Prop}\,(\Gamma \to \wedge(A\,,B))}\ {}^{\flat}_{-}$$

Read as an inference rule of sequent calculus, this becomes:

$$\frac{(\Gamma\,,\Gamma)\ \Longrightarrow\ (A\,,B)}{\Gamma\ \Longrightarrow\ A \wedge B}\ \wedge R^{\star}$$

Using the fact that an arrow in a product of categories is the pair consisting of an arrow in each, we may write each sequent of the pair in the premise separately to get the given rule.

The counit of this adjunction is the ordered pair of projections, $(fst\,, snd)$. Given any derivations $\mathcal{D}_1 : \textsc{Prop}\,(\Gamma_1\,, A \to C)$ and $\mathcal{D}_2 : \textsc{Prop}\,(\Gamma_2\,, B \to D)$, we may obtain a derivation in the hom set $\textsc{Prop} \times \textsc{Prop}\,((\Gamma_1\,,\Gamma_2)\,,\Delta(A \wedge B) \to (C\,,D))$ by precomposing the counit component in context as shown:

$$\begin{array}{lccc}
\textsc{Prop} \times \textsc{Prop}: & \Gamma_1\,, A \wedge B & \Gamma_2\,, A \wedge B & \Big\}(\Gamma_1\,,\Gamma_2)\,,\varepsilon \\
& \Big\downarrow{\scriptstyle \mathrm{id}\,, fst} & \Big\downarrow{\scriptstyle \mathrm{id}\,, snd} & \\
& \Gamma_1\,, A & \Gamma_2\,, B & \\
& \Big\downarrow{\scriptstyle \mathcal{D}_1} & \Big\downarrow{\scriptstyle \mathcal{D}_2} & \\
& C & D &
\end{array}$$

where $(\Gamma_1\,,\Gamma_2)\,,\varepsilon$ is the context comonad image of ε. This construction corresponds to the sequent rule:

$$\frac{(\Gamma_1\,,\Gamma_2)\,,(A\,,B)\ \Longrightarrow\ (C\,,D)}{(\Gamma_1\,,\Gamma_2)\,,(A \wedge B\,,A \wedge B)\ \Longrightarrow\ (C\,,D)}\ \wedge L^{\star}$$

By separating both pairs, this can be rewritten as the pair of rules:

$$\frac{\Gamma_1\,,A\ \Longrightarrow\ C \qquad \Gamma_2\,,B\ \Longrightarrow\ D}{\Gamma_1\,,A \wedge B\ \Longrightarrow\ C} \qquad \text{and} \qquad \frac{\Gamma_1\,,A\ \Longrightarrow\ C \qquad \Gamma_2\,,B\ \Longrightarrow\ D}{\Gamma_2\,,A \wedge B\ \Longrightarrow\ D}$$

Instantiating these at $\mathtt{D} \coloneqq \mathtt{B}$, respectively, $\mathtt{C} \coloneqq \mathtt{A}$, we obtain the following rules, $\wedge\mathrm{L}_1$ and $\wedge\mathrm{L}_2$:

$$\frac{\Gamma_1 , \mathtt{A} \implies \mathtt{C} \qquad \overline{\Gamma_2 , \mathtt{B} \implies \mathtt{B}}\ \mathit{init}}{\Gamma_1 , \mathtt{A} \wedge \mathtt{B} \implies \mathtt{C}} \qquad \text{and} \qquad \frac{\overline{\Gamma_1 , \mathtt{A} \implies \mathtt{A}}\ \mathit{init} \qquad \Gamma_2 , \mathtt{B} \implies \mathtt{D}}{\Gamma_2 , \mathtt{A} \wedge \mathtt{B} \implies \mathtt{D}}$$

This pair of rules was used for conjunction on the left in Gentzen's original system of sequent calculus, LJ, and has a pleasing symmetry with the $\vee\mathrm{R}$ rules to follow. However, due to the nature of contexts as collections of premises interpreted conjunctively, these two rules have a common weakening to the given single rule, which has the advantage of being invertible. The equivalence is given by:

$$\frac{\dfrac{\Gamma , \mathtt{A} \implies \mathtt{C}}{\Gamma , \mathtt{A} , \mathtt{B} \implies \mathtt{C}}\ w\mathrm{L}}{\Gamma , \mathtt{A} \wedge \mathtt{B} \implies \mathtt{C}}\ \wedge\mathrm{L} \qquad \frac{\dfrac{\Gamma , \mathtt{B} \implies \mathtt{C}}{\Gamma , \mathtt{A} , \mathtt{B} \implies \mathtt{C}}\ w\mathrm{L}}{\Gamma , \mathtt{A} \wedge \mathtt{B} \implies \mathtt{C}}\ \wedge\mathrm{L} \qquad \frac{\dfrac{\overline{\Gamma , \mathtt{A} \implies \mathtt{A}}\ \mathit{init}}{\Gamma , \mathtt{A} \wedge \mathtt{B} \implies \mathtt{A}}\ \wedge\mathrm{L}_1 \qquad \dfrac{\Gamma , \mathtt{A} , \mathtt{B} \implies \mathtt{C}}{\Gamma , \mathtt{A} , \mathtt{A} \wedge \mathtt{B} \implies \mathtt{C}}\ \wedge\mathrm{L}_2}{\Gamma , \mathtt{A} \wedge \mathtt{B} \implies \mathtt{C}}\ \mathit{cut}$$

Disjunction

The case for disjunction is nearly dual to that for conjunction, except as we have already seen in section 5.2, the presence of an ambient context requires the mediation of the *distributive law* (lemma 3.3.2). With this, we have the bijection of hom sets:

$$\frac{\textsc{Prop} \times \textsc{Prop}\,(\Delta\Gamma , (\mathtt{A} , \mathtt{B}) \to \Delta\mathtt{C})}{\textsc{Prop}\,(\Gamma , \vee(\mathtt{A} , \mathtt{B}) \to \mathtt{C})}$$

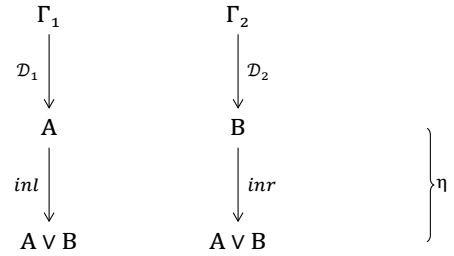
Read as an inference rule of sequent calculus, this becomes:

$$\frac{(\Gamma , \Gamma) , (\mathtt{A} , \mathtt{B}) \implies (\mathtt{C} , \mathtt{C})}{\Gamma , \mathtt{A} \vee \mathtt{B} \implies \mathtt{C}}\ \vee\mathrm{L}^{\star}$$

By separating the pair of sequents in the premise we obtain the given rule.

The unit of this adjunction is the ordered pair of coprojections, $(\mathit{inl} , \mathit{inr})$. Given any derivations $\mathcal{D}_1 : \textsc{Prop}\,(\Gamma_1 \to \mathtt{A})$ and $\mathcal{D}_2 : \textsc{Prop}\,(\Gamma_2 \to \mathtt{B})$, we may obtain a derivation in the hom set $\textsc{Prop} \times \textsc{Prop}\,((\Gamma_1 , \Gamma_2) \to \Delta(\mathtt{A} \vee \mathtt{B}))$ by postcomposing the unit component as shown:

$\textsc{Prop} \times \textsc{Prop}$:



This construction corresponds to the sequent rule:

$$\frac{(\Gamma_1 , \Gamma_2) \implies (A , B)}{(\Gamma_1 , \Gamma_2) \implies (A \vee B , A \vee B)} \vee R^\star$$

By separating both pairs, this can be rewritten as the pair of rules:

$$\frac{\Gamma_1 \implies A \qquad \Gamma_2 \implies B}{\Gamma_1 \implies A \vee B} \qquad \text{and} \qquad \frac{\Gamma_1 \implies A \qquad \Gamma_2 \implies B}{\Gamma_2 \implies A \vee B}$$

Instantiating at $\Gamma_2 \coloneqq B$, respectively $\Gamma_1 \coloneqq A$, we get the given rules $\vee R_1$ and $\vee R_2$:

$$\frac{\Gamma_1 \implies A \qquad \dfrac{}{B \implies B}\ \mathit{init}}{\Gamma_1 \implies A \vee B} \qquad \text{and} \qquad \frac{\dfrac{}{A \implies A}\ \mathit{init} \qquad \Gamma_2 \implies B}{\Gamma_2 \implies A \vee B}$$

Truth

The adjunction for truth is summarized in diagram 5.3. It has the bijection of hom sets:

$$\frac{\mathbb{1}(* \to *)}{\textsc{Prop}\,(\Gamma \to \top)}\ {}^\flat$$

Read as an inference rule of sequent calculus, this becomes:

$$\frac{* \implies *}{\Gamma \implies \top} \top R^\star$$

The premise of this rule is an initial sequent. Dispatching it yields the given rule.

Truth has an adjoint-theoretic moral left rule arising from composition with the counit as well. However, it resides in the category $\mathbb{1}$, where it allows the transformation of the identity sequent into itself – hardly much use.

Falsehood

The case for falsehood is dual to that for truth since as we have seen in section 5.4, context distribution is trivial in the category $\mathbb{1}$. Here we have the bijection of hom sets:

$$\frac{\mathbb{1}(* \to *)}{\textsc{Prop}\,(\Gamma , \bot \to A)}$$

Read as an inference rule of sequent calculus, this becomes:

$$\frac{* \implies *}{\Gamma , \bot \implies A} \bot L^\star$$

The premise is again an initial sequent. Dispatching it yields the given rule.

It is interesting to note that the rule for using $\bot$ has historically been presented as a right rule, representing the notion of proof by contradiction or "*reductio ad absurdum*",

$$\frac{\Gamma \implies \bot}{\Gamma \implies A} \bot R$$

even though this presentation doesn't fit the pattern of naming rules for the connective that is *introduced* into the *conclusion* of the rule, on the left or right, as the case may be. In any event, the two rules are equivalent:

$$\dfrac{\Gamma \Rightarrow \bot \qquad \dfrac{}{\Gamma, \bot \Rightarrow A}\ \bot L}{\Gamma \Rightarrow A}\ \mathit{cut} \qquad\qquad \dfrac{\dfrac{}{\Gamma, \bot \Rightarrow \bot}\ \mathit{init}}{\Gamma, \bot \Rightarrow A}\ \bot R$$

Falsehood has another, moral, right rule as well, but alas it is just as useless as truth's moral left rule.

Implication

The adjunction for implication is summarized in diagram 5.5. It has the bijection of hom sets:

$$\dfrac{\textsc{Prop}\,(\Gamma, A \to B)}{\textsc{Prop}\,(\Gamma \to A \supset B)}\ {}^{\flat}$$

Reading this as an inference of sequent calculus yields the given rule.

The counit of this adjunction is the evaluation map, eval. Given any derivation $\mathcal{D} : \Gamma, B \longrightarrow C$, we may obtain a derivation in $\Gamma, A \supset B, A \longrightarrow C$ by precomposing the counit component in context as shown:

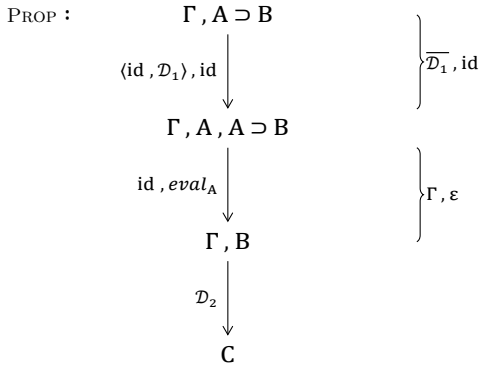
$$\begin{array}{lcl} \textsc{Prop}: & \Gamma, A \supset B, A & \Big\} \Gamma, \varepsilon \\ & \Big\downarrow{\scriptstyle \mathrm{id}, \mathit{eval}_A} & \\ & \Gamma, B & \\ & \Big\downarrow{\scriptstyle \mathcal{D}} & \\ & C & \end{array}$$

This corresponds to the sequent calculus inference rule:

$$\dfrac{\Gamma, B \Rightarrow C}{\Gamma, A \supset B, A \Rightarrow C}\ \supset L^{\star}$$

For meta-theoretic, perhaps even aesthetic, reasons, the logical rules of sequent calculus each act with respect to a single *principal formula* having the principal connective named in the rule. The presence of A in the antecedent of the conclusion of this rule violates this pattern. Instead, the given rule requires A to be derived from Γ, which we can describe

categorically (up to a tacit context permutation) as:



Despite this stylistic difference, the two rules are equivalent:

$$\dfrac{\dfrac{\Gamma \implies A}{\Gamma, A \supset B \implies A}\, wL \qquad \dfrac{\Gamma, B \implies C}{\Gamma, A \supset B, A \implies C}\, \supset L^{\star}}{\Gamma, A \supset B \implies C}\, cut \qquad \dfrac{\dfrac{}{\Gamma, A \implies A}\, init \qquad \dfrac{\Gamma, B \implies C}{\Gamma, A, B \implies C}\, wL}{\Gamma, A \supset B, A \implies C}\, \supset L$$

In spite of this equivalence the given rule for $\supset L$ warrants attention because it behaves like *cut* in that it essentially composes its two argument derivations sequentially. However, unlike *cut*, $\supset L$ composes them at a hom set rather than at an object. Indeed, by the inference rules for implication, *cut* is equivalent to the following identity rule for implication:

$$\dfrac{\Gamma, A \supset A \implies B}{\Gamma \implies B}\, \supset_{\mathrm{id}} L$$

which is therefore also admissible, and perhaps even more intuitively so than *cut*. We can interpret this rule as:

$$\Gamma \cong \Gamma, \top \xrightarrow{\mathrm{id}\,,\, \lambda_{\mathrm{id}_A}} \Gamma, A \supset A \xrightarrow{\quad \mathcal{D} \quad} B$$

The equivalence is given by,

$$\dfrac{\dfrac{\Gamma \implies A \qquad \Gamma, A \implies B}{\Gamma, A \supset A \implies B}\, \supset L}{\Gamma \implies B}\, \supset_{\mathrm{id}} L \qquad\qquad \dfrac{\dfrac{\dfrac{}{\Gamma, A \implies A}\, init}{\Gamma \implies A \supset A}\, \supset R \qquad \Gamma, A \supset A \implies B}{\Gamma \implies B}\, cut$$

Will see in the sequel that imposing constraints on the application of the $\supset L$ rule plays a crucial role in defining strategies for proof search.

Universal Quantification

The adjunction for universal quantification is summarized in diagram 5.6. It has the bijection of hom sets:

$$\dfrac{\mathrm{PROP}(\Phi\,,\, x : X)\,(\hat{x}^{*}\Gamma \to A)}{\mathrm{PROP}(\Phi)\,(\Gamma \to \forall x : X\,.\,A)}\; {}^{\flat}$$

Read as an inference rule of sequent calculus, this becomes:

$$\frac{\Phi\,,x:\mathrm{X}\ \mid\ \Gamma\ \Longrightarrow\ \mathrm{A}}{\Phi\ \mid\ \Gamma\ \Longrightarrow\ \forall x:\mathrm{X}\,.\,\mathrm{A}}\ \forall\mathrm{R}^{\star} \tag{7.1}$$

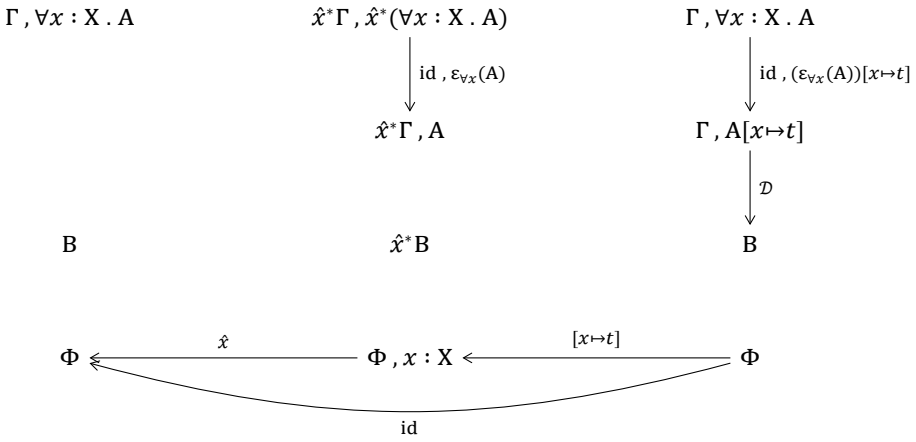
The explicit typing contexts convey exactly the same information as the side-condition of the given rule, with the context variable x acting as the eigenvariable.

For any universally quantified *proposition in context*, $\Phi \mid \forall x:\mathrm{X}\,.\,\mathrm{A}\ \textsc{prop}$, the component of the counit of the adjunction for $\forall$ is an arrow,

$$\varepsilon_{\forall x}(\mathrm{A}) \quad : \quad \textsc{Prop}(\Phi\,,x:\mathrm{X})\,(\hat{x}^{*}(\forall x:\mathrm{X}\,.\,\mathrm{A}) \to \mathrm{A})$$

As in the case of the natural deduction rule $\forall-$, we reindex the counit component in context by the single substitution determined by a term to form the sequent left rule.

Given any *term in context* $\Phi \mid t:\mathrm{X}$ and derivation $\mathcal{D} : \textsc{Prop}(\Phi)\,(\Gamma\,,\mathrm{A}[x{\mapsto}t] \to \mathrm{B})$, we may reindex the component of the counit in context by the single substitution $[x{\mapsto}t]$ and precompose the resulting arrow with $\mathcal{D}$ to obtain a derivation in $\textsc{Prop}(\Phi)\,(\Gamma\,,\forall x:\mathrm{X}\,.\,\mathrm{A} \to \mathrm{B})$ as shown:



(7.2)

This construction corresponds to the given left rule for universal quantification with an explicit typing context:

$$\frac{\Phi\ \Longrightarrow\ t:\mathrm{X} \qquad \Phi\ \mid\ \Gamma\ ,\ \mathrm{A}[x{\mapsto}t]\ \Longrightarrow\ \mathrm{B}}{\Phi\ \mid\ \Gamma\ ,\ \ \forall\, x:\mathrm{X}\,.\,\mathrm{A}\ \Longrightarrow\ \mathrm{B}}\ \forall\mathrm{L}^{\star}$$

The premise on the left represents the judgement $\Phi \mid t:\mathrm{X}$ in sequent notation. This premise is to be proved by a derivation in the type theory.

Existential Quantification

The case for existential quantification is nearly dual to that for universal quantification, except as we have already seen in section 5.7, the presence of an ambient propositional

context requires the mediation of *Frobenius reciprocity* (lemma 3.3.3). With this, we have the bijection of hom sets:

$$\frac{\textsc{Prop}(\Phi\,,x:\mathrm{X})\,(\hat{x}^*\Gamma\,,\mathrm{A}\to\hat{x}^*\mathrm{B})}{\textsc{Prop}(\Phi)\,(\Gamma\,,\exists x:\mathrm{X}\,.\,\mathrm{A}\to\mathrm{B})}$$

Read as an inference rule of sequent calculus, this becomes:

$$\frac{\Phi\,,x:\mathrm{X}\;\mid\;\Gamma\;,\;\mathrm{A}\;\Longrightarrow\;\mathrm{B}}{\Phi\;\mid\;\Gamma\;,\;\exists\,x:\mathrm{X}\,.\,\mathrm{A}\;\Longrightarrow\;\mathrm{B}}\;\exists\mathrm{L}^\star \tag{7.3}$$

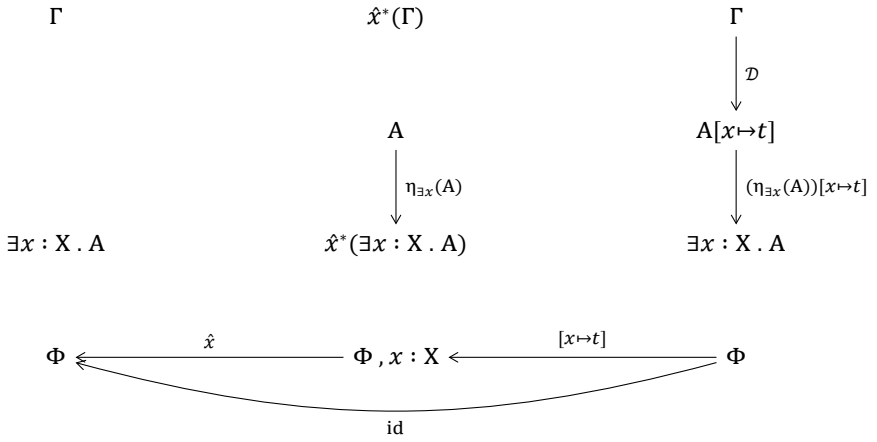
Again, the explicit typing contexts convey exactly the information of the side-condition of the given rule.

For any existentially quantified *proposition in context* $\Phi \mid \exists x:\mathrm{X}\,.\,\mathrm{A}$ PROP, the component of the unit of the adjunction for $\exists$ is an arrow,

$$\eta_{\exists x}(\mathrm{A}) \quad : \quad \textsc{Prop}(\Phi\,,x:\mathrm{X})\,(\mathrm{A}\to\hat{x}^*(\exists x:\mathrm{X}\,.\,\mathrm{A}))$$

As in the case of the natural deduction rule $\exists+$, we reindex the unit component by the single substitution determined by a term to form the sequent right rule.

