On the unity of duality

Noam Zeilberger

August 6, 2008

Abstract

Most type systems are agnostic regarding the evaluation strategy for the underlying languages, with the value restriction for ML which is absent in Haskell as a notable exception. As type systems become more precise, however, detailed properties of the operational semantics may become visible because properties captured by the types may be sound under one strategy but not the other. For example, intersection types distinguish between call-by-name and call-by-value functions, because the subtyping law $(A \to B) \cap (A \to C) \leq A \to (B \cap C)$ is unsound for the latter in the presence of effects.

In this paper we develop a proof-theoretic framework for analyzing the interaction of types with evaluation order, based on the notion of *polarity.* Polarity was discovered through linear logic, but we propose a fresh origin in Dummett's program of justifying the logical laws through alternative verificationist or pragmatist "meaning-theories", which include a bias towards either introduction or elimination rules. We revisit Dummett's analysis using the tools of Martin-Löf's judgmental method, and then show how to extend it to a unified *polarized logic,* with Girard's "shift" connectives acting as intermediaries. This logic safely combines intuitionistic and dual intuitionistic reasoning principles, while simultaneously admitting a *focusing interpretation* for the classical sequent calculus.

Then, by applying the Curry-Howard isomorphism to polarized logic, we obtain a single programming language in which evaluation order is reflected at the level of *types.* Different logical notions correspond directly to natural programming constructs, such as pattern-matching, explicit substitutions, values and call-by-value continuations. We give examples demonstrating the expressiveness of the language and type system, and prove a basic but modular type safety result. We conclude with a brief discussion of extensions to the language with additional effects and types, and sketch the sort of explanation this can provide for operationally-sensitive typing phenomena.

1 Introduction

An essay by John Reynolds centers on the dangers of specifying the semantics of a language by means of a definitional interpreter, when the meaning of the defining language is itself potentially unclear (Reynolds, 1972). In a functional programming language, the result of function application can depend on whether evaluation order is call-by-value or call-by-name (due to the presence of non-termination and other side-effects), and Reynolds observes that a direct style interpreter leaves this choice only *implicit* in the evaluation order of the defining language—thus carrying little explanatory power, particularly in the case of a "meta-circular interpreter" (a Lisp interpreter written in Lisp, for example). He then goes on to give a careful account of how to make evaluation order *explicit* by writing the interpreter in continuation-passing style.

Yet a language definition includes not only the dynamic semantics of expressions, but also their static syntax. Does the lesson of Reynolds' essay extend "all the way down"? For example, should evaluation order be explicit at the level of *types?* The reason one might at first doubt this possibility is that, for the most part, we have been getting along fine using very similar type systems for functional languages (such as ML and Haskell) with very different evaluation strategies. And for good reason: the theoretical foundation of all these languages is the Curry-Howard correspondence with intuitionistic natural deduction, i.e., the simply-typed lambda calculus. And the lambda-calculus can be given many different evaluation strategies.

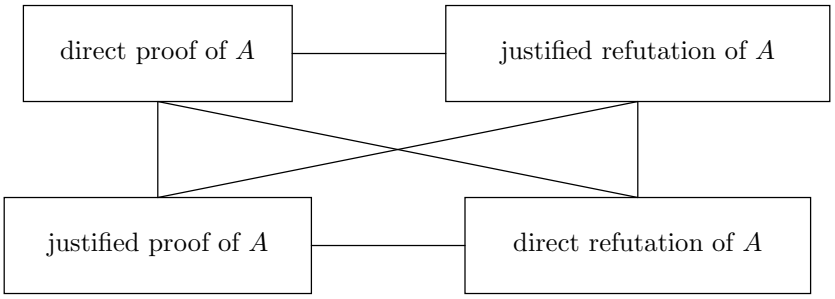
Yet, one notable crack in this foundation is the so-called *value restriction* on polymorphism, or second-order quantification. The original implementation of ML was unsound due to the interaction of poly-

morphism with effects such as mutable storage and callcc (Harper and Lillibridge, 1991), and prompted various workarounds, with the eventual adoption of a non-logical, syntactic restriction on polymorphism-introduction (Wright, 1995; Milner et al., 1997). It might be easy to dismiss this as an anomaly, but recent studies of intersection and union types in operational settings (Pfenning and Davies, 2001; Dunfield and Pfenning, 2004) have suggested a wider pattern of *operationally-sensitive typing phenomena.* The usual intersection introduction rule, for instance, is unsound in effectful call-by-value languages (requiring a value restriction), as is the standard law of subtyping $(A \to B) \cap (A \to C) \leq A \to (B \cap C)$. While these are sound under call-by-name evaluation (even in the presence of effects), there, *unions* pose similar problems, e.g., $(A \to C) \cap (B \to C) \leq (A \cup B) \to C$ becomes unsound.

In short, at a sufficient level of granularity it becomes clear that type systems must reflect their underlying language's evaluation order. But does this destroy the logical interpretation of programming languages? We propose otherwise: one can recover evaluation order as a logical notion by first refining the logical foundation. In this paper, we will explain how evaluation order can be described in terms of *focusing* and *polarity,* proof-theoretic innovations introduced within the setting of linear logic by Andreoli (1992) and Girard (1993), and later extensively developed by Girard (2001) and Laurent (2002). Rather than starting directly with the technical machinery of linear logic, however, we will attempt to give a rational reconstruction of polarity and focusing by going back to Michael Dummett's examination, in the 1976 William James Lectures, of the *justification* of the logical laws.

Briefly, Dummett suggested that the logical laws could be justified by alternative proof-theoretic "meaning-theories", which employ opposite biases towards either the introduction or the elimination rules of natural deduction. In one approach, which Dummett calls *verificationist,* it is assumed that any provable proposition has a direct proof (i.e., ending in a sequence of introduction rules)—this assumption can be used to justify any other potential law (e.g., an elimination rule). But alternatively, one can assume that any valid *consequence* of a proposition can be found directly (i.e., beginning with a sequence of elimination rules), and use this to justify other potential laws. Dummett calls this dual approach a *pragmatist* meaning-theory. How can these two alternatives be related?

Our answer, which goes beyond Dummett's requirement of *harmony,* combines insights from Martin-Löf's judgmental method (Martin-Löf, 1996; Pfenning and Davies, 2001) and Laurent's polarized linear logic. First, we identify the different tools of Dummett's analysis as different *judgments* about propositions. By concentrating on proofs and refutations (rather than arbitrary consequences), we can highlight the symmetries between these judgments as a sort of "square of opposition":



The top half of the square corresponds to the verificationist meaning-theory, the bottom half to the pragmatist. But we can go further: these two sets of judgments are actually about different kinds of propositions! Direct proof and justified refutation are about *positive* propositions. Direct refutation and justified proof are about *negative* propositions. In other words—which is the lesson we learn from linear logic—the verificationist and pragmatist approaches define distinct, oppositely polarized connectives. What this means is that rather than considering the two approaches only in isolation, we can treat them as interacting *fragments* of a unified, polarized logic. Indeed, not only can we ascribe the usual operation of negation to the horizontal arrows,[1] but we can go further and find connectives completing the square. The diagonal arrows correspond

[1] To be precise, these are actually two different, oppositely polarized negations.

to the dualizing operation $(-)^\perp$ familiar from linear logic, while the vertical arrows correspond to the more recently-discovered "shift" connectives (Girard, 2001).

The title of this paper is a pun on Girard's "unity of logic" (Girard, 1993), and a series of papers by different authors, all suggesting a duality between call-by-value and call-by-name languages (Filinski, 1989; Curien and Herbelin, 2000; Selinger, 2001; Wadler, 2003). By giving this polarized logic a Curry-Howard interpretation, we construct a *single* programming language in which evaluation order is explicitly reflected at the level of types, and call-by-value and call-by-name evaluation freely interact. The different judgments of the square correspond to different syntactic categories, some quite familiar from the theory and practice of programming languages. For example, direct proofs correspond precisely to values in the ordinary sense of ML, and justified refutations to call-by-value continuations defined by pattern-matching. This framework, we propose, is well-suited for studying computational behavior from a logical perspective.

2 The logic of proofs and refutations

2.1 The meaning(s) of the connectives

In the 1976 William James Lectures, Michael Dummett considered the possibility of *justifying* the logical laws (Dummett, 1991). Arguing against the formalist position that the laws need no justification, and likewise against the holistic position that they can only be considered in toto, Dummett proposed that the logical laws could be justified through an analysis of the "meaning" of the connectives. Moreover, that this analysis could be purely proof-theoretic—or to put it glibly, Dummett expanded upon Wittgenstein's "meaning is use", describing how to read the meanings of the logical connectives in the logical laws themselves.