Given any *term in context* $\Phi \mid t:\mathrm{X}$ and derivation $\mathcal{D}:\textsc{Prop}(\Phi)\,(\Gamma\to\mathrm{A}[x\mapsto t])$, we may reindex the component of the unit by the single substitution $[x\mapsto t]$ and postcompose the resulting arrow with $\mathcal{D}$ to obtain a derivation in $\textsc{Prop}(\Phi)\,(\Gamma\to\exists x:\mathrm{X}\,.\,\mathrm{A})$ as shown:



(7.4)

This construction corresponds to the given right rule for existential quantification with an explicit typing context:

$$\frac{\Phi\;\Longrightarrow\;t:\mathrm{X}\qquad\Phi\;\mid\;\Gamma\;\Longrightarrow\;\mathrm{A}[x\mapsto t]}{\Phi\;\mid\;\Gamma\;\Longrightarrow\;\exists x:\mathrm{X}\,.\,\mathrm{A}}\;\exists\mathrm{R}^\star$$

7.3 An Indexed Sequent Calculus

In contrast to natural deduction derivations, sequent calculus derivations may be constructed *unilaterally*, beginning from the root and working up to the leaves. This property

makes sequent systems well-suited to the task of proof search. Another property that makes a derivation system good for proof search is the **subformula property**, which demands that all formulas appearing in the premises of an inference rule be subformulas of those appearing in its conclusion.[2] The reason that this property is so desirable is the way in which it helps bound the search space: it ensures that all the formulas we must consider are sitting right in front of us, so that we never have to guess one out of thin air.

A quick scan of figure 7.1 reveals that all of the rules of cut-free sequent calculus satisfy the subformula property – except for $\forall\mathsf{L}$ and $\exists\mathsf{R}$, where the representative or witness term occurs only in the premises.[3] This raises the question of how, in general, we are to find such a term. The categorical perspective we have been exploring can shed some light on this matter.

The key insight is that in diagrams (7.2) and (7.4), the operation of reindexing by the single omission $\hat{x}$ and composing with the counit or unit component need not be immediately followed by the operation reindexing by the single substitution $[x{\mapsto}t]$ and composing with $\mathcal{D}$. Rather, we may be able to linger in the fiber $\textsc{Prop}(\Phi\,,x:\mathrm{X})$, continuing the derivation there until such time as it becomes clear which term to choose.

We illustrate this point with an instructive example. Suppose that we wish to prove the following sequent in a language with no function symbols:

$$\exists y:\mathrm{Y}\,.\,\forall u:\mathrm{X}\,.\,\mathrm{R}(u\,,y)\implies\forall x:\mathrm{X}\,.\,\exists v:\mathrm{Y}\,.\,\mathrm{R}(x\,,v)$$

Intuitively, this expresses the inference, "if there is something to which everything is related, then everything is related to something". The principal connectives of the succedent and antecedent are respectively right and left connectives, so we may fearlessly apply their respective invertible eigenrules, leading to an equivalent goal sequent:

$$x:\mathrm{X}\,,y:\mathrm{Y}\;\mid\;\forall u:\mathrm{X}\,.\,\mathrm{R}(u\,,y)\implies\exists v:\mathrm{Y}\,.\,\mathrm{R}(x\,,v)$$

Suppose that we next reindex by $\hat{v}$ and consider the unit component $\eta_{\exists v}$ there:

$$x:\mathrm{X}\,,y:\mathrm{Y}\,,v:\mathrm{Y}\;\mid\;\mathrm{R}(x\,,v)\implies\exists v:\mathrm{Y}\,.\,\mathrm{R}(x\,,v)$$

But before choosing a substitution for v, we first reindex by $\hat{u}$ and consider the counit component $\varepsilon_{\forall u}$ there:

$$x:\mathrm{X}\,,y:\mathrm{Y}\,,v:\mathrm{Y}\,,u:\mathrm{X}\;\mid\;\forall u:\mathrm{X}\,.\,\mathrm{R}(u\,,y)\implies\mathrm{R}(u\,,y)$$

[2] There is a closely-related notion of subformula property for natural deduction derivations where the global assumptions are considered as well [Pra65].

[3] Many authors define the subformula property in such a way as to allow these rules to satisfy it as well by considering any instance of a quantified proposition to be among its subformulas. This seems rather disingenuous to us since the term t cannot be said to occur in any meaningful way in the conclusions of these rules.

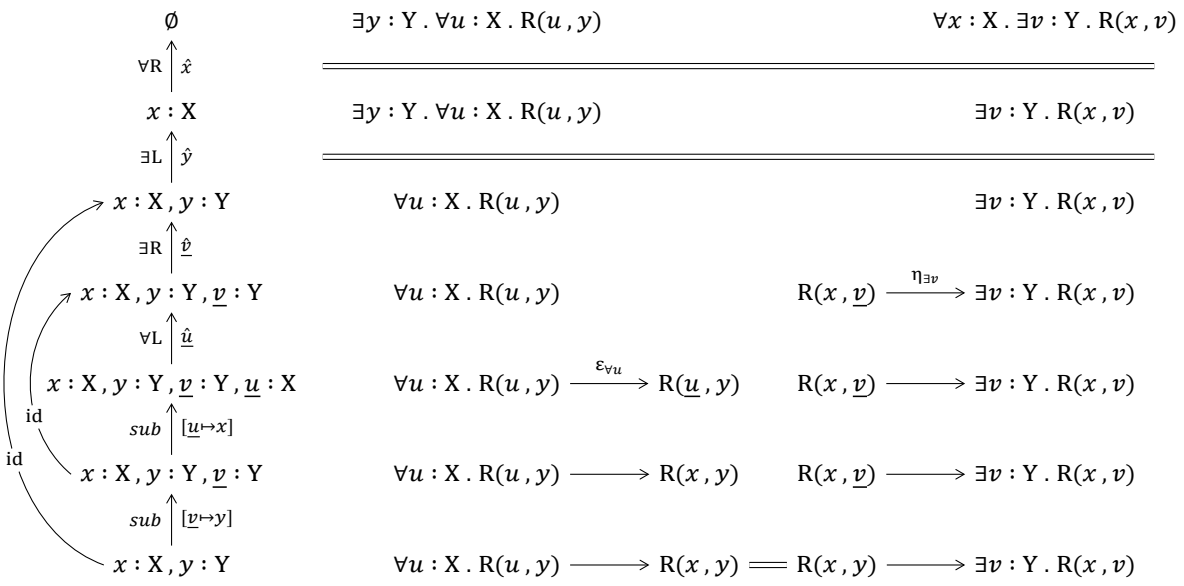


Figure 7.2: Categorical operational semantics of a derivation

We could then finish the proof if we could find substitutions for u and v in the context $x : \mathrm{X}, y : \mathrm{Y}$ that would unify $\mathrm{R}(u, y)$ with $\mathrm{R}(x, v)$. Of course $[u \mapsto x]$ and $[v \mapsto y]$ are the only possibilities. Applying them allows us to build the derivation that we seek. This categorical operational semantics is summarized in figure 7.2, where we have underlined the variables for which we need to provide substitutions, given rule name annotations in anticipation of the coming definitions and drawn the fibers horizontally in order to fit on the page.

This suggests that we may refine the quantifier anderrules so that they don't incorporate a choice of representative, respectively, witness. In diagrams (7.2) and (7.4), by simply translating composition with the respective counit in context or unit into a sequent calculus inference rule, we obtain:

$$\frac{\Phi, x : \mathrm{X} \mid \hat{x}^*\Gamma \ , \ \mathrm{A} \implies \hat{x}^*\mathrm{B}}{\Phi, x : \mathrm{X} \mid \hat{x}^*\Gamma \ , \ \hat{x}^*(\forall x : \mathrm{X} \,.\, \mathrm{A}) \implies \hat{x}^*\mathrm{B}} \quad \text{and} \quad \frac{\Phi, x : \mathrm{X} \mid \hat{x}^*\Gamma \implies \mathrm{A}}{\Phi, x : \mathrm{X} \mid \hat{x}^*\Gamma \implies \hat{x}^*(\exists x : \mathrm{X} \,.\, \mathrm{A})}$$

What we want is substitution instances of these rules, but we don't yet know *which* instances. Notice that in the conclusions of these rules, the choice of substituting term does not matter, since x is a dummy variable throughout. Regardless of term that is chosen, the conclusions will be respectively,

$$\Phi \mid \Gamma \ , \ \forall x : \mathrm{X} \,.\, \mathrm{A} \implies \mathrm{B} \qquad \text{and} \qquad \Phi \mid \Gamma \implies \exists x : \mathrm{X} \,.\, \mathrm{A}$$

Therefore, we reformulate the rules to use these conclusions, but in order to do so we must add a premise witnessing the existence of some term of type X in context Φ. We also adopt the versions of the quantifier eigenrules with explicit contexts given in (7.1) and (7.3).

Definition 7.3.1 (Indexed sequent quantifier rules)

$$\frac{\Phi, x : \mathrm{X} \mid \Gamma \Longrightarrow \mathrm{A}}{\Phi \mid \Gamma \Longrightarrow \forall x : \mathrm{X} \,.\, \mathrm{A}}\ \forall\mathrm{R} \qquad \frac{\Phi \Longrightarrow \underline{x} : \mathrm{X} \quad \Phi, \underline{x} : \mathrm{X} \mid \Gamma\ ,\ \mathrm{A}[x \mapsto \underline{x}] \Longrightarrow \mathrm{B}}{\Phi \mid \Gamma\ ,\ \forall x : \mathrm{X} \,.\, \mathrm{A} \Longrightarrow \mathrm{B}}\ \forall\mathrm{L}$$

$$\frac{\Phi, x : \mathrm{X} \mid \Gamma\ ,\ \mathrm{A} \Longrightarrow \mathrm{B}}{\Phi \mid \Gamma\ ,\ \exists x : \mathrm{X} \,.\, \mathrm{A} \Longrightarrow \mathrm{B}}\ \exists\mathrm{L} \qquad \frac{\Phi \Longrightarrow \underline{x} : \mathrm{X} \quad \Phi, \underline{x} : \mathrm{X} \mid \Gamma \Longrightarrow \mathrm{A}[x \mapsto \underline{x}]}{\Phi \mid \Gamma \Longrightarrow \exists x : \mathrm{X} \,.\, \mathrm{A}}\ \exists\mathrm{R}$$

The premise $\Phi \Longrightarrow \underline{x} : \mathrm{X}$ of the quantifier anderrules will itself be the conclusion of a derivation in the type theory. On its own, this premise would be unprovable because, since $\Phi, \underline{x} : \mathrm{X}$ constitutes a valid context, we know that $\underline{x} \notin \Phi$. This is why we apply the underlining annotation to the variable "$\underline{x}$": to remind ourselves that we are obliged to eventually provide a substitution for it by a term in context Φ in order to obtain a derivation in the context Φ. For this reason we will call context variables arising from quantifier anderrules **andervariables** or **obligation variables**. In the sequel we will see that they interpret the *logic variables* of logic programming.

It is important to understand that in the categorical semantics there is no inherent difference between *generic variables* and *obligation variables*; both are interpreted as (typing) context variables. The only distinction is in *how* such a variable comes to be introduced into a local scope. If this introduction is the result of a quantifier *eigenrule* then the variable should be thought of as generic and should not be substituted for, precisely because the eigenrule represents a bijection of derivation arrows between its conclusion and premise sequents. In contrast, if the introduction is the result of a quantifier *anderrule*, then the variable should be thought of as an obligation variable, because in order to obtain a derivation arrow residing in the original fiber a substitution must eventually be provided for it. In other words, the different treatment of eigenvariables and andervariables arises not from *what they are*, but rather from what they tell us about *where we are* (i.e. in which fiber) in relation to where we want to be.

This brings us to the question of how to represent substitutions in derivations. In the most direct correspondence to the indexed categorical semantics, we could simply take a derivation and rewrite it by applying the substitution at each node of the tree. However, this would be intolerably verbose as it would require copying the tree each time a substitution is performed. Instead, we will present a more compact notation. We will represent substitutions using a pair of inference rules, one for the type theory and one for the logic:

Definition 7.3.2 (indexed sequent substitution rules)

$$\frac{\Phi \implies s[\underline{x}\mapsto t] : \mathrm{Y}}{\Phi \implies s : \mathrm{Y}}\ sub\ [\underline{x}\mapsto t] \qquad \frac{\Phi \mid \Gamma[\underline{x}\mapsto t] \implies \mathrm{A}[\underline{x}\mapsto t]}{\Phi, \underline{x} : \mathrm{X} \mid \Gamma \implies \mathrm{A}}\ sub\ [\underline{x}\mapsto t]$$

We usually suppress the word "sub" in order to save space. These rules may at first seem counterintuitive because they are contravariant to the normal direction of substitution. This is because they represent the act of reindexing the entire derivation below them by the given substitution. In order to reindex a derivation by a substitution, the same substitution must be applied to the entire frontier. As explained in the preceding discussion, substitutions should be made only for obligation variables, which we indicate with the underlining annotation. Without this restriction, substitution would allow the inference of a sequent from one of its instances, which would be unsound.

The indexed quantifier anderrules and substitution rules permit the decomposition of the rules $\forall\mathrm{L}$ and $\exists\mathrm{R}$ in table 7.1 in the manner suggested by the categorical semantics depicted in diagrams (7.2) and (7.4). Intuitively, the creation of an obligation variable by an indexed quantifier anderrule represents deferring the selection of a term instance, while the eventual elimination of that obligation variable by substitution represents making the deferred choice.

Definition 7.3.3 (indexed sequent calculus) We will call the sequent system with the quantifier rules of definition 7.3.1 and substitution rules of definition 7.3.2 the **indexed sequent calculus**. The inference rules for the propositional connectives do not involve the typing context, and remain the same as in table 7.1.

Proposition 7.3.4 (equivalence of ordinary and indexed sequent calculus) The indexed sequent calculus ("i") is equivalent to the sequent calculus presented in table 7.1 – henceforth, "ordinary sequent calculus" ("o") – in the sense that the same sequents are provable in each.

Proof. We have already seen that the ordinary quantifier eigenrules are equivalent to their indexed counterparts, where the information carried in the side conditions of the ordinary rules corresponds precisely to that in the typing contexts of the indexed ones:

$$\begin{array}{rcl} \dfrac{\Gamma \implies \mathrm{A}[x\mapsto e]}{\Gamma \implies \forall x : \mathrm{X}\,.\,\mathrm{A}}\ \forall\mathrm{R}_o^{\dagger} & \Leftrightarrow & \dfrac{\Phi, x : \mathrm{X} \mid \Gamma \implies \mathrm{A}}{\Phi \mid \Gamma \implies \forall x : \mathrm{X}\,.\,\mathrm{A}}\ \forall\mathrm{R}_i \\[2ex] \dfrac{\Gamma\ ,\ \mathrm{A}[x\mapsto e] \implies \mathrm{B}}{\Gamma\ ,\ \exists\, x : \mathrm{X}\,.\,\mathrm{A} \implies \mathrm{B}}\ \exists\mathrm{L}_o^{\dagger} & \Leftrightarrow & \dfrac{\Phi, x : \mathrm{X} \mid \Gamma\ ,\ \mathrm{A} \implies \mathrm{B}}{\Phi \mid \Gamma\ ,\ \exists\, x : \mathrm{X}\,.\,\mathrm{A} \implies \mathrm{B}}\ \exists\mathrm{L}_i \end{array} \tag{7.5}$$

† e does not occur in the conclusion

It is also the case that the ordinary quantifier anderrules are equivalent to their indexed counterparts followed immediately by a reindexing:

$$\frac{t : \mathrm{X} \qquad \Gamma \ , \ \mathrm{A}[x \mapsto t] \implies \mathrm{B}}{\Gamma \ , \ \forall x : \mathrm{X} \,.\, \mathrm{A} \implies \mathrm{B}} \ \forall \mathrm{L}_o \quad \Leftrightarrow \quad \frac{\dfrac{\Phi \implies t : \mathrm{X}}{\Phi \implies \underline{x} : \mathrm{X}} \ [\underline{x} \mapsto t] \qquad \dfrac{\Phi \mid \Gamma \ , \ \mathrm{A}[x \mapsto t] \implies \mathrm{B}}{\Phi \, , \underline{x} : \mathrm{X} \mid \Gamma \ , \ \mathrm{A}[x \mapsto \underline{x}] \implies \mathrm{B}} \ [\underline{x} \mapsto t]}{\Phi \mid \Gamma \ , \ \forall x : \mathrm{X} \,.\, \mathrm{A} \implies \mathrm{B}} \ \forall \mathrm{L}_i$$

$$\frac{t : \mathrm{X} \qquad \Gamma \implies \mathrm{A}[x \mapsto t]}{\Gamma \implies \exists x : \mathrm{X} \,.\, \mathrm{A}} \ \exists \mathrm{R}_o \quad \Leftrightarrow \quad \frac{\dfrac{\Phi \implies t : \mathrm{X}}{\Phi \implies \underline{x} : \mathrm{X}} \ [\underline{x} \mapsto t] \qquad \dfrac{\Phi \mid \Gamma \implies \mathrm{A}[x \mapsto t]}{\Phi \, , \underline{x} : \mathrm{X} \mid \Gamma \implies \mathrm{A}[x \mapsto \underline{x}]} \ [\underline{x} \mapsto t]}{\Phi \mid \Gamma \implies \exists x : \mathrm{X} \,.\, \mathrm{A}} \ \exists \mathrm{R}_i \tag{7.6}$$

So an ordinary sequent derivation is equivalent to an indexed one in which quantifier anderrules are followed immediately by substitutions for the generated obligation variables.

Next, recall that reindexing commutes with the propositional connectives because it is a bicartesian closed functor, and that it commutes with the quantifiers by the Beck-Chevalley condition. Therefore, we can permute a substitution with any logical inference rule r,

$$\frac{\dfrac{\Phi' \mid \Gamma'[\underline{x} \mapsto t] \implies \mathrm{A}'[\underline{x} \mapsto t]}{\Phi' \, , \underline{x} : \mathrm{X} \mid \Gamma' \implies \mathrm{A}'} \ r}{\Phi \, , \underline{x} : \mathrm{X} \mid \Gamma \implies \mathrm{A}} \ [\underline{x} \mapsto t] \qquad \Leftrightarrow \qquad \frac{\dfrac{\Phi' \mid \Gamma'[\underline{x} \mapsto t] \implies \mathrm{A}'[\underline{x} \mapsto t]}{\Phi \mid \Gamma[\underline{x} \mapsto t] \implies \mathrm{A}[\underline{x} \mapsto t]} \ r}{\Phi \, , \underline{x} : \mathrm{X} \mid \Gamma \implies \mathrm{A}} \ [\underline{x} \mapsto t]$$

and similarly for rules with two premises, except of course for the quantifier anderrule instance that introduces the obligation variable of the substitution, because that variable does not occur in the context of its conclusion. Finally, we may permute two substitutions using the *substitution lemma* (lemma 4.3.5):

$$\frac{\dfrac{\Phi \mid \Gamma[\underline{y} \mapsto b][\underline{x} \mapsto a] \implies \mathrm{A}[\underline{y} \mapsto b][\underline{x} \mapsto a]}{\Phi \, , \underline{x} : \mathrm{X} \mid \Gamma[\underline{y} \mapsto b] \implies \mathrm{A}[\underline{y} \mapsto b]} \ [\underline{x} \mapsto a]}{\Phi \, , \underline{x} : \mathrm{X} \, , \underline{y} : \mathrm{Y} \mid \Gamma \implies \mathrm{A}} \ [\underline{y} \mapsto b] \qquad \Leftrightarrow \qquad \frac{\dfrac{\Phi \mid \Gamma[\underline{y} \mapsto b][\underline{x} \mapsto a] \implies \mathrm{A}[\underline{y} \mapsto b][\underline{x} \mapsto a]}{\Phi \, , \underline{y} : \mathrm{Y} \mid \Gamma[\underline{x} \mapsto a] \implies \mathrm{A}[\underline{x} \mapsto a]} \ [\underline{y} \mapsto b[\underline{x} \mapsto a]]}{\Phi \, , \underline{y} : \mathrm{Y} \, , \underline{x} : \mathrm{X} \mid \Gamma \implies \mathrm{A}} \ [\underline{x} \mapsto a]$$

By induction on derivations, we may permute a substitution down the logical branches of an indexed proof to the position immediately above the quantifier anderrule that introduces its obligation variable. Similarly, by the presumed functoriality of substitution in the type theory, we may permute a substitution down a type theoretic branch to the position immediately above the quantifier anderrule that introduces its obligation variable. In this way, we can contract the scope of an obligation variable to the single sequent lying between the quantifier anderrule instance that introduces it and the substitution that eliminates it. By equivalences (7.5) and (7.6), we thus obtain a derivation in the ordinary sequent system. □

Example 7.3.5 In the system of indexed sequent calculus, the proof in figure 7.2 is rep-

resented by the derivation,

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x:\mathrm{X},y:\mathrm{Y} \implies y:\mathrm{Y}}\,\mathit{init}}{x:\mathrm{X},y:\mathrm{Y} \implies \underline{v}:\mathrm{Y}}\,\sigma_1\cdot\sigma_2 \qquad \dfrac{\dfrac{\dfrac{}{x:\mathrm{X},y:\mathrm{Y},\underline{v}:\mathrm{Y} \implies x:\mathrm{X}}\,\mathit{init}}{x:\mathrm{X},y:\mathrm{Y},\underline{v}:\mathrm{Y} \implies \underline{u}:\mathrm{X}}\,\sigma_1 \qquad \dfrac{\dfrac{\dfrac{}{x:\mathrm{X},y:\mathrm{Y} \mid \mathrm{R}(x,y) \implies \mathrm{R}(x,y)}\,\mathit{init}}{x:\mathrm{X},y:\mathrm{Y},\underline{v}:\mathrm{Y} \mid \mathrm{R}(x,y) \implies \mathrm{R}(x,\underline{v})}\,\sigma_2 \coloneqq [\underline{v}\mapsto y]}{x:\mathrm{X},y:\mathrm{Y},\underline{v}:\mathrm{Y},\underline{u}:\mathrm{X} \mid \mathrm{R}(\underline{u},y) \implies \mathrm{R}(x,\underline{v})}\,\sigma_1 \coloneqq [\underline{u}\mapsto x]}{x:\mathrm{X},y:\mathrm{Y},\underline{v}:\mathrm{Y} \mid \forall u:\mathrm{X}.\,\mathrm{R}(u,y) \implies \mathrm{R}(x,\underline{v})}\,\forall\mathrm{L}}{x:\mathrm{X},y:\mathrm{Y} \mid \forall u:\mathrm{X}.\,\mathrm{R}(u,y) \implies \exists v:\mathrm{Y}.\,\mathrm{R}(x,v)}\,\exists\mathrm{R}}{x:\mathrm{X} \mid \exists y:\mathrm{Y}.\,\forall u:\mathrm{X}.\,\mathrm{R}(u,y) \implies \exists v:\mathrm{Y}.\,\mathrm{R}(x,v)}\,\exists\mathrm{L}}{\emptyset \mid \exists y:\mathrm{Y}.\,\forall u:\mathrm{X}.\,\mathrm{R}(u,y) \implies \forall x:\mathrm{X}.\,\exists v:\mathrm{Y}.\,\mathrm{R}(x,v)}\,\forall\mathrm{R}$$

Intuitively, this derivation should be read by beginning at the root and following the logical branch (on the right), the successful closure of which requires the given substitutions. The soundness of these substitutions is then verified by using them to close the type-theoretic branches. Recall that substitutions must be applied in the same order and in all branches to reindex a derivation. But once a branch contains no obligation variables in positions where substitutions could be made, further substitutions on that branch can have no effect, and may be suppressed (e.g. σ_2 in the center branch above). More generally, we may wish to suppress the writing of substitutions anywhere that they have no effect (e.g. σ_1 in the left branch above). We may also combine adjacent substitutions into a single simultaneous substitution, whose well-definedness is guaranteed by the substitution lemma (e.g. $\sigma \coloneqq \sigma_1 \cdot \sigma_2$).

It is instructive to see what goes wrong when trying to prove the converse sequent, which is in fact not a theorem.

Example 7.3.6

$$\dfrac{\dfrac{\dfrac{}{\emptyset \implies v:\mathrm{Y}}}{\emptyset \implies \underline{y}:\mathrm{Y}}\,\sigma_1\cdot\sigma_2 \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\underline{y}:\mathrm{Y},u:\mathrm{X} \implies u:\mathrm{X}}\,\mathit{init}}{\underline{y}:\mathrm{Y},u:\mathrm{X} \implies \underline{x}:\mathrm{X}}\,\sigma_1 \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{u:\mathrm{X},v:\mathrm{Y} \mid \mathrm{R}(u,v) \implies \mathrm{R}(u,v)}\,\mathit{init}}{\underline{y}:\mathrm{Y},u:\mathrm{X},v:\mathrm{Y} \mid \mathrm{R}(u,v) \implies \mathrm{R}(u,\underline{y})}\,\sigma_2 \coloneqq [\underline{y}\mapsto v]}{\underline{y}:\mathrm{Y},u:\mathrm{X},\underline{x}:\mathrm{X},v:\mathrm{Y} \mid \mathrm{R}(\underline{x},v) \implies \mathrm{R}(u,\underline{y})}\,\sigma_1 \coloneqq [\underline{x}\mapsto u]}{\underline{y}:\mathrm{Y},u:\mathrm{X},\underline{x}:\mathrm{X} \mid \exists v:\mathrm{Y}.\,\mathrm{R}(\underline{x},v) \implies \mathrm{R}(u,\underline{y})}\,\exists\mathrm{L}}{\underline{y}:\mathrm{Y},u:\mathrm{X} \mid \forall x:\mathrm{X}.\,\exists v:\mathrm{Y}.\,\mathrm{R}(x,v) \implies \mathrm{R}(u,\underline{y})}\,\forall\mathrm{L}}{\underline{y}:\mathrm{Y} \mid \forall x:\mathrm{X}.\,\exists v:\mathrm{Y}.\,\mathrm{R}(x,v) \implies \forall u:\mathrm{X}.\,\mathrm{R}(u,\underline{y})}\,\forall\mathrm{R}}{\emptyset \mid \forall x:\mathrm{X}.\,\exists v:\mathrm{Y}.\,\mathrm{R}(x,v) \implies \exists y:\mathrm{Y}.\,\forall u:\mathrm{X}.\,\mathrm{R}(u,y)}\,\exists\mathrm{R}$$

What goes wrong is that we cannot close the leftmost branch by proving $\emptyset \implies v:\mathrm{Y}$ in the type theory. Because v is not an obligation variable we can't substitute another term for it, and because $v \notin \emptyset$ we can't close the branch with an *init*, so we are stuck. In the categorical operational semantics, this corresponds to the fact that the substitution $[\underline{y}\mapsto v]$ cannot be performed prior to the application of the inference that brings the variable v into scope. Beginning the derivation with the other anderrule leads to a similar predicament, as the reader may wish to verify.

The verbosity of these derivations is in large part due to the bureaucracy of the typing contexts. These may be inferred up to an initial context of the end-sequent from the rest of the derivation. Therefore, we give an abbreviated syntax for the calculus in which the typing contexts are implicit.

Definition 7.3.7 (abbreviated syntax for indexed sequent calculus) The abbreviated syntax has:

propositional connective rules: as in table 7.1.

quantifier rules:

$$\frac{\Gamma \implies \mathrm{A}}{\Gamma \implies \forall x : \mathrm{X} \,.\, \mathrm{A}} \;\forall\mathrm{R}\;(x) \qquad \frac{\underline{x} : \mathrm{X} \qquad \Gamma \;,\; \mathrm{A}[x \mapsto \underline{x}] \implies \mathrm{B}}{\Gamma \;,\; \forall x : \mathrm{X} \,.\, \mathrm{A} \implies \mathrm{B}} \;\forall\mathrm{L}$$

$$\frac{\underline{x} : \mathrm{X} \qquad \Gamma \implies \mathrm{A}[x \mapsto \underline{x}]}{\Gamma \implies \exists x : \mathrm{X} \,.\, \mathrm{A}} \;\exists\mathrm{R} \qquad \frac{\Gamma \;,\; \mathrm{A} \implies \mathrm{B}}{\Gamma \;,\; \exists x : \mathrm{X} \,.\, \mathrm{A} \implies \mathrm{B}} \;\exists\mathrm{L}\;(x)$$

where in each case α-conversion is performed on the bound variable x so that it does not occur free in the conclusion.

substitution rules:

$$\frac{s[\underline{x} \mapsto t] : \mathrm{Y}}{s : \mathrm{Y}} \;sub\;[\underline{x} \mapsto t] \qquad \frac{\Gamma[\underline{x} \mapsto t] \implies \mathrm{A}[\underline{x} \mapsto t]}{\Gamma \implies \mathrm{A}} \;sub\;[\underline{x} \mapsto t]$$

where $\underline{x}$ is an obligation variable, that is, a variable not annotating a quantifier eigenrule instance or occurring free in the end-sequent.

eigenvariable axiom:

$$\frac{}{x : \mathrm{X}} \;eigen\;(x)$$

provided that $x : \mathrm{X}$ is an eigenvariable in the current context, that is, either a rule instance $\forall\mathrm{R}\;(x)$ or $\exists\mathrm{L}\;(x)$ occurs in the current branch and in that rule instance x has type X.

This version of the quantifier and substitution rules are just like their more verbose counterparts of definitions 7.3.1 and 7.3.2, except with an implicit typing context. The side conditions on the quantifier rules ensure that the implicit typing context remains valid. The variable annotating the quantifier eigenrules is to indicate that the context is being

augmented by the eigenvariable, and the new eigenvariable axiom corresponds to the axiom *init* in the type theory, which may be applied only if the eigenvariable actually occurs in the context. To save space, we will suppress the writing of "*sub*" and "*eigen*" in derivations. A valid abbreviated derivation may be elaborated back into verbose form, up to a parametric initial context, by reconstructing the explicit typing contexts.

To illustrate the use of the abbreviated syntax, we use it to repeat the derivations of examples 7.3.5 and 7.3.6.

Example 7.3.8

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{y : \mathrm{Y}}\ y}{\underline{v} : \mathrm{Y}}\ \sigma \qquad \dfrac{\dfrac{\dfrac{}{x : \mathrm{X}}\ x}{\underline{u} : \mathrm{X}}\ \sigma \qquad \dfrac{}{\mathrm{R}(x, y) \implies \mathrm{R}(x, y)}\ \mathit{init}}{\mathrm{R}(\underline{u}, y) \implies \mathrm{R}(x, \underline{v})}\ \sigma \coloneqq [\underline{u} \mapsto x, \underline{v} \mapsto y]}{\forall u : \mathrm{X} \,.\, \mathrm{R}(u, y) \implies \mathrm{R}(x, \underline{v})}\ \forall \mathrm{L}}{\forall u : \mathrm{X} \,.\, \mathrm{R}(u, y) \implies \exists v : \mathrm{Y} \,.\, \mathrm{R}(x, v)}\ \exists \mathrm{R}}{\exists y : \mathrm{Y} \,.\, \forall u : \mathrm{X} \,.\, \mathrm{R}(u, y) \implies \exists v : \mathrm{Y} \,.\, \mathrm{R}(x, v)}\ \exists \mathrm{L}\ (y)}{\exists y : \mathrm{Y} \,.\, \forall u : \mathrm{X} \,.\, \mathrm{R}(u, y) \implies \forall x : \mathrm{X} \,.\, \exists v : \mathrm{Y} \,.\, \mathrm{R}(x, v)}\ \forall \mathrm{R}\ (x)$$

Example 7.3.9

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{v : \mathrm{Y}}\ v}{\underline{y} : \mathrm{Y}}\ \sigma \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{u : \mathrm{X}}\ u}{\underline{x} : \mathrm{X}}\ \sigma \qquad \dfrac{}{\mathrm{R}(u, v) \implies \mathrm{R}(u, v)}\ \mathit{init}}{\mathrm{R}(\underline{x}, v) \implies \mathrm{R}(u, \underline{y})}\ \sigma \coloneqq [\underline{x} \mapsto u, \underline{y} \mapsto v]}{\exists v : \mathrm{Y} \,.\, \mathrm{R}(\underline{x}, v) \implies \mathrm{R}(u, \underline{y})}\ \exists \mathrm{L}\ (v)}{\forall x : \mathrm{X} \,.\, \exists v : \mathrm{Y} \,.\, \mathrm{R}(x, v) \implies \mathrm{R}(u, \underline{y})}\ \forall \mathrm{L}}{\forall x : \mathrm{X} \,.\, \exists v : \mathrm{Y} \,.\, \mathrm{R}(x, v) \implies \forall u : \mathrm{X} \,.\, \mathrm{R}(u, \underline{y})}\ \forall \mathrm{R}\ (u)}{\forall x : \mathrm{X} \,.\, \exists v : \mathrm{Y} \,.\, \mathrm{R}(x, v) \implies \exists y : \mathrm{Y} \,.\, \forall u : \mathrm{X} \,.\, \mathrm{R}(u, y)}\ \exists \mathrm{R}$$

Here again, the goal $v : \mathrm{Y}$ cannot be closed since the quantifier eigenrule instance that brings v into scope is not in that branch.

7.4 Indexed Sequent Tactics in Coq

As a demonstration of indexed sequent calculus, we have written a small tactic library for the Coq proof assistant [Coq12; BC04]. Coq is an automated proof verifier and certified program development environment based on the Calculus of (Co)Inductive Constructions (CIC) [CH88; CP90]. CIC is a constructive dependent type theory in the style of Martin-Löf [Mar84] with support for inductive and coinductive definitions. In Coq, every term has a type, which is in turn itself a term with a type. There is an infinite well-founded hierarchy of types. Inhabitation of a type by a term is decidable, as is convertibility of terms. Thus Coq may automatically compute whether a given term belongs to a given type, or whether two terms of a given type are equivalent. The convertibility relation comprises several subrelations, including β- and η-equivalence, as well as relations for unfolding definitions and

let-bindings, and for performing recursion on inductively defined objects. Coq is able to compute any of several normal forms of a term. In this way, Coq functions as a program development environment.

Under the *Curry-Howard correspondence*, where propositions are treated as types and their proofs as inhabitant terms, Coq can verify whether a purported proof indeed justifies a given proposition. To facilitate this process, Coq includes a *proof mode*, in which the user may interactively use high-level *tactics* to transform a goal proposition into a collection of (hopefully) simpler goals. Once the user has successfully proved a proposition by recursively solving all of the generated subgoals, Coq uses the tactic definitions to automatically transform the proof tree into a term of CIC inhabiting the type corresponding to the proved proposition. In this way, Coq functions as an automated proof verifier.

The ability to express strong specifications using dependent types and to include propositions and their proofs within programs allows for the construction of certified programs, those where some aspects of their behavior are guaranteed by the program itself. Further, the ability to write programs within the tactic language allows for the writing of decision procedures, or more generally, search strategies, to direct an automated search for proofs.

Coq's embedded tactic language, called "Ltac", is a functional programming language with facilities for non-linear pattern matching and backtracking. Coq tactics may also be written directly in Coq's implementation language of OCaml, though we did not pursue this approach. The high-level nature of the Ltac language permits only limited manipulation of the global derivation state. At any point in time, a derivation may contain multiple incomplete branches, each ending in an open goal. However, in the interface to Coq only one such goal may be active at a time, and any tactics applied generally affect only the active goal. This is problematic from the point of view of our indexed sequent calculus because the reindexing triggered by a substitution rule must be applied globally, that is, to all open goals.

Fortunately, Coq provides another feature that we may use to achieve the non-locality we seek. Coq's **existential variables** generalize the *logic variables* of Prolog. They are typed meta-variables that may be instantiated to any type-appropriate term, either by the system, as in Prolog, through unification in the course of proof search; or directly by the user, who may at any point manually assign them an instantiation. Unlike Prolog, Coq enforces the constraint that all existential variables must eventually become instantiated in the course of a proof. This is required for soundness, as example 5.7.1 illustrates. Coq's existential variables provide us a means of effecting non-locality across derivation branches. We use them to represent the obligation variables arising through the application of quantifier anderrules and Coq automatically keeps track of instantiations that occur across goals.

At the time of this writing, the status of existential variables in Coq is in a state of

flux. Their implementation began as a high-level wrapper, rather than as a part of Coq's proof engine itself. This greatly limited the facilities for manipulating them in the tactic language. Since Coq version 8.4 (released in 2012) this has changed, but the tactic language has not yet caught up with these changes.

As a result, our indexed tactic library remains unsatisfactory in some ways. Among them is that the creation of an existential variable by the application of a quantifier anderrule does not generate a goal corresponding to the type-theoretic premise of the indexed sequent rule. This presents no problem if all existential variable becomes instantiated as the result of substitutions. However, if all logical branches of a derivation are closed but there remain uninstantiated existential variables, then we must use a command from Coq's imperative command language, called "the Vernacular" in order to transform these uninstantiated existential variables into goals.

Another shortcoming is that dependencies in the local context involving existential variables are not automatically enforeced as is done for local assumptions (corresponding to generic context variables and local logical assumptions) and local definitions (let-aliases). However, we have largely overcome this shortcoming by aliasing the existential variables with local definitions and managing their context dependencies manually.

Despite these shortcomings, we believe that the indexed sequent tactics provide an interesting proof-of-concept, illustrating how the indexed categorical perspective can be practically applied in proof assistants.

The indexed sequent tactics for Coq correspond closely to the rules of indexed sequent calculus presented in definition 7.3.3. Their definitions may be found in appendix B. Here we illustrate their use by repeating example 7.3.5 in Coq by using these tactics.

Example 7.4.1

```
Lemma some_all__all_some (X Y : Type) (R : X → Y → Prop) :
  (∃ y : Y , ∀ u : X , R u y) → (∀ x : X , ∃ v : Y , R x v) .
Proof .
  ⊃R . ∀R . ∃L H . ∃R . ∀L H_y . sub u x . sub v y . init H_y_u .
Qed .
```

As can be seen, Coq supports a Unicode-based notation very close to that commonly used in mathematics. The statement of the lemma should be immediately comprehensible. The meaning of the proof, however, may not be so obvious. Coq proof scripts do not explicitly show the effects of tactic applications on the state of a derivation. In order to see these, one must step through the derivation within the Coq interactive environment. However, in this case it should be clear when comparing with the derivation tree in example

7.3.5 that this proof contains exactly the same tactics and in exactly the same order (when read upward from the root) as those in the logical (rightmost) subderivation of the indexed sequent proof, except for the initial ⊃R, which simply moves the hypothesis into the local context.

Aside from the already mentioned fact that the tactics for quantifier anderrules do not generate type-theoretic goals for their obligation variables, the only notable difference to the indexed sequent calculus as presented in definition 7.3.3 is that the ⊃L rule presents its two subgoals to the user in right-to-left order. The reason for this is to better fit with the *backward-chaining* approach used in logic programming, which we will encounter in the sequel. With backward-chaining, it is the rightmost branch of ⊃L that is developed first, potentially triggering reindexings that cause obligation variables to become instantiated. This aspect is, however, inessential since Coq provides commands allowing the user to select any open goal for activation.

Chapter 8

Proof Search Strategies

In this chapter we present several logical systems used in logic programming and show how the computation mechanism for each may be regarded as a complete search strategy in the indexed sequent calculus presented in section 7.3. By a **search strategy** we mean a not necessarily deterministic or complete procedure for choosing an inference to apply to the frontier of a derivation that is not yet a proof of its end-formula.

We begin with SLD-resolution for Horn logic, the system underlying the programming language Prolog. Next, we present uniform proof for hereditarily Harrop logic, the first-order fragment of the system underlying the programming language λ-Prolog. We show that uniform proof is in fact complete for a syntactically richer language, which we call the language of constructive sequents.

Finally we present the strategy of focused proof search adapted to intuitionistic first-order logic. We show that, as in the original formulation for linear logic, intuitionistic focusing is able to greatly reduce the search space of potential proofs without sacrificing completeness, a result we have not found in the literature. We close by comparing this focusing strategy with a sequent system from the literature having a certain very desirable property related to the input-output semantics of logic programming.

When searching for sequent *proofs*, we may dispense with the structural rules of *weakening* and *cut*. Following custom, we will consider contexts to be sets so that *contraction* is also redundant.[1] Thus, in the following the only structural rule that we need to consider is the *initial sequent axiom*.

[1] However, we will sometimes use explicit contractions in order to indicate that a particular member of the context is being selected for attention.

8.1 SLD-Resolution for Horn Logic

Horn logic is the logical system underlying the logic programming language Prolog. For a thorough account of Horn logic, Prolog, and its computation mechanism of SLD-resolution, we refer the reader to [Llo84] or [NM95], and provide here only a brief summary. Our purpose in this section is to show how SLD-resolution may be interpreted as a search strategy in our indexed sequent calculus.

In Horn logic a **literal** is either an atomic proposition (positive literal) or the negation of an atomic proposition (negative literal). A **disjunctive clause** (or just "clause") is the universal closure of a disjunction of literals:

$$\forall \overrightarrow{x : \mathrm{X}} \,.\, \underbrace{\mathrm{A}_1 \vee \cdots \vee \mathrm{A}_m}_{\text{positive literals}} \vee \underbrace{\neg \mathrm{B}_1 \vee \cdots \vee \neg \mathrm{B}_n}_{\text{negative literals}}$$

A **definite clause** is one with exactly one positive literal and a **negative clause** is one with no positive literals. A **Horn clause** is either a definite clause or a negative clause.

There is a semi-decision procedure called **SLD-resolution** for determining the unsatisfiability of a finite set of Horn clauses. That is, if such a set is unsatisfiable, then SLD-resolution will eventually discover this fact; however, if the set is satisfiable, the procedure may fail to terminate. One might reasonably wonder why we should care about determining the unsatisfiability of a set of Horn clauses. The answer involves the constructivity of SLD-resolution and the classical logic that is being assumed.

First, notice that a disjunctive clause is classically equivalent to the universal closure of an implication of a disjunction of atoms by a conjunction of atoms:

$$\begin{aligned} & \forall \overrightarrow{x : \mathrm{X}} \,.\, \mathrm{A}_1 \vee \cdots \vee \mathrm{A}_m \vee \neg \mathrm{B}_1 \vee \cdots \vee \neg \mathrm{B}_n \\ \equiv_c \quad & \forall \overrightarrow{x : \mathrm{X}} \,.\, (\mathrm{A}_1 \vee \cdots \vee \mathrm{A}_m) \vee \neg (\mathrm{B}_1 \wedge \cdots \wedge \mathrm{B}_n) \\ \equiv_c \quad & \forall \overrightarrow{x : \mathrm{X}} \,.\, (\mathrm{B}_1 \wedge \cdots \wedge \mathrm{B}_n) \supset (\mathrm{A}_1 \vee \cdots \vee \mathrm{A}_m) \end{aligned}$$

We will call a proposition in this form an **implicative clause**. The disjunction in positive position is called the **clause head** and the conjunction in negative position is the **clause tail**. Thus a definite disjunctive clause is classically equivalent to an implicative clause with an atomic head:

$$\forall \overrightarrow{x : \mathrm{X}} \,.\, (\mathrm{T}_1 \wedge \cdots \wedge \mathrm{T}_n) \supset \mathrm{H} \tag{8.1}$$

We will call a proposition of this form a **Horn program formula** or "Horn program clause". A finite set of program formulas is called a **program**.