But *which* laws determine the meanings of the connectives? Dummett gives two alternative interpretations, which he calls *verificationist* and *pragmatist meaning-theories.* The first is more familiar, going back to an offhand remark by Gentzen (while speaking about his system of natural deduction) that "an introduction rule gives, so to say, a definition of the constant in question" (Gentzen, 1935, p. 80). Gentzen's remark was first developed mathematically by Prawitz (1974), by treating the meaning of a proposition as given by its *canonical proofs,* or verifications.[2] Under what Dummett calls the "Fundamental Assumption"—if a proposition is true then it must have a canonical proof—one obtains an *upwards justification procedure* for arbitrary logical laws: an inference rule is justified if any canonical proofs of its premises can be transformed into a canonical proof of its conclusion. The difference between Prawitz's original formulation and Dummett's generalization is mostly in the definition of "canonical", and the corresponding scope of the justification procedure. Prawitz defined canonical proofs as ending in an introduction rule, and used this to justify the standard elimination rules. For example, consider the standard set of rules for conjunction in intuitionistic natural deduction:

$$\frac{A \quad B}{A \wedge B} \qquad \frac{A \wedge B}{A} \qquad \frac{A \wedge B}{B}$$

By (Prawitz's) definition, a canonical proof of $A \wedge B$ must end in the conjunction rule applied to derivations $\mathcal{D}_1$ and $\mathcal{D}_2$ of A and B, respectively. This definition can be applied as an *inversion principle* to justify both elimination rules: any canonical proof of their (common) premise $A \wedge B$ must have embedded proofs $\mathcal{D}_1$ of A and $\mathcal{D}_2$ of B, which can be used directly to derive the two rules' respective conclusions. Visualize this as a pair of reductions, showing how to eliminate either elimination rule when applied to a canonical proof:

$$\frac{\dfrac{\mathcal{D}_1 \quad \mathcal{D}_2}{\dfrac{A \quad B}{A \wedge B}}}{A} \rightsquigarrow \mathcal{D}_1 \quad A \qquad\qquad \frac{\dfrac{\mathcal{D}_1 \quad \mathcal{D}_2}{\dfrac{A \quad B}{A \wedge B}}}{B} \rightsquigarrow \mathcal{D}_2 \quad B$$

[2]Without a direct connection to Gentzen's work, though, this idea was already explored by various people in the 30s, particularly Wittgenstein ("It is what is regarded as the justification of an assertion that constitutes the sense of the assertion" (Wittgenstein, 1974, I,§40)), and Brouwer-Heyting-Kolmogorov in their interpretations of intuitionistic logic (Heyting, 1974; Kolmogorov, 1932).

Dummett extended this sort of justification to arbitrary logical laws—not just the standard elimination rules—by defining canonical proof as a hereditary notion. Canonical proofs now end in a *sequence* of introduction rules, which we recall for reference:

$$\frac{A \quad B}{A \wedge B} \qquad \frac{}{\mathsf{T}} \qquad \frac{A}{A \vee B} \quad \frac{B}{A \vee B} \qquad \text{(no rule for } \mathsf{F}\text{)} \qquad \frac{\begin{array}{c}[A]\\ \vdots \\ B\end{array}}{A \supset B}$$

Thus a canonical proof of $A \wedge B$ is defined as the conjunction rule applied to *canonical* proofs of A and B, while a canonical proof of $A \vee B$ is the appropriate injection applied to a canonical proof of A or B. There is one canonical proof of T, and no canonical proof of F. However, implication represents a base case: a canonical proof of $A \supset B$ is simply one ending in the introduction rule, the premise of which requires we show how to derive B from the assumption A by *some* means. That is, the derivation of B need not be canonical (consider that at some point we may want to apply an elimination rule on A). Atomic propositions represent another base case: the only way to prove X canonically is from an initial premise X.

Let us see how this stronger notion of canonical proof can be used to justify more complex inference rules. Consider the following "multi-step" elimination:

$$\frac{A \wedge (B_1 \vee B_2) \qquad \begin{array}{c}[A][B_1]\\ \vdots \\ C\end{array} \qquad \begin{array}{c}[A][B_2]\\ \vdots \\ C\end{array}}{C}$$

By Dummett's inversion principle, a canonical proof of the first premise contains a canonical proof $\mathcal{D}$ of A, together with a canonical proof $\mathcal{E}_i$ of B_i, for one of $i = 1$ or 2. Substituting $\mathcal{D}$ for hypothesis $[A]$ and $\mathcal{E}_i$ for $[B_i]$ in the $(i+1)$th premise of the rule, we obtain a derivation of C directly.