Every Horn program is satisfiable, for example, by the **Herbrand base**, which is the model-theoretic interpretation consisting of all closed atomic propositions of the language.

The Herbrand base provides a canonical maximal interpretation for a Horn program. There exists a canonical minimal interpretation as well, called the **least Herbrand model**, which is essentially the inductive closure determined by a set of Horn program clauses.

So if a set of Horn clauses is to be unsatisfiable then it must contain at least one negative clause. A negative clause is classically equivalent to the negation of the existential closure of a conjunction of atoms:

$$\begin{array}{rl} & \forall \overrightarrow{x : \mathrm{X}} \,.\, \neg \mathrm{B}_1 \vee \cdots \vee \neg \mathrm{B}_n \\ \equiv_c & \forall \overrightarrow{x : \mathrm{X}} \,.\, \neg (\mathrm{B}_1 \wedge \cdots \wedge \mathrm{B}_n) \\ \equiv & \neg \exists \overrightarrow{x : \mathrm{X}} \,.\, \mathrm{B}_1 \wedge \cdots \wedge \mathrm{B}_n \end{array}$$

So the negation of a negative clause is classically equivalent to a proposition of the form:

$$\exists \overrightarrow{x : \mathrm{X}} \,.\, \mathrm{G}_1 \wedge \cdots \wedge \mathrm{G}_n \tag{8.2}$$

We will call such a proposition a **Horn goal formula** or "Horn goal".

Let Γ be a Horn program and G a Horn goal, then:

$$\begin{array}{rl} & \Gamma \ , \ \neg \mathrm{G} \quad \text{is unsatisfiable} \\ \Leftrightarrow & \bigwedge(\Gamma) \wedge \neg \mathrm{G} \quad \text{is unsatisfiable} \\ \Leftrightarrow & \neg(\bigwedge(\Gamma) \wedge \neg \mathrm{G}) \quad \text{is valid} \\ \Leftrightarrow & \neg \bigwedge(\Gamma) \vee \mathrm{G} \quad \text{is valid} \\ \Leftrightarrow & \bigwedge(\Gamma) \supset \mathrm{G} \quad \text{is valid} \\ \Leftrightarrow & \Gamma \vDash \mathrm{G} \end{array}$$

That is, a collection of Horn clauses consisting of a program together with a negative clause is unsatisfiable just in case the corresponding goal is a logical consequence of the program. This is a more intuitive and generalizable way to think about Horn logic.

By the completeness theorem for first-order logic, this implies that there is a proof of G from assumptions Γ. Note that in the form of Horn program and goal formulas, the propositions of Horn logic do not involve the connective $\neg$. This makes plausible a result proved in [Mil89a], that classical and intuitionistic derivability coincide for Horn logic (and indeed, that both coincide with derivability in minimal logic as well). So by thinking of Horn logic in terms of Horn program and goal formulas, we may remain within the system of intuitionistic logic that we have been studying.

Thus by deciding the unsatisfiability of a set of Horn clauses containing a single negative clause, SLD-resolution is also determining whether a goal is a logical consequence of a program. The reader may verify that adding additional negative clauses to the set corresponds

to determining whether the disjunction of the corresponding goals is a consequence of the program.

This is certainly useful, but what makes it even more useful, indeed what makes it useful *as a programming language*, is the fact that SLD-resolution does this constructively, in the sense that if a goal formula is a consequence of a program, then it is able to provide witnesses for the existentially quantified variables.

Although this is not the way it is usually presented, we can think of *SLD-resolution* as a simple search strategy in the indexed sequent calculus of section 7.3. In fact, it can be defined to be a single derived inference rule, applied repeatedly. For Γ a Horn program containing the clause $\forall \overrightarrow{x : \mathrm{X}} \,.\, \mathrm{T}_1 \wedge \cdots \wedge \mathrm{T}_n \supset \mathrm{H}$, it is a rule equivalent to:

$$\dfrac{\dfrac{}{y_1 : \mathrm{Y}_1}\,\sigma\ \cdots\ \dfrac{}{y_k : \mathrm{Y}_k}\,\sigma \qquad \dfrac{\dfrac{}{\Gamma \Rightarrow \mathrm{G}_1}\,\sigma \quad \cdots \quad \dfrac{\dfrac{\dfrac{}{x_1 : \mathrm{X}_1}\,\sigma\ \cdots\ \dfrac{}{x_j : \mathrm{X}_j}\,\sigma \qquad \dfrac{\dfrac{}{\Gamma \Rightarrow \mathrm{T}_1 \wedge \cdots \wedge \mathrm{T}_n}\,\sigma \qquad \dfrac{\dfrac{}{\sigma(\mathrm{H}) \Rightarrow \sigma(\mathrm{G}_i)}\,\mathit{init}}{\mathrm{H} \Rightarrow \mathrm{G}_i}\,\sigma}{\Gamma\ ,\ \mathrm{T}_1 \wedge \cdots \wedge \mathrm{T}_n \supset \mathrm{H} \Rightarrow \mathrm{G}_i}\,\supset\mathrm{L}}{\Gamma\ ,\ \forall \overrightarrow{x : \mathrm{X}} \,.\, \mathrm{T}_1 \wedge \cdots \wedge \mathrm{T}_n \supset \mathrm{H} \Rightarrow \mathrm{G}_i}\,\forall\mathrm{L}s}{\Gamma \Rightarrow \mathrm{G}_i}\,c\mathrm{L} \quad \cdots \quad \dfrac{}{\Gamma \Rightarrow \mathrm{G}_m}\,\sigma}{\Gamma \Rightarrow \mathrm{G}_1 \wedge \cdots \wedge \mathrm{G}_m}\,\wedge\mathrm{R}s}{\Gamma \Rightarrow \exists \overrightarrow{y : \mathrm{Y}} \,.\, \mathrm{G}_1 \wedge \cdots \wedge \mathrm{G}_m}\,\exists\mathrm{R}s$$

where a rule "$-s$" is shorthand for repeated application of the given rule until no longer applicable, and the context Γ is suppressed in the right branch of the rule $\supset\mathrm{L}$ since we commit ourselves to using only the clause head in this branch.[2]

This can in turn be seen as a composition of a right phase:

$$\dfrac{y_1 : \mathrm{Y}_1 \quad \cdots \quad y_k : \mathrm{Y}_k \qquad \dfrac{\Gamma \Rightarrow \mathrm{G}_1 \quad \cdots \quad \Gamma \Rightarrow \mathrm{G}_m}{\Gamma \Rightarrow \mathrm{G}_1 \wedge \cdots \wedge \mathrm{G}_m}\,\wedge\mathrm{R}s}{\Gamma \Rightarrow \exists \overrightarrow{y : \mathrm{Y}} \,.\, \mathrm{G}_1 \wedge \cdots \wedge \mathrm{G}_m}\,\exists\mathrm{R}s$$

a left phase:

$$\dfrac{\dfrac{x_1 : \mathrm{X}_1 \quad \cdots \quad x_j : \mathrm{X}_j \qquad \dfrac{\Gamma \Rightarrow \mathrm{T}_1 \wedge \cdots \wedge \mathrm{T}_n \qquad \mathrm{H} \Rightarrow \mathrm{G}_i}{\Gamma\ ,\ \mathrm{T}_1 \wedge \cdots \wedge \mathrm{T}_n \supset \mathrm{H} \Rightarrow \mathrm{G}_i}\,\supset\mathrm{L}}{\Gamma\ ,\ \forall \overrightarrow{x : \mathrm{X}} \,.\, \mathrm{T}_1 \wedge \cdots \wedge \mathrm{T}_n \supset \mathrm{H} \Rightarrow \mathrm{G}_i}\,\forall\mathrm{L}s}{\Gamma \Rightarrow \mathrm{G}_i}\,c\mathrm{L}$$

and a unification phase:

$$\dfrac{\dfrac{}{\sigma(\mathrm{H}) \Rightarrow \sigma(\mathrm{G}_i)}\,\mathit{init}}{\mathrm{H} \Rightarrow \mathrm{G}_i}\,\sigma$$

where σ is a substitution such that $\sigma(\mathrm{H}) = \sigma(\mathrm{G}_i)$. This substitution is applied to all branches of the derivation and the process is repeated. If $n = 0$, that is, if a program clause has an

[2] This can either be considered a notational shorthand, or may be achieved literally by repeated weakening.

empty tail, then we may replace $\mathtt{H}$ with the equivalent proposition $\top \supset \mathtt{H}$ for the sake of uniformity.[3] Note that the $\exists \mathtt{R}s$ step is non-trivial only once, at the the beginning of the derivation. Note also that this description is not deterministic.

In each iteration, the goal atom $\mathtt{G}_i$ to which the left phase is applied is called the **selected goal**. Likewise, the clause in Γ to which the contraction $c\mathtt{L}$ is applied is called the **selected program clause**. It turns out that the order in which goals are selected does not affect provability or the set of possible computed answers (to be discussed momentarily), a result known as the "independence of computation rule". However, this is not the case for choosing program clauses to apply. If a choice of program clause leads to a *failed derivation* then we must backtrack to that choice point and try a different choice. We may give up and declare failure globally only after trying every program clause in each step. But success and failure are not the only possible outcomes. Some choices may lead to the construction of infinite derivation trees, a source of potential non-termination.

The substitution σ is called a **unifier** of $\mathtt{H}$ and $\mathtt{G}_i$. First-order unification is decidable, that is, there are known algorithms (complete, terminating procedures) that return a most general unifier if a unifier exists and report failure otherwise. If a proof is found, then the composition of all substitutions performed in the course of the proof, $\sigma \coloneqq \sigma_1 \cdot \ldots \cdot \sigma_p$, is known as the **computed substitution** and the sequence of existential witnesses $(\sigma(y_1), \cdots, \sigma(y_k))$ is called the **computed answer**. There may be more than one SLD-proof of a Horn sequent, and different proofs my yield different computed answers. Implementations of Horn logic such as Prolog generally provide some facility for allowing the user to choose whether to accept a computed answer or to reject it and have the system continue searching for another solution.

This ability of SLD-resolution to compute witnesses for the existentially quantified variables in a goal is what gives Prolog its **input-output semantics**, which in turn is what makes it suitable as a programming language. SLD-resolution can be used to find satisfying substitutions for goals involving inductively-defined predicates. A stereotypical example is the following.

Example 8.1.1 We may inductively define addition of natural numbers with the two Horn program clauses:

$$\begin{aligned} \mathit{plus\_zero} &: \forall d : \mathbb{N} \,.\, \top \supset \mathrm{P}(z\,, d\,, d) \\ \mathit{plus\_succ} &: \forall a\,, b\,, c : \mathbb{N} \,.\, \mathrm{P}(a\,, b\,, c) \supset \mathrm{P}(sa\,, b\,, sc) \end{aligned}$$

[3] This is inessential; not doing so simply requires the addition of a case analysis on the form of the program clause. But we will soon see the advantage of having all program formulas in a common syntactic form. In general, the conversion of program formulas to a logically equivalent set in such a form is a process known as "program elaboration". Thus we may think of this substitution as Horn program elaboration, though it is so simple that it usually goes unnamed.

where $z : \mathbb{N}$ represents the constant zero and $s : \mathbb{N} \longrightarrow \mathbb{N}$ represents the successor function. If Γ is the Horn program $\{plus\_zero\ ,\ plus\_succ\}$, then we may effectively compute the subtraction $3 - 2$ by proving the Horn sequent:

$$\Gamma \implies \exists n : \mathbb{N}\,.\,\mathrm{P}(n\,,2\,,3)$$

where we write numeric constants in decimal rather than unary notation for convenience. An SLD-proof (in this case, the only one) is the following:

$$\begin{array}{lll} \sigma_1 & \coloneqq & [n \mapsto sa\,,b \mapsto 2\,,c \mapsto 2] \\ \sigma_2 & \coloneqq & [a \mapsto 0\,,d \mapsto 2] \end{array}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{1:\mathbb{N}}}{sa:\mathbb{N}}\,\sigma_2}{n:\mathbb{N}}\,\sigma_1 \quad \dfrac{\dfrac{\dfrac{\vdots}{0:\mathbb{N}}}{a:\mathbb{N}}\,\sigma_{1,2} \quad \dfrac{\dfrac{\vdots}{2:\mathbb{N}}}{b:\mathbb{N}}\,\sigma_1 \quad \dfrac{\dfrac{\vdots}{2:\mathbb{N}}}{c:\mathbb{N}}\,\sigma_1 \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{2:\mathbb{N}}}{d:\mathbb{N}}\,\sigma_2 \quad \dfrac{\dfrac{}{\Gamma \Rightarrow \top}\,\top\mathrm{R} \quad \dfrac{\dfrac{}{\mathrm{P}(0,2,2) \Rightarrow \mathrm{P}(0,2,2)}\,init}{\mathrm{P}(0,d,d) \Rightarrow \mathrm{P}(a,2,2)}\,\sigma_2}{\Gamma\ ,\ \top \supset \mathrm{P}(0,d,d) \Rightarrow \mathrm{P}(a,2,2)}\,\supset\!\mathrm{L}}{\Gamma\ ,\ \forall d:\mathbb{N}\,.\,\top \supset \mathrm{P}(0,d,d) \Rightarrow \mathrm{P}(a,2,2)}\,\forall\mathrm{L}}{\Gamma \Rightarrow \mathrm{P}(a,2,2)}\,c\mathrm{L}(plus\_zero)}{\Gamma \Rightarrow \mathrm{P}(a,b,c)}\,\sigma_1 \quad \dfrac{\dfrac{}{\mathrm{P}(sa,2,3) \Rightarrow \mathrm{P}(sa,2,3)}\,init}{\mathrm{P}(sa,b,sc) \Rightarrow \mathrm{P}(n,2,3)}\,\sigma_1}{\Gamma\ ,\ \mathrm{P}(a,b,c) \supset \mathrm{P}(sa,b,sc) \Rightarrow \mathrm{P}(n,2,3)}\,\supset\!\mathrm{L}}{\dfrac{\Gamma\ ,\ \forall a,b,c:\mathbb{N}\,.\,\mathrm{P}(a,b,c) \supset \mathrm{P}(sa,b,sc) \Rightarrow \mathrm{P}(n,2,3)}{\Gamma \Rightarrow \mathrm{P}(n,2,3)}\,c\mathrm{L}(plus\_succ)}\,\forall\mathrm{L}s}{\Gamma \Rightarrow \exists n:\mathbb{N}\,.\,\mathrm{P}(n,2,3)}\,\exists\mathrm{R}$$

The elided subderivations in the type theory prove that the numeric constants are indeed natural numbers, presumably by using the inductive definition of the natural numbers.

The *computed substitution* is then,

$$\sigma \coloneqq \sigma_1 \cdot \sigma_2 = [n \mapsto 1\,,a \mapsto 0\,,b \mapsto 2\,,c \mapsto 2\,,d \mapsto 2]$$

and the *computed answer* is $\sigma(n) = 1$, thus proving that three minus two is indeed one.

The corresponding proof in Coq using the indexed sequent tactics discussed in section 7.4 is as follows.

```
(* addition , as an inductively defined predicate *)
Inductive Plus : nat → nat → nat → Prop :=
  | plus_O : ∀ d : nat , Plus O d d
  | plus_S : ∀ a b c : nat , Plus a b c → Plus (S a) b (S c)
.

Example three_minus_two :
  ∃ n : nat , Plus n 2 3 .
Proof .
  (* the next line just introduces the constructors for Plus into the local context *)
  generalize plus_O ; intro plus_O ; generalize plus_S ; intro plus_S .
```

```
  ∃R . ∀Ls plus_S . ⊃L plus_S_a_b_c .
  - sub n (S a) . sub b 2 . sub c 2 . init plus_S_a_b_c' .
  - ∀Ls plus_O . sub a 0 . sub d 2 . init plus_O_d .
Qed .
```

We should mention that like Prolog, Coq is able to prove this type of simple theorem automatically.

By convention, all quantifiers are suppressed from the concrete syntax of Horn logic. This is unambiguous since both program and goal formulas are closed and only universal quantifiers are permitted to occur within Horn programs and only existential quantifiers are permitted to occur in Horn goals. Consequently, only anderrules of quantifiers will occur in the course of a derivation and all typing context variables that occur will be *obligation variables*, which in logic programming are called **logic variables**.

8.2 Uniform Proof for Hereditarily Harrop Logic

Essentially what we have done in the last section is to begin with a logical system (Horn logic) and describe a proof search strategy that is complete for it (SLD-resolution). One could just as well go the other way around by beginning with a proof search strategy and trying to find logical systems for which it is complete. This is what was done in a series of articles by Dale Miller and his collaborators, culminating in [Mil+91]. Their motivation was to generalize SLD-resolution to more expressive logics that could be used to give a purely logical interpretation to programming features such as module abstraction and local parameters. These investigations led to the development of a number of logic programming languages, notably λ-Prolog.

The strategy they investigated is a very simple one: given any sequent, if the succedent is not atomic then apply a sequent right rule corresponding to its principal connective. Miller has named this strategy **uniform proof**. Uniform proof has the property of being goal-directed, which is claimed to capture an intuitive notion of computation.[4] In [Mil+91], the authors decline to specify a course of action for atomic succedents but note that an obvious choice is *backward-chaining* (to be discussed shortly), which is also goal-directed. For further details about hereditarily Harrop logic and uniform proof, consult [Mil89a], [Mil+91], [Nad93] and [DP94]. Our purpose in this section is to characterize uniform proof with backward-chaining on atoms as a complete indexed sequent search strategy for the system of hereditarily Harrop logic.

[4] Though, we feel inclined to point out, only one out of many; at least to our mind (intuitiveness being in the mind of the beholder).

In uniform proof search, the connectives are given computational meaning by interpreting them as search instructions. The notation "$\Gamma \vdash_O A$" expresses the judgement that there exists a uniform proof of the inference $\Gamma \vdash A$.[5]

Definition 8.2.1 (connectives as search instructions) In *uniform proof*, the principal connective of a proposition is interpreted as a **search instruction** by inverting its sequent calculus right rules, as follows:

SUCCESS :

$$\Gamma \vdash_O \top \qquad \text{always}$$

FAILURE :

$$\Gamma \vdash_O \bot \qquad \text{never}$$

BOTH :

$$\Gamma \vdash_O A \wedge B \qquad \text{iff} \qquad \Gamma \vdash_O A \quad \text{and} \quad \Gamma \vdash_O B$$

EITHER :

$$\Gamma \vdash_O A \vee B \qquad \text{iff} \qquad \Gamma \vdash_O A \quad \text{or} \quad \Gamma \vdash_O B$$

AUGMENT :

$$\Gamma \vdash_O A \supset B \qquad \text{iff} \qquad \Gamma, A \vdash_O B$$

GENERIC :

$$\Gamma \vdash_O \forall x : X \,.\, A \qquad \text{iff} \qquad \Gamma \vdash_O A[x \mapsto e] \quad \text{for parameter } e \notin \mathrm{FV}(\Gamma, A)$$

INSTANCE :

$$\Gamma \vdash_O \exists x : X \,.\, A \qquad \text{iff} \qquad \Gamma \vdash_O A[x \mapsto t] \quad \text{for some term } t : X$$

Generalizing the terminology of Horn logic, in logic programming propositions that are permitted to occur in sequent antecedents are called **program formulas**, while those that are permitted to occur in sequent succedents are called **goal formulas**. One way to specify the language of a logical system is by giving a recursive grammar specifying its program and goal formulas. In [Mil+91], the language of hereditarily Harrop logic is presented.

Definition 8.2.2 (first-order hereditarily Harrop logic) The subsets of first-order propositions comprising the **hereditarily Harrop program formulas**, $\mathcal{P}_{h\mathrm{H}}$, and the **hereditarily Harrop goal formulas**, $\mathcal{G}_{h\mathrm{H}}$ are defined respectively by:

$$\begin{aligned} P &::= A \mid P \wedge P \mid \forall x : X \,.\, P \mid G \supset A \\ G &::= A \mid \top \mid G \wedge G \mid G \vee G \mid \forall x : X \,.\, G \mid \exists x : X \,.\, G \mid P \supset G \end{aligned}$$

[5] The "O" signifies "operational inference", a term due to Miller [Mil89a].

where A is is an atomic proposition.

In [Mil+91] it is shown that uniform provability is (sound and) complete with respect to intuitionistic provability for hereditarily Harrop logic; that is, for Γ a hereditarily Harrop program and G a hereditarily Harrop goal,

$$\Gamma \vdash_{\mathrm{I}} \mathsf{G} \qquad \Leftrightarrow \qquad \Gamma \vdash_{\mathrm{O}} \mathsf{G}$$

the demonstration describes a procedure for transforming an arbitrary intuitionistic proof into a uniform one.

However, in order for uniform proof to be implemented as a strategy to search for proofs, it must also specify a course of action when the succedent of a goal sequent is atomic. A natural choice is the goal-directed strategy of backward-chaining.

Backward-chaining is a partial strategy that involves program formulas of the form,

$$\forall \overrightarrow{x : \mathsf{X}} \,.\, \mathsf{G} \supset \mathsf{M}$$

where G is a goal formula and M is both a program formula and a goal formula; that is $\mathsf{M} \in \mathcal{P} \cap \mathcal{G}$. We will call such a proposition a **generalized implicative definite clause**, or just "definite clause" when no confusion is likely to result. Notice that in the grammar for hereditarily Harrop program formulas, the head of an implication must be atomic. So in hereditarily Harrop logic M will be an atom. **Backward-chaining** is the strategy of attempting to unify a goal formula with the head of a definite clause in the program, and if the unification succeeds, of applying the unifying substitution to the entire derivation and replacing the matched goal with the tail of the program clause that matched it.

This is a generalization of the combination of the left phase and the unification phase of SLD-resolution defined in the last section, since now the tail of a program clause need not be a conjunction of atoms but may instead be any goal formula. In other words, we may define backward-chaining as the following derived inference rule in the indexed sequent calculus:

$$\dfrac{\dfrac{\dfrac{\vdots}{x_1 : \mathsf{X}_1}\,\sigma \quad \cdots \quad \dfrac{\vdots}{x_n : \mathsf{X}_n}\,\sigma \qquad \dfrac{\dfrac{\sigma(\Gamma) \implies \sigma(\mathsf{T})}{\Gamma \implies \mathsf{T}}\,\sigma \qquad \dfrac{}{\sigma(\mathsf{H}) \implies \sigma(\mathsf{A})}\,\mathit{init}}{\mathsf{H} \implies \mathsf{A}}\,\sigma}{\Gamma \,,\, \mathsf{T} \supset \mathsf{H} \implies \mathsf{A}}\,\supset\mathrm{L}}{\Gamma \,,\, \forall \overrightarrow{x : \mathsf{X}} \,.\, \mathsf{T} \supset \mathsf{H} \implies \mathsf{A}}\,\forall\mathrm{L}s}{\Gamma \implies \mathsf{A}}\,c\mathrm{L} \tag{8.3}$$

In the right branch of the $\supset\mathrm{L}$ rule we again commit ourselves to using only the head of the selected program clause (H) and not the rest of the program (Γ) to prove the atomic goal (A). Therefore, this branch of the derivation must terminate immediately in an *init*

(modulo substitutions introduced by the indexed system). Miller has called this restricted use of the $\supset\mathsf{L}$ rule "simple" [Mil89a]. It is the restriction of the $\supset\mathsf{L}$ rule to simple instances that leads to backward-chaining on atoms.

The recursive grammar definition of hereditarily Harrop logic does not require program formulas to be definite clauses. However, any hereditarily Harrop program may be converted into a logically equivalent set of hereditarily Harrop definite clauses by a procedure known as hereditarily Harrop program elaboration, introduced in [Mil89b] and simplified in [Nad93].

Lemma 8.2.3 (hereditarily Harrop program elaboration) The following translation converts any hereditarily Harrop program formula into a logically equivalent set of hereditarily Harrop definite clauses:

$$\begin{array}{rcl} & \underline{elab} & \\ \mathcal{P}_{h\mathrm{H}} & \longrightarrow & \wp(\mathcal{P}_{h\mathrm{H}}) \\ \mathrm{A} & \longmapsto & \{\top \supset \mathrm{A}\} \\ \mathrm{P}_1 \wedge \mathrm{P}_2 & \longmapsto & elab(\mathrm{P}_1) \cup elab(\mathrm{P}_2) \\ \forall x : \mathrm{X} \,.\, \mathrm{P} & \longmapsto & \{\forall x : \mathrm{X} \,.\, \mathrm{P}' \mid \mathrm{P}' \in elab(\mathrm{P})\} \\ \mathrm{G} \supset \mathrm{A} & \longmapsto & \{\mathrm{G} \supset \mathrm{A}\} \end{array}$$

The definition of elaboration is extended to programs by setting $elab(\Gamma) \coloneqq \bigcup_{\mathrm{P}\in\Gamma} elab(\mathrm{P})$.

In [Nad93] it is shown that the strategy of uniform proof with backward-chaining on atoms is complete for first-order hereditarily Harrop logic.

8.3 Constructive Sequents

We now revisit the subject of goal-directed proof search from the perspective of our adjoint-theoretic characterization of the connectives. We will show that this leads to a syntactically richer, though ultimately no more expressive, logical system for which goal-directed search is a complete proof search strategy.

Intuitionistic logic enjoys the property of being **constructive**. In proof theory, this is often taken to mean that it satisfies the **disjunction property** and the **existence property**, which state respectively:

$$\vdash \mathrm{A} \vee \mathrm{B} \quad \Leftrightarrow \quad \vdash \mathrm{A} \text{ or } \vdash \mathrm{B} \qquad \text{and} \qquad \vdash \exists x : \mathrm{X} \,.\, \mathrm{A} \quad \Leftrightarrow \quad \vdash \mathrm{A}[x{\mapsto}t] \text{ for some } t : \mathrm{X}$$

In light of our investigation into adjunctions and the connectives, we would like to take a different perspective and argue that constructivity is characterized by freeness. Therefore, we will say that it is about granting left connectives a property that right connectives automatically enjoy: that the *only* way to construct a proof of a proposition with a given principal

connective is by using one of its introduction rules. Since falsehood has no introduction rules, we may add the rather trivial **falsehood property**:

$$\vdash \bot \quad \Leftrightarrow \quad \text{never}$$

This is simply another way of stating the consistency of a logic, since if $\bot$ were provable, then by $\bot-$, every proposition would be provable.

The proofs are immediate: in Gentzen's cut-free sequent calculus there are no applicable rules in the case of $\bot$, two in case of $\vee$ and one in case of $\exists$. In each case, the premises are exactly the conditions claimed. Thus we may always begin the search for a sequent proof of a **left proposition** – one where the principal connective is a *left connective* – from no assumptions by (nondeterministically) applying a right rule of its principal connective.

In contrast, the *right connectives* enjoy a much stronger property. Because their sequent calculus right rules are invertible, we may safely begin the search for a sequent proof of a **right proposition** from *any* set of assumptions with the right rule of its principal connective and never have to reconsider the decision. Note that this does not hold in general for left propositions. For example, any attempt to begin the search for a proof of even $\bot \implies \bot$, $\mathsf{A} \vee \mathsf{B} \implies \mathsf{A} \vee \mathsf{B}$ or $\exists x : \mathsf{X} \,.\, \mathsf{A} \implies \exists x : \mathsf{X} \,.\, \mathsf{A}$ with a right rule is doomed to failure.

With this in mind, we would like to generalize the definitions of the disjunction and existence properties.

Definition 8.3.1 ($*$-property) For $\vdash$ a consequence relation, $\mathcal{P}$ a set of propositions and $*$ (the name of) a connective, we say that $\mathcal{P}$ satisfies the $*$-property if for any finite $\Gamma \subseteq \mathcal{P}$ and G a proposition with principal connective $*$, the natural deduction rules $*+$, equivalently, the sequent calculus rules $*\mathsf{R}$, are collectively invertible for all instances of $\Gamma \vdash \mathsf{G}$.

The preceeding discussion implies that for intuitionistic first-order derivability, the empty theory has the $*$-property for every connective and that every theory has the $*$-property for right connectives.

In the early 1960s it was observed by Harrop [Har60] and Kleene [Kle62] that the disjunction and existence properties (for the empty theory) of intuitionistic first-order logic may be strengthened.

Proposition 8.3.2 In intuitionistic first-order logic we have the following two properties.

Strong disjunction property: If Γ contains no member in which a disjunction appears strictly positively then,

$$\Gamma \vdash \mathsf{A} \vee \mathsf{B} \qquad \Leftrightarrow \qquad \Gamma \vdash \mathsf{A} \quad \text{or} \quad \Gamma \vdash \mathsf{B}$$

Strong existence property: If Γ contains no member in which either a disjunction or existential appears strictly positively then,

$$\Gamma \vdash \exists x : \mathtt{X} \,.\, \mathtt{A} \qquad \Leftrightarrow \qquad \Gamma \vdash \mathtt{A}[x \mapsto t] \quad \text{for some term} \quad t : \mathtt{X}$$

The reverse directions of these two statements are an immediate consequence of the respective sequent right rules, thus their real content is describing conditions sufficient for instances of these rules to be invertible. As a historical matter, the derivation systems used by Harrop and Kleene differ from the ones presented here, however proofs using intuitionistic natural deduction can be found in [Pra65]. To these we again add the rather trivial **strong falsehood property**.

Lemma 8.3.3 (strong falsehood property) In intuitionistic first-order logic, if Γ contains no member in which $\bot$ appears strictly positively then,

$$\Gamma \vdash \bot \qquad \Leftrightarrow \qquad \text{never}$$

Proof. In Gentzen's cut-free sequent calculus, we consider what the first inference rule of a proof of $\Gamma \implies \bot$ could be. It can't be *init* because $\bot \notin \Gamma$ by assumption. It can't be a right rule because there is no $\bot\mathtt{R}$ rule. So it must be a logical left rule. But each such rule instance has the property that it has a premise that is a sequent $\Gamma' \implies \bot$ in which $\bot$ does not occur strictly positively in Γ'. Since an infinite chain of left rules cannot constitute a proof, no proof exists. □

It follows from the strong left-connective properties and the even stronger property for right connectives described above that:

Corollary 8.3.4 If Γ contains no member in which a left proposition appears strictly positively then for any non-atomic proposition $\mathtt{G}$, The sequent $\Gamma \implies \mathtt{G}$ has an intuitionistic sequent proof if and only if it has one whose first step consists of a right rule that introduces the principal connective of $\mathtt{G}$.

We may use this fact to determine a fragment of intuitionistic first-order logic for which the simple strategy of applying right rules whenever possible is complete with respect to full intuitionistic sequent provability. To do this we examine the circumstances under which the premise of corollary 8.3.4 could fail to hold throughout an entire sequent derivation. The only rules that allow a (sub)formula to "cross the turnstile", that is, to move between the antecedent and the succedent are those for implication.

When read from conclusion to premises, the only rule that allows a formula to cross the turnstile from right to left is $\supset\mathtt{R}$, which turns the hypothesis of an implication in

the succedent into a member of the antecedent. Thus, in order to ensure that no left-proposition occurs strictly positively in the antecedent, we must ensure that none occurs strictly positively in the hypothesis of an implication that itself occurs strictly positively in the succedent. Similarly, the only rule that allows a formula to cross the turnstile from left to right is $\supset\mathsf{L}$, which turns the hypothesis of an implication in the antecedent into a succedent. Since these two situations may be mutually recursive, in order to ensure that no left-proposition occurs strictly positively in the antecedent, we must ensure that none occurs positively at all in the antecedent, or negatively in the succedent. This justifies the following:

Proposition 8.3.5 (sufficient condition for completeness of uniform proof) If $\Gamma \implies \mathsf{G}$ is a sequent in which left connectives occur only negatively in Γ and only positively in G then it has an intuitionistic proof just in case it has one that follows the strategy of always applying a right rule when the succedent is not atomic.

We will say that a formula or connective occurs positively, respectively, negatively, in a sequent if it occurs positively, respectively, negatively, in the succedent or negatively, respectively, positively, in the antecedent. This is natural since a sequent is simply a meta-level implication.

Definition 8.3.6 (constructive sequent) We will call sequents in which left connectives occur only positively **constructive sequents** and their constituent parts **constructive antecedents** or "constructive contexts", composed of constructive assumptions; and **constructive succedents** or "constructive goals".

Thus proposition 8.3.5 states that uniform provability is complete with respect to intuitionistic provability for constructive sequents. We may now give a recursive grammar description for the language of constructive sequents.

Definition 8.3.7 (first-order constructive logic) The subsets of first-order propositions comprising the **constructive program formulas**, $\mathcal{P}_c$, and the **constructive goal formulas**, $\mathcal{G}_c$, are defined respectively by:

$$\begin{array}{lcl} \mathsf{P} & ::= & \mathsf{A} \mid \top \mid \mathsf{P} \wedge \mathsf{P} \mid \forall x : \mathsf{X} \,.\, \mathsf{P} \mid \mathsf{G} \supset \mathsf{P} \\ \mathsf{G} & ::= & \mathsf{A} \mid \top \mid \bot \mid \mathsf{G} \wedge \mathsf{G} \mid \mathsf{G} \vee \mathsf{G} \mid \forall x : \mathsf{X} \,.\, \mathsf{G} \mid \exists x : \mathsf{X} \,.\, \mathsf{G} \mid \mathsf{P} \supset \mathsf{G} \end{array}$$

where A is an atomic proposition.

This language is strictly richer than that of hereditarily Harrop logic from definition 8.2.2, and therefore gives rise to a syntactically richer notion of logic programming languages. But unfortunately, the added expressiveness is illusory, as the following lemmas show.

Lemma 8.3.8 Any constructive program formula is logically equivalent to one having the property that all implication heads are atomic.

Proof. In constructive program formulas the head of an implication must be either an atom or a right proposition. If it is the latter then the implication is equivalent to a constructive program formula where outermost implications have strictly simpler heads:

$$\begin{array}{rcl} \mathsf{G} \supset \top & \equiv_{\mathrm{I}} & \top \\ \mathsf{G} \supset (\mathsf{P}_1 \wedge \mathsf{P}_2) & \equiv_{\mathrm{I}} & (\mathsf{G} \supset \mathsf{P}_1) \wedge (\mathsf{G} \supset \mathsf{P}_2) \\ \mathsf{G}_1 \supset (\mathsf{G}_2 \supset \mathsf{P}) & \equiv_{\mathrm{I}} & (\mathsf{G}_1 \wedge \mathsf{G}_2) \supset \mathsf{P} \\ \mathsf{G} \supset (\forall x : \mathrm{X} \,.\, \mathsf{P}) & \equiv_{\mathrm{I}} & \forall x : \mathrm{X} \,.\, \mathsf{G} \supset \mathsf{P} \end{array}$$

We may use this fact to rewrite all program formulas to logically equivalent ones where all implication heads are atomic. □

Lemma 8.3.9 Permitting $\top$ to occur in program formulas does not allow us to prove any additional goals.

Proof. We may rewrite away any such occurrences by the logical equivalences:

$$\begin{array}{rcl} \top \wedge \mathsf{P} \equiv_{\mathrm{I}} \mathsf{P} \wedge \top & \equiv_{\mathrm{I}} & \mathsf{P} \\ \mathsf{G} \supset \top & \equiv_{\mathrm{I}} & \top \\ \forall x : \mathrm{X} \,.\, \top & \equiv_{\mathrm{I}} & \top \end{array}$$

and the fact that $\Gamma, \top \implies \mathsf{G}$ iff $\Gamma \implies \mathsf{G}$. □

Lemma 8.3.10 A constructive goal containing $\bot$ is either logically equivalent to a hereditarily Harrop one, or else unprovable from a constructive program.

Proof. By the logical equivalences:

$$\begin{array}{rcl} \bot \wedge \mathsf{G} \equiv_{\mathrm{I}} \mathsf{G} \wedge \bot & \equiv_{\mathrm{I}} & \bot \\ \bot \vee \mathsf{G} \equiv_{\mathrm{I}} \mathsf{G} \vee \bot & \equiv_{\mathrm{I}} & \mathsf{G} \\ \exists x : \mathrm{X} \,.\, \bot & \equiv_{\mathrm{I}} & \bot \end{array}$$

any constructive goal containing $\bot$ may either be rewritten to an equivalent hereditarily Harrop one or else is of the form:

$$\mathsf{G}_\bot ::= \bot \mid \mathsf{P} \supset \mathsf{G}_\bot \mid \forall x : \mathrm{X} \,.\, \mathsf{G}_\bot$$

A constructive sequent $\Gamma \implies \bot$ is unprovable by the strong falsehood property (lemma 8.3.3). A constructive sequent $\Gamma \implies \mathsf{P} \supset \mathsf{G}_\bot$ or $\Gamma \implies \forall x : \mathrm{X} \,.\, \mathsf{G}_\bot$ is unprovable because $\supset$ and $\forall$ are right connectives so their right rules are invertible:

$$\frac{\Gamma, \mathsf{P} \implies \mathsf{G}_\bot}{\Gamma \implies \mathsf{P} \supset \mathsf{G}_\bot}\ \supset\mathrm{R} \qquad \text{and} \qquad \frac{\Phi, x : \mathrm{X} \mid \Gamma \implies \mathsf{G}_\bot}{\Phi \mid \Gamma \implies \forall x : \mathrm{X} \,.\, \mathsf{G}_\bot}\ \forall\mathrm{R}$$

By induction on the size of the goal, any constructive sequent $\Gamma \implies \mathrm{G}_{\bot}$ is unprovable. □

Thus we may always rewrite a constructive sequent to an equivalent one that is either hereditarily Harrop, or which we know to be unprovable. Nevertheless, the language of constructive sequents may be useful for providing a more intuitive expression of program and goal formulas.

We may extend the program elaboration function to the language of constructive programs so that we may use backward-chaining for atomic goals:

Lemma 8.3.11 (constructive program elaboration) The following translation converts any constructive program formula into a logically equivalent set of constructive definite clauses:

$$
\begin{array}{rcl}
 & \underline{elab} & \\
\mathcal{P}_c & \longrightarrow & \wp(\mathcal{P}_c) \\
\mathrm{A} & \longmapsto & \{\top \supset \mathrm{A}\} \\
\top & \longmapsto & \emptyset \\
\mathrm{P}_1 \wedge \mathrm{P}_2 & \longmapsto & elab(\mathrm{P}_1) \cup elab(\mathrm{P}_2) \\
\forall x : \mathrm{X} \,.\, \mathrm{P} & \longmapsto & \{\forall x : \mathrm{X} \,.\, \mathrm{P}' \mid \mathrm{P}' \in elab(\mathrm{P})\} \\
\mathrm{G} \supset \mathrm{A} & \longmapsto & \{\mathrm{G} \supset \mathrm{A}\} \\
\mathrm{G} \supset \top & \longmapsto & \emptyset \\
\mathrm{G} \supset (\mathrm{P}_1 \wedge \mathrm{P}_2) & \longmapsto & elab(\mathrm{G} \supset \mathrm{P}_1) \cup elab(\mathrm{G} \supset \mathrm{P}_2) \\
\mathrm{G}_1 \supset (\mathrm{G}_2 \supset \mathrm{P}) & \longmapsto & elab((\mathrm{G}_1 \wedge \mathrm{G}_2) \supset \mathrm{P}) \\
\mathrm{G} \supset \forall x : \mathrm{X} \,.\, \mathrm{P} & \longmapsto & \{\forall x : \mathrm{X} \,.\, \mathrm{P}' \mid \mathrm{P}' \in elab(\mathrm{G} \supset \mathrm{P})\}
\end{array}
$$

This simply extends hereditarily Harrop program elaboration (lemma 8.2.3) by the equivalences in the proof of lemma 8.3.8.

8.4 Focused Proof Search

Finally, in this section we no longer restrict ourselves to various fragments of intuitionistic first-order logic such as Horn or hereditarily Harrop logic, and consider instead proof search in the whole, unrestricted logic. Although the semi-decision procedure of simply trying all proofs is complete, it is also hopelessly inefficient. This is due to the high degree of redundancy involved in such a naïve strategy: many attempted proofs will reach the same outcome because they differ in only inessential ways. Fortunately, we can do much better than this.

By considering the adjunction-based properties of derivations that we have been studying, we are naturally led to the strategy of eagerly and nondeterministically applying the

invertible eigenrules whenever possible. But this strategy can be refined even further into a type of search strategy called a "focusing" (or "focussing") strategy without sacrificing completeness.

Focusing was first developed by Andreoli for proof search in classical linear logic [And92; And01]. However, the principles of focusing are quite general and may be applied to intuitionistic (cartesian) logic as well. These are twofold. First, we should prefer to apply invertible rules whenever possible because doing so will never require future backtracking. Second, when we are forced to make a nondeterministic choice, which we may have to reconsider in the future, we should continue to explore the consequences of that choice rather than make gratuitous unrelated nondeterministic choices.

In Andreoli's description of **focusing**, the logical inference rules are partitioned into two sets, called asynchronous and synchronous. The **asynchronous rules** are invertible while **synchronous rules** are not and generally involve some form of nondeterministic choice. Focused proof search proceeds in two alternating phases, beginning in an inversion phase. In an **inversion phase**, asynchronous rules are applied repeatedly until none is applicable. A sequent to which no asynchronous rule is applicable is called a **neutral sequent**. When a sequent in the frontier of a derivation becomes neutral one of its propositions is nondeterministically selected for focus. This marks the transition from an inversion phase to a focused phase. In a **focused phase**, only synchronous rules may be applied, and only if the *principal formula* of a rule instance is the focused one. The application of a synchronous rule transmits the focus to the *active formula* of each premise (thus, each premise of a synchronous rule must have exactly one active formula). Within a derivation branch, focus is lost when no synchronous rule is applicable to the focused proposition. This marks the transition from a focused phase back to an inversion phase.

The situation becomes more complex when focusing is combined with the polarization of atoms, which we do not consider here in depth. Briefly, atomic propositions may be arbitrarily (but consistently) assigned either **negative polarity** or **positive polarity**. Negative atoms may be focused only on the left, and positive ones only on the right of a sequent. Since no logical sequent rule has an atomic principal formula, once an atom is selected for focus the derivation branch must immediately terminate in an *init* (possibly preceded by substitutions in the indexed case), which may be applied only during a focused phase. If *init* cannot be applied with the focused atom, then we must fail and backtrack. Assigning all atoms negative polarity results in a *backward-chaining* strategy, and dually, assigning all atoms positive polarity results in a forward-chaining strategy. Assigning differing polarities to different atoms results in a mixed strategy. Further details may be found in [CP08]. In the sequel we do not polarize atoms and thus allow unrestricted use of the initial sequent axiom.

In focused derivations, we will indicate a focused proposition by underlining it. Further, we will indicate the transitions between inversion and focused phases with the inference rules,

$$\frac{\Gamma, \underline{A} \Rightarrow B}{\Gamma, A \Rightarrow B}\ focus \qquad \frac{\Gamma, A \Rightarrow \underline{B}}{\Gamma, A \Rightarrow B}\ focus$$

$$\frac{\Gamma, A \Rightarrow B}{\Gamma, \underline{A} \Rightarrow B}\ blur \qquad \frac{\Gamma, A \Rightarrow B}{\Gamma, A \Rightarrow \underline{B}}\ blur$$

where, reading from conclusion to premise, *focus* may be applied only to a neutral sequent, and *blur* may be applied only when no synchronous rule applies to the focused proposition.

We note in passing that it is customary to use a different symbol to indicate a focused sequent (typically, "⟶"). However, it is unnecessary to do so: either the sequent is focused, in which case exactly one of its propositions is underlined, or else it is unfocused, in which case none is.

In the rest of this section we will assume that our sequent calculus has two non-invertible ∧L rules (like Gentzen's original system LJ) rather than a single invertible one, as described in chapter 7. The reasons for this are threefold. First, it permits the perfect correspondence between the concepts of *eigenrule* and *asynchronous rule*, and respectively, *anderrule* and *synchronous rule*. Next, it will allow us to use a theorem of Kleene's regarding the permutability of adjacent inferences in a sequent derivation (involving the consideration of 156 cases!) without modification. Finally, it permits easier comparison to another focused sequent system due to Dyckhoff and Pinto, to be described shortly. However, for practical proof search, we would likely prefer the system with the single invertible ∧L rule to further reduce nondeterminism.

Example 8.4.1 (a focused sequent proof) To illustrate the strategy of focusing we present the following example of a focused sequent proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \Rightarrow \underline{A}}\ init \quad \dfrac{}{A, \underline{B} \Rightarrow B}\ init}{\underline{A \supset B}, A \Rightarrow B}\ \supset L}{\underline{A \supset B}, A \Rightarrow B}\ focus}{A \supset B, A \Rightarrow \underline{B}}\ blur \qquad \dfrac{}{A \supset B, A, \underline{C} \Rightarrow C}\ init}{A \supset B, A, \underline{B \supset C} \Rightarrow C}\ \supset L \qquad \dfrac{}{A \supset B, A \Rightarrow \underline{A}}\ init}{\underline{A \supset (B \supset C)}, A \supset B, A \Rightarrow C}\ \supset L}{A \supset (B \supset C), A \supset B, A \Rightarrow C}\ focus}{A \supset (B \supset C), A \supset B \Rightarrow A \supset C}\ \supset R}{A \supset (B \supset C) \Rightarrow (A \supset B) \supset (A \supset C)}\ \supset R$$

This is not the only possible focused proof of this sequent, we could have just as well chosen

the implication $A \supset B$ for initial focus. However, beginning as we did, it would not be allowed to blur the focus on $B \supset C$ after the first application of $\supset L$ and focus on $A \supset B$ instead. Although doing so would still lead to a valid sequent proof, it would not be a *focused* proof.

The strategy of focusing subsumes that of uniform proof, which can be seen as follows. In the language of elaborated constructive sequents left propositions may not occur in the antecedent, so the only thing that can happen during an inversion phase is the decomposition of right propositions in the succedent. When a constructive sequent becomes neutral, it means that the succedent is either a left proposition or atomic. If it is a left proposition then we select it for focus and begin a focused phase. Alternating between inversion and succedent-focused phases, eventually the succedent must become atomic. In this case, we choose a clause from the antecedent for focus and decompose it until we reach the situation,

$$\dfrac{\dfrac{x_1 : X_1 \quad \cdots \quad x_n : X_n \qquad \dfrac{\Gamma \Longrightarrow \underline{G} \qquad \Gamma, \underline{A} \Longrightarrow A'}{\Gamma, \underline{G \supset A} \Longrightarrow A'}\,\supset L}{\Gamma, \underline{\forall \overrightarrow{x : X} \,.\, G \supset A} \Longrightarrow A'}\,\forall L s}{\Gamma \Longrightarrow A'}\; focus$$

where A and A' are both atomic. Now, if we maintain focus on A in the right branch of $\supset L$ (for example by assigning it negative polarity) then the only thing we can do there is to unify A and A', otherwise we must fail and backtrack. Notice that in the left branch of $\supset L$, the focus has returned to the succedent, so we simply repeat the process again as needed.

Andreoli's strategy of focusing is trivially sound for intuitionistic first-order logic, as it is merely imposes restrictions on the application of its inference rules. By simply erasing the focusing annotations and collapsing *focus* and *blur* inferences, every focused sequent derivation becomes an ordinary sequent derivation. Andreoli originally proved focusing complete for classical linear logic, in the sense that every provable sequent is provable by a focused derivation, by examining the permutability of adjacent inferences. An analogous result holds for intuitionistic first-order logic.

Definition 8.4.2 (adjacent inferences) In a sequent derivation, we will say that two primitive inferences are adjacent if they are separated by only structural rule instances.

Definition 8.4.3 (non-nested inferences) In a sequent derivation, we will say that two primitive inferences that can be applied to the same sequent are non-nested if they have different principal formulas.

In other words, primitive inferences R_1 and R_2 are non-nested if either R_1 and R_2 are both left rule instances acting on distinct assumptions, or else R_1 is a left rule instance acting on some assumption and R_2 is a right rule instance acting on the goal.

Unless a pair of non-nested inferences includes an *axiom*, it is always possible to apply them as adjacent inferences in either order. Given two adjacent non-nested inferences performed in one order, we wish to determine whether we could instead perform them in the opposite order, and still recover the same proof state (possibly after applying further inferences) that results from the original order of application. The following definitions make this notion precise.

Definition 8.4.4 (permutable inferences) In a sequent derivation with adjacent non-nested inferences R_1 and R_2 such that R_2 is below R_1, we say that R_1 is permutable below (or before) R_2 if applying R_1 to the sequent that was originally the conclusion of R_2 and then applying R_2 to some resulting premise creates a frontier from which the original frontier may be recovered by the application of further inferences. We also consider axioms such as $\bot\mathsf{L}$ and $\top\mathsf{R}$ to be permutable below any other inference, despite the fact that applying them first eliminates the opportunity, or need, to apply the other inference.

Definition 8.4.5 (permutable inference rules) For inference rules R_1 and R_2, we say that R_1 is permutable below R_2 if for any adjacent non-nested instances of R_1 and R_2, the instance of R_1 is permutable below the instance of R_2.

There is a theorem due to Kleene [Kle52] that the only "forbidden permutations" in intuitionistic first-order logic not involving the connectives $\top$ and $\bot$ are,

$$(\forall\mathsf{L}\,,\forall\mathsf{R})\,,(\forall\mathsf{L}\,,\exists\mathsf{L})\,,(\exists\mathsf{R}\,,\exists\mathsf{L})\,,(\supset\mathsf{L}\,,\supset\mathsf{R})\,,(\supset\mathsf{L}\,,\vee\mathsf{L})\,,(\vee\mathsf{R}\,,\vee\mathsf{L})\,,(\exists\mathsf{R}\,,\vee\mathsf{L})$$

meaning that the first rule in each pair is not permutable below the second. Including the nullary connectives adds the pairs,

$$(\supset\mathsf{L}\,,\top\mathsf{R})\,,(\forall\mathsf{L}\,,\top\mathsf{R})\,,(\forall\mathsf{L}\,,\bot\mathsf{L})\,,(\forall\mathsf{L}\,,\top\mathsf{R})$$

Notice that in each case, the first rule is an anderrule (hence synchronous) while the second is an eigenrule (hence asynchronous). Therefore asynchronous inferences may always be permuted below synchronous ones; and within the set of asynchronous, respectively, synchronous rules, the permuting of inferences preserves provability. This suffices to show the completeness of focusing for intuitionistic first-order logic.

It turns out that for all permutations of eigenrules, and for those of anderrules not involving $\supset\mathsf{L}$, we have an even stronger and simpler result.

Definition 8.4.6 (strictly commuting inferences) For non-nested primitive inferences R_1 and R_2, we will say that they are strictly commuting if the frontier obtained by first applying R_1 and then applying R_2 to all of the resulting premises to which it is applicable is the same as that obtained by reversing the roles of R_1 and R_2.

Definition 8.4.7 (strictly commuting inference rules) We will say that a pair of inference rules is strictly commuting if all pairs of their respective instances are strictly commuting.

Lemma 8.4.8 (eigenrules commute strictly) All pairs of eigenrules are strictly commuting.

Proof. The eigenrules for $\top$ and $\bot$ commute strictly with every eigenrule since their application results in an empty frontier. The strict commutativity of the remaining eigenrules is presented in figure 8.1, except for $(\vee\mathsf{L}, \vee\mathsf{L})$, which does not readily fit in the width of a page, but is nonetheless just as straightforward as the other cases, as the interested reader may check. □

Lemma 8.4.9 (strictly commuting anderrules) All pairs of anderrules not involving $\supset\mathsf{L}$ and another right anderrule are also strictly commuting.

Proof. See figure 8.2. □

The reason for the failure of strict commutativity of $\supset\mathsf{L}$ with $\wedge\mathsf{L}$ and $\forall\mathsf{L}$ is that doing $\supset\mathsf{L}$ first causes the principal formula of the other rule to appear in both of the resulting premises, where different choices of conjunct, respectively, representative, may then be made. Nonetheless, in both cases, as well as that for $(\supset\mathsf{L}, \supset\mathsf{L})$, permutability can be salvaged (as guaranteed by Kleene's theorem) by retaining a copy of the principal formula of the other rule in one of the subderivations and then applying that rule again after the $\supset\mathsf{L}$. This would suggest the heuristic of preferring $\supset\mathsf{L}$ inferences to $\wedge\mathsf{L}$ (when using the two non-invertible rules) or $\forall\mathsf{L}$ in proof search.

We now examine some proof-theoretic properties of focusing through the lens of our adjoint-theoretic interpretation of intuitionistic first-order logic. The justification for eagerly applying asynchronous rules is clear: these rules are eigenrules, hence invertible. In fact, their application preserves, not only provability, but indeed the set of all normal derivations, up to bijection, under the Prawitz translation. Because of the invertibility of left eigenrules, there is no need to retain their principal formulas in the premises. Thus each asynchronous inference strictly decreases the size of the resulting goal sequents in the frontier of a derivation, so can't cause non-termination.

Consider the issue of the order of application of eigenrules. For instances that are nested, such as the two instances of $\supset\mathsf{R}$ in example 8.4.1, there is no choice, the outermost one must be applied first. Since the composition of eigenrules induces a composite natural bijection, we may think of trees of nested eigenrules as a single derived admissible eigenrule.

On the other hand, for instances that are not nested we have a genuine choice. Given any two non-nested eigenrule instances we know that that the frontier resulting from one

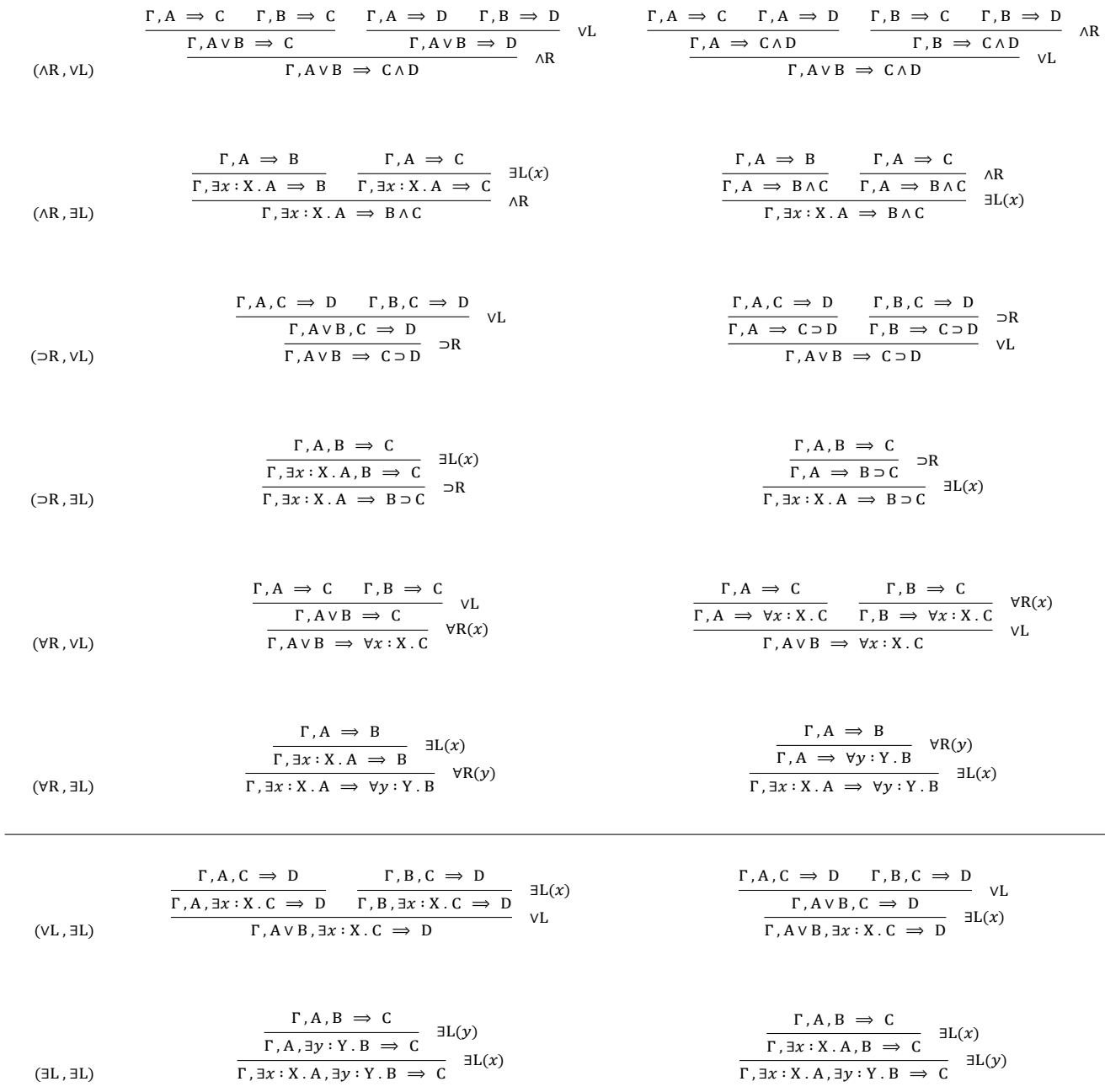


Figure 8.1: strict commutativity of eigenrules

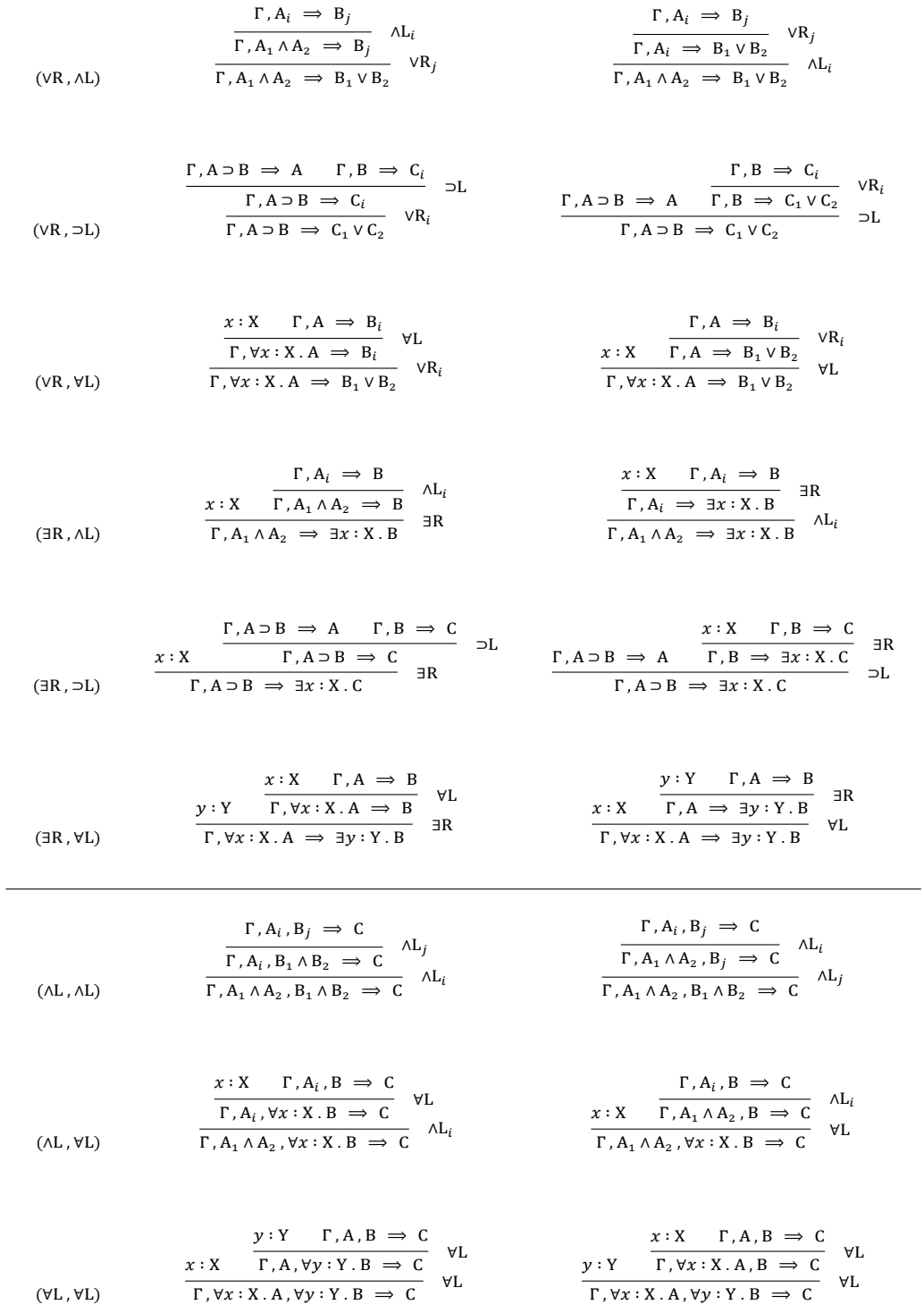


Figure 8.2: strictly commutative anderrules

order of application will have a set of proofs whose Prawitz translation is in bijection with the set obtained from the frontier resulting from the other order of application. But as we have just seen by the strict commutativity of the eigenrules, an even stronger result holds: the frontiers will be exactly the same.

Our adjoint-theoretic understanding of the eigenrules explains this as well. Consider the example of $(\wedge\mathsf{R},\vee\mathsf{L})$ in figure 8.1. On the one hand we have the composite natural bijection,

$$\textsc{Prop}_\Gamma\,(\mathsf{A}\vee\mathsf{B}\to\mathsf{C}\wedge\mathsf{D}) \quad\cong\quad ((\textsc{Prop}_\Gamma\times\textsc{Prop}_\Gamma)\times(\textsc{Prop}_\Gamma\times\textsc{Prop}_\Gamma))\,((\mathsf{A}\,,\mathsf{B})\,,(\mathsf{A}\,,\mathsf{B})\to(\mathsf{C}\,,\mathsf{C})\,,(\mathsf{D}\,,\mathsf{D}))$$

and on the other,

$$\textsc{Prop}_\Gamma\,(\mathsf{A}\vee\mathsf{B}\to\mathsf{C}\wedge\mathsf{D}) \quad\cong\quad ((\textsc{Prop}_\Gamma\times\textsc{Prop}_\Gamma)\times(\textsc{Prop}_\Gamma\times\textsc{Prop}_\Gamma))\,((\mathsf{A}\,,\mathsf{A})\,,(\mathsf{B}\,,\mathsf{B})\to(\mathsf{C}\,,\mathsf{D})\,,(\mathsf{C}\,,\mathsf{D}))$$

Since the cartesian product is associative and commutative up to isomorphism, these are each isomorphic to

$$\textsc{Prop}_\Gamma{}^4\,((\mathsf{A}\,,\mathsf{A}\,,\mathsf{B}\,,\mathsf{B})\to(\mathsf{C}\,,\mathsf{D}\,,\mathsf{C}\,,\mathsf{D}))$$

resulting in the same frontier. Similar results hold for all other possible pairs of non-nested adjacent eigenrule inferences, as the reader may check.

Because the order of asynchronous inference application is irrelevant in this strong sense, Andreoli called the nondeterminism involved **inessential nondeterminism**, and proposed to consider such inferences to be acting in parallel, as a single derived inference rule comprising the inversion phase. This has the effect of eliminating a great deal of redundancy from proof search as it collapses all derivations differing only in the order of application of adjacent eigenrules into a single derivation while preserving completeness in the strong sense of preserving all normal derivations under the Prawitz translation.

Unfortunately, not all nondeterminism in proof search is inessential. The choices made when applying synchronous rules introduce a form of **essential nondeterminism** into proof search. Kleene's result guarantees that by making such choices in a focused way we don't sacrifice provability. But notice that this is a weaker statement than to say that we don't give up any normal proofs. However, lemma 8.4.9 shows us that synchronous inferences in sets not containing both $\supset\mathsf{L}$ and another right anderrule may also be seen as acting in parallel.

We now consider several notions of completeness in the context of logic programming. Since the whole purpose of a proof search strategy is to prune the space of possible proofs under consideration, it would be counterproductive for a search strategy to find every possible proof, that is, to possess **all-proofs completeness**. It would also be undesirable to fail to find any proofs at all, if indeed there were proofs to be found. Such a situation would correspond to a failure of **provability completeness**, which is the usual notion. However,

these extremes are not the only possible kinds of completeness, and the question of just what sort of completeness a strategy does, or should, enjoy turns out to be a delicate one.

The concept of *input-output semantics* in logic programming means that mere provability completeness is not sufficient for such purposes. For example, given the logic program for addition in example 8.1.1, and the goal $\exists x, y : \mathbb{N} \,.\, \mathsf{P}(x, y, 3)$, which asks whether there is a pair of natural numbers that sum to 3, we should not be satisfied if a proof search strategy is able to find a proof with computed answer $(x := 0, y := 3)$, but not one with, say, $(x := 1, y := 2)$. Such a situation would correspond to a failure of **answer completeness**. For the purpose of logic programming, this would seem to be the minimum standard of completeness that we desire. But it is unclear to what aspect of formal proof structure this type of completeness corresponds.

Another form of completeness that we could seek, which is stronger than answer completeness but weaker than all-proofs completeness is **normal-proofs completeness**, where "normal proofs" refers to normal natural deductions under the Prawitz translation. This type of completeness has the advantage that it has a clear proof-theoretic interpretation. However, from the perspective of logic programming, it is stronger than necessary since different normal proofs may correspond to the same input-output behavior. Nevertheless, its clear connection to proof semantics, which is so-far lacking in the case of computed answer completeness, makes it a good candidate for study.

Such an approach has been pursued by Roy Dyckhoff and Luís Pinto [DP96; DP99] Their sequent system, called "MJ" may be seen as a focusing strategy, though it is based not on the work of Andreoli, but rather on a term calculus proposed by Hugo Herbelin. In [Her95] Herbelin presented a term calculus of realizers for an intuitionistic sequent calculus that is isomorphic to normal terms of the λ-calculus. Under the Curry-Howard correspondence, Dyckhoff and Pinto repurposed this as a sequent calculus isomorphic to normal natural deduction.

Dyckhoff and Pinto were motivated by the desire to investigate Brouwer–Heyting–Kolmagorov (BHK) style semantics; that is, not questions of, "is this proposition valid", but rather, "what are the possible proofs of this proposition". Thus they were interested in issues of proof enumeration rather than just provability. As proof objects they chose normal natural deductions for their clear interpretation as computational procedures under the Curry-Howard correspondence. But they also wanted to work in a sequent system, for the reasons mentioned at the beginning of section 7.3. And this is precisely what Herbelin's calculus provided.

Sequent system MJ, presented in figure 8.3, may be seen as a focused sequent calculus with focusing only on the left. That is, although propositions in the context may be focused, those in the goal may not be. Notice that each of the left eigenrules triggers a loss of

$$\frac{}{\Gamma, \underline{P} \Rightarrow P}\ \mathit{init} \qquad \frac{\Gamma, A, \underline{A} \Rightarrow B}{\Gamma, A \Rightarrow B}\ cL$$

$$\frac{}{\Gamma \Rightarrow \top}\ \top R \qquad \frac{}{\Gamma, \underline{\bot} \Rightarrow A}\ \bot L$$

$$\frac{\Gamma \Rightarrow A \quad \Gamma \Rightarrow B}{\Gamma \Rightarrow A \wedge B}\ \wedge R \qquad \frac{\Gamma, \underline{A} \Rightarrow C}{\Gamma, \underline{A \wedge B} \Rightarrow C}\ \wedge L_1 \quad \frac{\Gamma, \underline{B} \Rightarrow C}{\Gamma, \underline{A \wedge B} \Rightarrow C}\ \wedge L_2$$

$$\frac{\Gamma \Rightarrow A}{\Gamma \Rightarrow A \vee B}\ \vee R_1 \quad \frac{\Gamma \Rightarrow B}{\Gamma \Rightarrow A \vee B}\ \vee R_1 \qquad \frac{\Gamma, A \Rightarrow C \quad \Gamma, B \Rightarrow C}{\Gamma, \underline{A \vee B} \Rightarrow C}\ \vee L$$

$$\frac{\Gamma, A \Rightarrow B}{\Gamma \Rightarrow A \supset B}\ \supset R \qquad \frac{\Gamma \Rightarrow A \quad \Gamma, \underline{B} \Rightarrow C}{\Gamma, \underline{A \supset B} \Rightarrow C}\ \supset L$$

$$\frac{\Gamma \Rightarrow A[e \mapsto x]}{\Gamma \Rightarrow \forall x : X \,.\, A}\ \forall R^{\dagger} \qquad \frac{t : X \quad \Gamma, \underline{A[x \mapsto t]} \Rightarrow B}{\Gamma, \underline{\forall x : X \,.\, A} \Rightarrow B}\ \forall L$$

$$\frac{t : X \quad \Gamma \Rightarrow A[x \mapsto t]}{\Gamma \Rightarrow \exists x : X \,.\, A}\ \exists R \qquad \frac{\Gamma, A[e \mapsto x] \Rightarrow B}{\Gamma, \underline{\exists x : X \,.\, A} \Rightarrow B}\ \exists L^{\dagger}$$

† e does not occur in the conclusion

Figure 8.3: Sequent System MJ

focus. This can be understood in terms of Andreoli's characterization of focusing: the applicability of a left eigenrule implies that we should now be in an inversion phase, so if we are transitioning from a focused phase we blur, then we apply the eigenrule and remain in the inversion phase or transition to a right-focused phase, which are not distinguished in this derivation system, until we select a new focus on the left with the contraction rule. Unlike Adreoli-style focusing, system MJ allows for the selection of a new focus on the left during an inversion or right-focused phase. Thus MJ permits more derivations than Andreoli-style focusing does. Since MJ-derivations are in bijection with normal natural deductions, this would imply that Andrioli-style focusing is not complete for normal proofs, though we have not yet been able to characterize the class of proofs missed. A further difference is that MJ duplicates left propositions in the context by contraction when focusing on them. By our adjoint-theoretic analysis, we know that this should never be necessary since the left rules of left propositions are eigenrules and so induce bijections of normal natural deduction derivations.

We do not yet know whether Andreoli-style focusing is sufficient to provide the answer-completeness desired in logic programming, or whether a system like MJ is needed.

Chapter 9

Conclusion, Related and Future Work

In this thesis we have shown how the combination of the hyperdoctrine interpretation of typed intuitionistic first-order logic with the adjoint-theoretic description of its connectives leads to a uniform presentation of Gentzen's formal derivation systems of natural deduction and sequent calculus in which the concept of connective chirality plays a central role. By taking the adjoint-theoretic characterization of the quantifiers seriously, we were led to decompose each of their non-invertible rules into a purely logical rule and a substitution. This in turn has led us to the indexed sequent calculus, a formalism that reifies the concept of logic variable and is thus well-suited to the task of proof search.

We have characterized the computation mechanisms for several systems of logic programming as search strategies within the indexed sequent calculus and seen that the strategies of SLD-resolution, uniform proof and focusing form a sequence in which each subsumes its predecessor as a special case. In analyzing the strategy of focusing we have seen that it is always safe to apply eigenrules eagerly because these rules determine a bijection of proof objects, and therefore preserve provability. We have also seen why we may collect any subderivation consisting solely of eigenrule inferences into an admissible "big step" rule, which not only preserves provability, but in fact yields the same derivation frontier regardless of the order in which the constituent inferences are applied. This justifies one of the two principles underlying the strategy of focusing.

In logic programming we desire strategies that are complete, not just for provability, but for answers as well. We have encountered at least one derivation system that fulfills this criterion, in the form of system MJ. Further investigation is needed to give answer completeness a proof-theoretic characterization and to determine under what conditions Andreoli-style focusing possesses it.

Another line of inquiry that is suggested is to give a categorical semantics for Andreoli's concept of parallelism in proof search. Because permuting instances of strictly commuting

inference rules has no effect on the rest of a derivation, it seems sensible to admit derived inference rules for combinations of these. This would allow a representation of sequent derivations that is more compact and that eliminates inessential nondeterminism. For instance, the derivation in example 7.3.5 could be represented as:

$$\dfrac{\dfrac{\dfrac{}{x:\mathrm{X},y:\mathrm{Y}\ \Rightarrow\ x:\mathrm{X}}\ \mathit{init}}{x:\mathrm{X},y:\mathrm{Y}\ \Rightarrow\ \underline{u}:\mathrm{X}}\ \sigma \qquad \dfrac{\dfrac{}{x:\mathrm{X},y:\mathrm{Y}\ \Rightarrow\ y:\mathrm{Y}}\ \mathit{init}}{x:\mathrm{X},y:\mathrm{Y}\ \Rightarrow\ \underline{v}:\mathrm{Y}}\ \sigma \qquad \dfrac{\dfrac{}{x:\mathrm{X},y:\mathrm{Y}\ \mid\ \mathrm{R}(x,y)\ \Rightarrow\ \mathrm{R}(x,y)}\ \mathit{init}}{x:\mathrm{X},y:\mathrm{Y},\underline{u}:\mathrm{X},\underline{v}:\mathrm{Y}\ \mid\ \mathrm{R}(\underline{u},y)\ \Rightarrow\ \mathrm{R}(x,\underline{v})}\ \sigma}{x:\mathrm{X},y:\mathrm{Y}\ \mid\ \forall u:\mathrm{X}.\,\mathrm{R}(u,y)\ \Rightarrow\ \exists v:\mathrm{Y}.\,\mathrm{R}(x,v)}\ \forall\mathrm{L}\parallel\exists\mathrm{R}$$
$$\dfrac{x:\mathrm{X},y:\mathrm{Y}\ \mid\ \forall u:\mathrm{X}.\,\mathrm{R}(u,y)\ \Rightarrow\ \exists v:\mathrm{Y}.\,\mathrm{R}(x,v)}{\emptyset\ \mid\ \exists y:\mathrm{Y}.\,\forall u:\mathrm{X}.\,\mathrm{R}(u,y)\ \Rightarrow\ \forall x:\mathrm{X}.\,\exists v:\mathrm{Y}.\,\mathrm{R}(x,v)}\ \forall\mathrm{R}\parallel\exists\mathrm{L}$$

Here σ is the obvious substitution and we have used the operator "$-\parallel-$" to represent the parallel composition of strictly commuting inference rules. The obvious categorical tool to interpret this sort of parallelism would be a tensor product.

The categorical semantics presented here can in principle be extended to higher-order logics. This would likely involve treating the type theory and the logic as a single dependent system. Categorical interpretations of constructive dependent type theories have been developed using both indexed categories [Tay99] and fibrations [Jac99]. Interpretations of formal derivation systems, like those presented here, should be possible for these as well.

Remaining within first-order logic, it is possible to add a logical constant representing the predicate of term equality. Such a relation, with the properties of Leibniz equality, may be given an adjoint-theoretic description, but using a somewhat richer notion of hyperdoctrine. Briefly, if we have existential quantifiers for diagonal morphisms in the base category ($\Delta : \mathrm{X} \longrightarrow \mathrm{X},\mathrm{X}$), then we may define equality by,

$$-=- \quad :=\quad \exists\Delta(\top)$$

where $\exists\Delta \dashv \Delta^*$ and Δ^* is the functor turning a proposition over two variables of the same type into one over a single variable of that type, which is substituted for both of the original variables:

$$y:\mathrm{X},z:\mathrm{X}\mid \mathrm{A}\ \textsc{prop} \qquad \overset{\Delta^*}{\longmapsto} \qquad x:\mathrm{X}\mid \mathrm{A}[y\mapsto x,z\mapsto x]\ \textsc{prop}$$

This adjunction has the natural bijection,

$$\dfrac{\mathit{refl} : \textsc{Prop}(x:\mathrm{X})\,(\top \to (\Delta^* \circ \exists\Delta)(\top))}{\mathrm{id} : \textsc{Prop}(y:\mathrm{X},z:\mathrm{X})\,(\exists\Delta(\top) \to \exists\Delta(\top))}$$

where $\mathit{refl} := \eta(\top)$ witnesses the **reflexivity** of equality. Turning this bijection upside-down and reading it as a sequent inference rule gives,

$$\dfrac{y:\mathrm{X},z:\mathrm{X}\ \mid\ y=z\ \Longrightarrow\ y=z}{x:\mathrm{X}\ \mid\ \top\ \Longrightarrow\ x=x}$$

Dispatching the initial sequent premise gives a right rule (and introduction rule) for equality. Further details may be found in [Law70] and [See83].

Another axis along which this work could be extended is that of the type theory and term language. In the preceding we have deliberately chosen a very simple version in order to focus our attention on the logic. The type theory that we used has only atomic types, and the term language has no relations other than syntactic equality. Clearly, this is not satisfactory for practical logic programming. In principle, there is no problem in making the category of types also bicartesian closed, though this will of course introduce relations among terms like those in the fibers. For a practical system, it would be desirable to have algebraic datatypes as well. Categorical interpretations for logic programming systems with algebraic datatypes have been investigated by Lipton, McGrail and Amato [FFL03; McG99; ALM09], who have also investigated adding arbitrary relations between terms, as are found in constraint logic programming.

Categorical semantics have been given for other logics as well, perhaps most comprehensively in the case of linear logic by Melliès in [Mel09]. There, adjunctions also play a role in defining the connectives, but a more complex one. It may be that intuitionistic logic is unique in having such an direct relationship between adjunctions and the connectives. But this raises interesting questions. Because the adjoint nature of the connectives plays such a central role in the algebraic properties of this logic, one wonders what abstract categorical structures there may be governing the properties of other logics.

Appendix A

Categorical Notation

Category theory is rife with ambiguous and conflicting notation. Our aim is to follow established conventions as much as possible to avoid contributing to the second of these problems; except when doing so would further exacerbate the first. With this as our guiding principle, we adopt the following conventions.

Categorical variables will generally be written as blackboard-bold capital Latin letters:

$$\mathbb{A}\,,\ \mathbb{B}\,,\ \mathbb{C}\,,\ \text{etc.}$$

Categorical constants will be written as small-cap abbreviations. We bow to convention and name our categories after their objects rather than their arrows. Examples include:

- SET : the category of sets and functions
- CAT : the 2-category of small categories, functors and natural transformations

Members of categories such as objects and arrows will have their sorts indicated by the infix typing relation symbol, "$:$", thusly:

- We express that $\mathtt{A}$ is an *object* in the category $\mathbb{C}$ like this:

$$\mathtt{A} : \mathbb{C}$$

- We express that f is an *arrow* in the category $\mathbb{C}$ like this:

$$f :: \mathbb{C}$$

- In categories with higher categorical structure, we may continue this pattern in the obvious way. In general, $\theta : \underbrace{\cdots}_{n+1} : \mathbb{C}$ (or $\theta \overset{(n)}{:} \mathbb{C}$), indicates that θ is an n-cell of $\mathbb{C}$.

For example, we can say that φ is a natural transformation (between functors between small categories) like this:

$$\varphi ::: \textsc{Cat}$$

Chaining of type declarations may be used as a shorthand. Thus

$$\mathsf{A} : \mathbb{C} : \textsc{Cat}$$

is short for

$$\mathbb{C} : \textsc{Cat} \qquad \text{and} \qquad \mathsf{A} : \mathbb{C}$$

Hom objects : We denote the object of morphisms between two objects of a category like this:

$$\mathbb{C}\,(\mathsf{A} \to \mathsf{B})$$

or, if the category is obvious or irrelevant, like this:

$$\mathsf{A} \longrightarrow \mathsf{B}$$

Composition will always be indicated with an infix symbol. For normal order composition we use "$\cdot$" (as it is very unobtrusive) and for applicative order composition we use "$\circ$" (as it is very customary). These two types of composition may be read as "then" and "after", respectively. Thus, the composition,

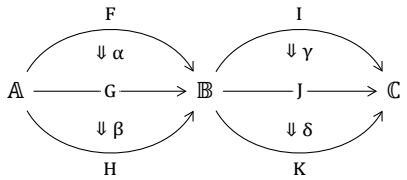
$$\mathsf{A} \xrightarrow{\quad f \quad} \mathsf{B} \xrightarrow{\quad g \quad} \mathsf{C}$$

can be expressed equally well by

$$f \cdot g \qquad \text{or} \qquad g \circ f$$

though we strongly prefer the former (first things first), and generally use the latter only when there is an argument following the composition, to which it is being applied (see below).

Higher composition : In 2-categories, we express so called "horizontal composition" of 2-cells (such as natural transformations) by "$\cdot\cdot$" (or "$\circ\circ$"). So, for example in situation,



the interchange law states:

$$(\alpha \cdot \beta) \cdot\cdot (\gamma \cdot \delta) = (\alpha \cdot\cdot \gamma) \cdot (\beta \cdot\cdot \delta) \quad : \quad F \cdot I \longrightarrow H \cdot K$$

The rationale comes from the globular interpretation of higher categories. There, ordinary composition of two n-cells is along their common $(n-1)$-cell boundary, while horizontal composition is along their common $(n-2)$-cell boundary. In general, composition of two n-cells along their common $(n-k)$-cell boundary is expressed with "$\cdot\underbrace{\cdots}_{k}\cdot$ " (or "$\overset{(k)}{\cdot}$").

Implicit promotion of n-cells : For $k = j+1$, a j-cell may be implicitly promoted to a k-cell as the identity k-cell on its underlying j-cell. By induction, promotion is well defined for all $k > j$. So, for example, with the diagram above,

$$(\alpha \cdot\cdot I) \cdot (G \cdot\cdot \gamma) = \alpha \cdot\cdot \gamma = (F \cdot\cdot \gamma) \cdot (\alpha \cdot\cdot J) \quad : \quad F \cdot I \longrightarrow G \cdot J$$

Products, coproducts and exponentials , when they exist in a category, are written infix as "$\times$", "$+$", and "$\supset$", respectively. $\times$ and $+$ are *left-associative*, while $\supset$ is *right-associative*. Their relative precedence is as in ordinary arithmetic and all have higher precedence than composition. For example, the iterated curry adjunction for cartesian closed categories can be written

$$\frac{f : A \times B \times \cdots \times Y \longrightarrow Z}{\lambda^{(n)}(f) : A \longrightarrow B \supset \cdots \supset Y \supset Z}$$

In particular, the category of functors from $\mathbb{A}$ to $\mathbb{B}$ and their natural transformations is "$\mathbb{A} \supset \mathbb{B}$". (you may know it as "$\mathbb{B}^{\mathbb{A}}$" or "$[\mathbb{A}, \mathbb{B}]$").

Initial and terminal objects , when they exist in a category, are denoted by "0" and "1", respectively (or sometimes by "$\bot$" and "$\top$", especially when the category is a *preorder* or the intended interpretation is *logic*). For $A : \mathbb{C}$,

$$\mathbb{C}(0 \to A) = \{i_A\} \qquad \text{and} \qquad \mathbb{C}(A \to 1) = \{!_A\}$$

Application of a function to its argument is indicated by juxtaposition, written with the function symbol preceding its argument, and the argument optionally enclosed in gratuitous parentheses (for emphasis). Application is *left-associative* and has higher precedence than composition and product, coproduct or exponential formation. So, for example if $\varphi : (\mathbb{C} \supset \mathbb{D})(F \to G)$ then,

$$\forall A, B : \mathbb{C} \,.\, \forall f : \mathbb{C}(A \to B) \,.\, F(f) \cdot \varphi(B) = \varphi(A) \cdot G(f)$$

expresses the naturality of φ.

Structures comprising multiple components, such as adjunctions will be written as tuples, with the intended interpretation depending on the sort of their structure. These tuples are generally dependent and some of the components may be redundant. The purpose of this notation is simply to act as a succinct way to bind names to the relevant components for future reference. Identifiers ranging over them will be written in script font, for example,

$$\text{let } \mathcal{A} \coloneqq (\mathbb{C}\,,\mathbb{D}\,,\mathsf{F}\,,\mathsf{G}\,,\eta\,,\varepsilon) \text{ be an adjunction...}$$

Appendix B

Indexed Sequent Tactics in Coq

```
(*
 *  indexed_tactics.v
 *
 *  Coq tactics implementing the rules of "indexed sequent calculus"
 *   for intuitionistic first-order logic
 *)


(* uses unicode notation *)
Require Export Utf8 .


(* ∀ tactics *)

Ltac is_forall t :=
  match type of t with
  | ∀ x : ?X , _ =>
    match type of X with
    | Prop => fail 1 (* it's an implication *)
    | _ => idtac (* it's a universal quantification *)
    end
  | _ => fail 1 t "is not a universal quantification"
  end
.

Ltac is_forall_goal :=
  match goal with
  | [ |- ∀ x : ?X , _ ] =>
    match type of X with
```

```
    | Prop  =>  fail 1  (* it's an implication *)
    | _  =>  idtac  (* it's a universal quantification *)
    end
  | _  =>  fail 1 "goal is not a universal quantification"
  end
.

Tactic Notation "∀R"  :=
  is_forall_goal ; intro
.

Tactic Notation "∀Rs"  :=
  let rec
    self := ∀R ; try self
  in
    self
.

Ltac forall_left  :=
  fun
    universal_hypothesis obligation_variable remaining_hypothesis
  =>
    is_forall universal_hypothesis ;
    match type of universal_hypothesis with
    | ∀ x : ?X , _  =>
      evar (obligation_variable : X) ;
      generalize universal_hypothesis ;
      intro remaining_hypothesis ;
      let
        z := (eval unfold obligation_variable in obligation_variable)
      in
        specialize (remaining_hypothesis z)
  end
.

Tactic Notation "∀L" constr (H)  :=
  is_forall H ;
  match type of H with
  | ∀ x : ?X , _  =>
    let
      y := fresh x in let H' := fresh H"_"y
    in
```

```
      forall_left H y H'
  end
.

Tactic Notation "∀Ls" constr (H) :=
  let rec
    self := fun hyp =>
      is_forall hyp ;
      match type of hyp with
      | ∀ x : ?X , _ =>
        let
          y := fresh x in let hyp' := fresh hyp"_"y
        in
          forall_left hyp y hyp' ; try (self hyp' ; clear hyp')
      end
  in
    self H
.


(* ∃ tactics *)

Ltac is_exists t :=
  match type of t with
  | ∃ x : _ , _ => idtac
  | _ => fail 1 t "is not an existential quantification"
  end
.

Ltac is_exists_goal :=
  match goal with
  | [ |- ∃ x : _ , _ ] => idtac
  | _ => fail 1 "goal is not an existential quantification"
  end
.

Ltac exists_left :=
  fun
    existential_hypothesis generic_variable remaining_hypothesis
  =>
    is_exists existential_hypothesis ;
    elim existential_hypothesis ;
```

```
    intro generic_variable ;
    intro remaining_hypothesis
.

Tactic Notation "∃L" constr (H) :=
  is_exists H ;
  match type of H with
  | ∃ x : _ , _ =>
    let
      y := fresh x in let H' := fresh H"_"y
    in
      exists_left H y H' ; clear H
  end
.

Tactic Notation "∃Ls" constr (H) :=
  let rec
    self := fun hyp =>
      is_exists hyp ;
      match type of hyp with
      | ∃ x : ?X , _ =>
        let
          y := fresh x in let hyp' := fresh hyp"_"y
        in
          exists_left hyp y hyp' ; clear hyp ; try (self hyp')
      end
  in
    self H
.

Tactic Notation "∃R" :=
  is_exists_goal ;
  match goal with
  | [ |- ∃ x : ?X , _ ] =>
    let
      y := fresh x
    in
      evar (y : X) ;
      let
        z := eval unfold y in y
      in
        exists z
```

```
  end
.

Tactic Notation "∃Rs" :=
  let rec
    self := ∃R ; try self
  in
    self
.


(* ∧ tactics *)

Ltac is_and t :=
  match type of t with
  | _ ∧ _ => idtac
  | _ => fail 1 t "is not a conjunction"
  end
.

Ltac is_and_goal :=
  match goal with
  | [ |- _ ∧ _ ] => idtac
  | _ => fail 1 "goal is not a conjunction"
  end
.

Tactic Notation "∧R" :=
  is_and_goal ; apply conj
.

Tactic Notation "∧Rs" :=
  let rec
    self := ∧R ; try self
  in
    self
.

Tactic Notation "∧L1" constr (H) :=
  is_and H ;
  let
    H1 := fresh H
```

```
  in
    generalize H ; intro H1 before H ; apply proj1 in H1
.

Tactic Notation "∧L2" constr (H) :=
  is_and H ;
  let
    H2 := fresh H
  in
    generalize H ; intro H2 before H ; apply proj2 in H2
.

Tactic Notation "∧L" constr (H) :=
  is_and H ;
  let
    H1 := fresh H in let H2 := fresh H
  in
    generalize H ; intro H1 ; apply proj1 in H1 ;
    generalize H ; intro H2 ; apply proj2 in H2 ;
    move H2 before H ; move H1 before H ; clear H
.

Tactic Notation "∧Ls" constr (H) :=
  let rec
    self := fun hyp =>
      is_and hyp ;
      let
        hyp1 := fresh hyp in let hyp2 := fresh hyp
      in
        generalize hyp ; intro hyp1 before hyp ; apply proj1 in hyp1 ;
        generalize hyp ; intro hyp2 before hyp1; apply proj2 in hyp2 ;
        try (self hyp1) ; try (self hyp2) ; clear hyp
  in
    self H
.


(* ∨ tactics *)

Ltac is_or t :=
  match type of t with
  | _ ∨ _ => idtac
```

```
  | _  =>  fail 1 t "is not a disjunction"
  end
.

Ltac is_or_goal  :=
  match goal with
  | [ |- _ ∨ _ ]  =>  idtac
  | _  =>  fail 1 "goal is not a disjunction"
  end
.

Tactic Notation "∨L" constr (H)  :=
  is_or H ;
  let
    H1 := fresh H in let H2 := fresh H
  in
    elim H ;
    [
      intro H1 before H ; clear H
      |
      intro H2 before H ; clear H
    ]
.

Tactic Notation "∨Ls" constr (H)  :=
  let rec
    self := fun hyp  =>
      is_or hyp ;
      let
        hyp1 := fresh hyp in let hyp2 := fresh hyp
      in
        elim hyp ;
        [
          intro hyp1 before hyp ; clear hyp ; try (self hyp1)
          |
          intro hyp2 before hyp ; clear hyp ; try (self hyp2)
        ]
  in
    self H
.

Tactic Notation "∨R1"  :=
```

```
  is_or_goal ; apply or_introl
.

Tactic Notation "∨R2" :=
  is_or_goal ; apply or_intror
.

Ltac or_intro ind :=
  let rec self :=
    fun n =>
      match goal with
      | [ |- _ ∨ _ ] =>
        match eval compute in n with
        | O => fail 2 "index out of bounds"
        | 1 => apply or_introl
        | S (S ?m) => apply or_intror ; self (S m)
        | _ => fail 2 n "is not a valid index"
        end
      | _ =>
        match eval compute in n with
        | 1 => idtac
        | _ => fail 3 "index out of bounds"
        end
      end
  in
    self ind
.

Tactic Notation "∨R" constr (num) :=
  or_intro num
.


(* ⊤ tactics *)

Notation "⊤" := True .

Ltac is_truth t :=
  match type of t with
  | ⊤ => idtac
  | _ => fail 1 t "is not truth"
  end
```

```
.

Ltac is_truth_goal :=
  match goal with
  | [ |- ⊤ ] => idtac
  | _ => fail 1 "goal is not truth"
  end
.

Tactic Notation "⊤R" :=
  is_truth_goal ; constructor
.

Tactic Notation "⊤L" constr (H) :=
  is_truth H ; fail "no left rule for truth"
.


(* ⊥ tactics *)

Notation "⊥" := False .

Ltac is_falsehood t :=
  match type of t with
  | ⊥ => idtac
  | _ => fail 1 t "is not falsehood"
  end
.

Ltac is_falsehood_goal :=
  match goal with
  | [ |- ⊥ ] => idtac
  | _ => fail 1 "goal is not falsehood"
  end
.

Tactic Notation "⊥L" constr (H) :=
  is_falsehood H ; destruct H
.

Tactic Notation "⊥R" :=
  is_falsehood_goal ; fail "no right rule for falsehood"
```

```
.


(* ⊃ tactics *)

Ltac is_implies t :=
  match type of t with
  | ?A → ?B => (* it's either ∀, ⊃ or other → *)
    match type of A with
    | Prop =>
      match type of B with
      | Prop => idtac (* it's an implication *)
      | _ => fail 2 (* the conclusion is not a proposition *)
      end
    | _ => fail 1 (* it's a universal quantification *)
    end
  | _ => fail 1 t "is not an implication"
  end
.

Ltac is_implies_goal :=
  match goal with
  | [ |- ?A → ?B ] => (* it's either ∀, ⊃ or other → *)
    match type of A with
    | Prop =>
      match type of B with
      | Prop => idtac (* it's an implication *)
      | _ => fail 2 (* the conclusion is not a proposition *)
      end
    | _ => fail 1 (* it's a universal quantification *)
    end
  | _ => fail 1 "goal is not an implication"
  end
.

Tactic Notation "⊃R" :=
  is_implies_goal ; intro
.

Tactic Notation "⊃Rs" :=
  let rec
    self := ⊃R ; try self
```

```
  in
    self
.

Tactic Notation "⊃L" constr (H) :=
  is_implies H ;
  let
    H' := fresh H"'"
  in
    match type of H with
    | ?A → ?B => cut B ; [intro H' before H ; clear H | apply H]
    end
.

Tactic Notation "⊃Ls" constr (H) :=
  let rec
    self := fun hyp =>
      is_implies hyp ;
      let
        hyp' := fresh hyp"'"
      in
        match type of hyp with
        | ?A → ?B =>
          cut B ; [intro hyp' before hyp ; clear hyp ; try (self hyp') | apply hyp]
        end
  in
    self H
.


(* sub and init *)

Tactic Notation "sub" hyp (x) constr (t) :=
  (* x must be a reference to an existential variable *)
  let
    x' := eval unfold x in x
  in
    is_evar x' ;
    first
    [
      (* in case t is also a reference: *)
      let t' := (eval unfold t in t) in unify x' t'
```

```
      |
      (* otherwise: *)
      unify x' t
      |
      fail 1  t "cannot be substituted for" x
    ] ; clear x
.

Tactic Notation "init" constr (H) :=  exact H .


(* big-step rule for focusing inversion phase *)

Ltac eigen :=
  match goal with
  | [ |- ⊤ ] => ⊤R
  | [ |- _ ∧ _ ] => ∧R
  | [ |- _ → _ ] => ⊃R
  | [ |- ∀ _ , _ ] => ∀R
  | [ H : ⊥ |- _ ] => ⊥L H
  | [ H : _ ∨ _ |- _] => ∨L H
  | [ H : ∃ _ , _ |- _] => ∃L H
  | _ => fail 1 "no eigenrule applies"
  end
.

Ltac eigens :=
  let rec
    self := eigen ; try self
  in
    self
.
```

Bibliography

[ALM09] Gianluca Amato, James Lipton, and Robert McGrail. "On the Algebraic Structure of Declarative Programming Languages". In: *Theoretical Computer Science* 410.46 (2009), 4626–4671.

[And92] Jean-Marc Andreoli. "Logic Programming with Focusing Proofs in Linear Logic". In: *Journal of Logic and Computation* 2.3 (1992), pp. 207–347.

[And01] Jean-Marc Andreoli. "Focussing and Proof Construction". In: *Annals of Pure and Applied Logic* 107 (2001).

[AM89] Andrea Asperti and Simone Martini. "Projections Instead of Variables: a category theoretic interpretation of logic programs". In: International Conference for Logic Programming. MIT Press, 1989.

[Awo10] Steve Awodey. *Category Theory*. second edition. Oxford Logic Guides 49. Oxford University Press, 2010.

[AB09] Steve Awodey and Andrej Bauer. *Introduction to Categorical Logic*. lecture notes. 2009. URL: http://www.andrew.cmu.edu/user/awodey/catlog/.

[BS09] John Baez and Mike Stay. "Physics, Topology, Logic and Computation: A Rosetta Stone". 2009. URL: http://arxiv.org/abs/0903.0340.

[BW98] Michael Barr and Charles Wells. *Category Theory for Computing Science*. third edition. Centre de Recherches Mathematiques, 1998.

[BC04] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development*. Texts in Theoretical Computer Science. Springer, 2004.

[CP08] Kaustuv Chaudhuri and Frank Pfenning. *A Logical Characterization of Forward and Backward Chaining in the Inverse Method*. Tech. rep. Carnegie Mellon University, 2008.

[Coq12] Coq Development Team. *The Coq Proof Assistant Reference Manual (version 8.4)*. INRIA, 2012. URL: http://coq.inria.fr/documentation.

[CH88] Thierry Coquand and Gérard Huet. "The Calculus of Constructions". In: *Information And Computation* 76 (1988), pp. 95–120.

[CP90] Thierry Coquand and Christine Paulin-Mohring. "Inductively Defined Types". In: *Proceedings of Colog*. Ed. by Per Martin-Löf and G. Mints. Vol. 417. Lecture Notes in Computer Science. Springer-Verlag, 1990.

[Dou93] Daniel Dougherty. "Some Lambda Calculi with Categorical Sums and Products". In: *Rewriting Techniques and Applications*. Vol. 690. Lecture Notes in Computer Science. Springer-Verlag, 1993, pp. 137–151.

[Dum91] Michael Dummett. *The Logical Basis of Metaphysics*. The William James Lectures, 1976. Harvard University Press, 1991.

[DP94] Roy Dyckhoff and Luís Pinto. "Uniform Proofs and Natural Deductions". In: *International Conference on Automated Deduction*. 1994.

[DP96] Roy Dyckhoff and Luís Pinto. *A Permutation-Free Sequent Calculus for Intuitionistic Logic*. Tech. rep. St. Andrews University, 1996.

[DP99] Roy Dyckhoff and Luís Pinto. "Proof Search in Constructive Logics". In: *Logic Colloquium 1997*. Ed. by S. Barry Cooper and John K. Truss. London Mathematical Society Lecture Notes 258. Association for Symbolic Logic. Cambridge University Press, 1999.

[FFL03] Stacy Finkelstein, Peter Freyd, and James Lipton. "A New Framework for Declarative Programming". In: *Theoretical Computer Science* 300 (2003), pp. 91–160.

[Gal93] Jean Gallier. "Constructive Logics. Part I: A Tutorial on Proof Systems and Typed λ-Calculi". In: *Theoretical Computer Science* 110.2 (1993).

[Gen35] Gerhard Gentzen. "Untersuchungen Über das Logische Schließen". In: *Mathematische Zeitschrift* 39 (1935), pp. 176–210, 405–431.

[Gha95a] Neil Ghani. "Adjoint Rewriting". PhD thesis. University of Edinburgh, 1995.

[Gha95b] Neil Ghani. "βη-Equality for Coproducts". In: *Typed Lambda Calculi and Applications*. Vol. 902. Lecture Notes in Computer Science. 1995, pp. 171–185.

[Gir72] Jean-Yves Girard. "Interprétation Fonctionnelle et Élimination des Coupures de l'Arithmétique d'Ordre Supérieur". PhD thesis. Université de Paris VII, 1972.

[Gir87] Jean-Yves Girard. "Linear Logic". In: *Theoretical Computer Science* 50 (1987), pp. 1–101.

[GLT89] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Vol. 7. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1989.

[HM92] Victor Harnik and Michael Makkai. "Lambek's Categorical Proof Theory and Läuchli's Abstract Realizability". In: *Journal of Symbolic Logic* 57.1 (1992), pp. 200–230.

[Har60] Ronald Harrop. "Concerning Formulas of the Types $A \to B \vee C$, $A \to (Ex)B(x)$". In: *Journal of Symbolic Logic* 25.1 (1960), pp. 27–32.

[Her95] Hugo Herbelin. "A λ-Calculus Structurally Isomorphic to Gentzen-style Sequent Calculus Structure". In: *Proceedings of the 1994 workshop Computer Science Logic*. Vol. 933. Lecture Notes in Computer Science. Springer, 1995, pp. 61–75.

[How80] W. A. Howard. "The Formulae-as-Types Notion of Construction". In: *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Ed. by J.P. Seldin and J. R. Hindley. Academic Press, 1980, pp. 479–490.

[Jac99] Bart Jacobs. *Categorical Logic and Type Theory*. Vol. 141. Studies in Logic. Elsevier, 1999.

[JG95] C. Barry Jay and Neil Ghani. "The Virtues of Eta-Expansion". In: *Journal of Functional Programming* 5.2 (1995), pp. 135–154.

[KP96] Yoshiki Kinoshita and John Power. "A Fibrational Semantics for Logic Programs". In: *Lecture Notes in Computer Science* 1050 (1996).

[Kle52] Stephen Kleene. "Permutability of Inferences in Gentzen's Calculi LK and LJ". In: *Memoirs of the American Mathematical Society* 10 (1952), pp. 1–26.

[Kle62] Stephen Kleene. "Disjunction and Existence Under Implication in Elementary Intuitionistic Formalisms". In: *Journal of Symbolic Logic* 27.1 (1962), pp. 11–18.

[Kri05] Ayalur Krishnan. "Universal Quantifiers in Logic Programming via Indexed Categories". PhD thesis. Wesleyan University, 2005.

[Lam68] Joachim Lambek. "Deductive Systems and Categories I". In: *Mathematical Systems Theory*. Springer-Verlag, 1968.

[Lam69] Joachim Lambek. "Deductive Systems and Categories II". In: *Lecture Notes in Mathematics* 86 (1969), pp. 76–122.

[Lam72] Joachim Lambek. "Deductive Systems and Categories III". In: *Lecture Notes in Mathematics* 274 (1972), pp. 57–82.

[LS86] Joachim Lambek and Philip Scott. *Introduction to Higher Order Categorical Logic*. Cambridge University Press, 1986.

[Law69] F. William Lawvere. "Adjointness in Foundations". In: *Dialectica* 23 (1969).

[Law70] F. William Lawvere. "Equality in Hyperdoctrines and the Comprehension Schema as an Adjoint Functor". In: *Proceedings of the New York Symposium on Applications of Categorical Logic*. Americal Mathematical Society, 1970.

[Llo84] J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 1984.

[Mac98] Saunders Mac Lane. *Categories for the Working Mathematician*. second edition. Graduate Texts in Mathematics. Springer, 1998.

[MM92] Saunders Mac Lane and Ieke Moerdijk. *Sheaves in Geometry and Logic*. Springer-Verlag, 1992.

[Mak93a] Michael Makkai. "The Fibrational Formulation of Intuitionistic Predicate Logic I: completeness according to Gödel, Kripke, and Läuchli. I." In: *Notre Dame Journal Formal Logic* 34.3 (1993), pp. 334–377.

[Mak93b] Michael Makkai. "The Fibrational Formulation of Intuitionistic Predicate Logic I: completeness according to Gödel, Kripke, and Läuchli. II." In: *Notre Dame Journal Formal Logic* 34.4 (1993), pp. 471–498.

[Mar84] Per Martin-Löf. *Intuitionistic Type Theory*. Studies in Proof Theory. Bibliopolis, 1984.

[McG99] Robert McGrail. "Monads, Predicates and Categorical Logic Programming". PhD thesis. Wesleyan University, 1999.

[Mel09] Paul-André Melliès. "Categorical Semantics of Linear Logic". In: *Interactive Models of Computation and Program Behaviour*. Société Mathématique de France, 2009.

[Mil89a] Dale Miller. "A Logical Analysis of Modules in Logic Programming". In: *Journal of Logic Programming* 6 (1989), pp. 79–108.

[Mil89b] Dale Miller. "Lexical Scoping as Universal Quantification". In: *Sixth International Conference on Logic Programming*. MIT Press, 1989.

[Mil+91] Dale Miller et al. "Uniform Proofs as a Foundation for Logic Programming". In: *Annals of Pure and Applied Logic* 51 (1991).

[Nad93] Gopalan Nadathur. "A Proof Procedure for the Logic of Hereditary Harrop Formulas". In: *Journal of Automated Reasoning* 11 (1993).

[NM95] Ulf Nilsson and Jan Małuszyński. *Logic, Programming and Prolog*. second edition. John Wiley and Sons, 1995. URL: http://www.ida.liu.se/~ulfni/lpp.

[nla] nlab contributers. *Principle of Equivalence*. URL: http://ncatlab.org/nlab/show/principle+of+equivalence (visited on 2012).

[Pfe09] Frank Pfenning. *Constructive Logic*. lecture notes. 2009. URL: http://www.cs.cmu.edu/~fp/courses/15317-f09/.

[Pot77] Garrel Pottinger. "Normalization as the Homomorphic Image of Cut-Elimination". In: *Annals of Mathematical Logic* 12 (1977), pp. 323–357.

[Pra65] Dag Prawitz. *Natural Deduction: A Proof-Theoretical Study*. Stockholm Studies in Philosophy 3. Almqvist and Wiksell, 1965.

[Pra71] Dag Prawitz. "Ideas and Results in Proof Theory". In: *Second Scandinavian Logic Symposium*. Ed. by J. E. Fenstad. North-Holland, 1971, pp. 235–307.

[See79] Robert Seely. "Weak Adjointness in Proof Theory". In: *Lecture Notes in Mathematics* 753 (1979), pp. 697–701.

[See83] Robert Seely. "Hyperdoctrines, Natural Deduction and the Beck Condition". In: *Zeitscrift für Mathematische Logik und Grundlagen der Mathematik* (1983).

[Tay99] Paul Taylor. *Practical Foundations of Mathematics*. Cambridge Studies in Advanced Mathematics 59. Cambridge University Press, 1999.

[TS00] Anne Troelstra and Helmut Schwichtenberg. *Basic Proof Theory*. second edition. Cambridge Tracts in Theoretical Computer Science 43. Cambridge University Press, 2000.

[Zuc74] J Zucker. "The Correspondence Between Cut-Elimination and Normalization". In: *Annals of Mathematical Logic* 7 (1974), pp. 1–112, 113–155.

Index