The verificationist meaning-theory and the upwards justification procedure are a way of vindicating Gentzen's conceptual prioritization of the logical rules. But Dummett argues that the notion that the introduction rules determine the meanings of the connectives "has no more force than the converse suggestion, that they are fixed by the elimination rules", going on to write, "The underlying idea is that the content of a statement is what you can *do* with it if you accept it.... This is, of course, the guiding idea of a pragmatist meaning-theory" [p. 280].[3]

If the elimination rules are taken as primitive, then any other inference rule may be justified by a dual, "downwards justification procedure": a rule is valid if any canonically-obtained consequence of its conclusion can be transformed into a canonically-obtained consequence of its premises. The notion of *canonically-obtained consequence* is defined analogously to canonical proof, but is somewhat complicated by the case of disjunction in natural deduction. So let us consider only the elimination rules for conjunction, truth, and implication:

$$\frac{A \wedge B}{A} \quad \frac{A \wedge B}{B} \qquad \text{(no rule for } \mathsf{T}\text{)} \qquad \frac{A \supset B \quad A}{B}$$

A canonical consequence of $A \wedge B$ applies one of the projection rules and then derives a canonical consequence from A or B; there are no canonical consequences of T; and a canonical consequence of $A \supset B$ applies modus ponens with any (not necessarily canonical) proof of A, then derives a canonical consequence from B. Again, we can validate the standard set of rules for conjunction—but this time justifying the introduction rule using the elimination rules—via the following reductions:

[3]Dummett attributes this idea to Martin-Löf, who he says "constructed an entire meaning-theory for the language of mathematics on the basis of the assumption that it is the elimination rules that determine meaning." This is likely a reference to Martin-Löf's work with Peter Hancock (Hancock and Martin-Löf, 1975), about which Martin-Löf wrote to Dummett shortly before the William James Lectures (Martin-Löf, 1976). In proof-theoretic terms, though, their realization of a pragmatist meaning-theory is different from Dummett's, since they still rely on a notion of canonical proof, albeit not granting it definitional status. Martin-Löf's 1983 Siena Lectures explicitly adopt a verificationist meaning-theory: "The meaning of a proposition is determined by... what counts as a verification of it" (Lecture 3)

$$\dfrac{\dfrac{A \quad B}{A \wedge B}}{\begin{array}{c} A \\ \mathcal{D}_1 \end{array}} \quad \rightsquigarrow \quad \begin{array}{c} A \\ \mathcal{D}_1 \end{array} \qquad \dfrac{\dfrac{A \quad B}{A \wedge B}}{\begin{array}{c} B \\ \mathcal{D}_2 \end{array}} \quad \rightsquigarrow \quad \begin{array}{c} B \\ \mathcal{D}_2 \end{array}$$

In words, any canonical consequence of (the introduction rule's conclusion) $A \wedge B$ contains a canonical consequence $\mathcal{D}_1$ of A or a canonical consequence $\mathcal{D}_2$ of B, and in either case we can obtain the consequence directly (from one of the introduction rule's premises). Again, we can also use this hereditary notion of canonical consequence to justify more complex logical laws, such as a multi-step introduction rule:

$$\dfrac{\begin{array}{cc} [A] & [A] \\ \vdots & \vdots \\ B_1 & B_2 \end{array}}{A \supset (B_1 \wedge B_2)}$$

Any canonical consequence of the conclusion applies modus ponens with a proof of A, then projects B_1 or B_2 and derives a canonical consequence. In either case, we can substitute the proof of A into one of the the rule's premises, and then derive the consequence directly.

How can these two, equally legitimate ways of understanding the connectives be related? Dummett first considers demanding "harmony between the two aspects of linguistic practice", which amounts to requiring that an accepted set of introduction and elimination rules be sound under *both* verificationist or pragmatist meaning-theories. As we saw above (albeit glossing over the case of disjunction), harmony holds for intuitionistic natural deduction. Technically, harmony is only a "modest demand", as Dummett puts it, and in addition we can ask that the rules be *complete* under either interpretation, a criterion he calls *stability.* Yet, without getting into a detailed explanation of stability, we would suggest that there is already something missing from this approach to reconciling the two meaning-theories. If we take seriously Dummett's idea that these are really *meaning*-theories, i.e., that they are two different ways of assigning meaning to the connectives, then another way of putting it is that they define *different connectives.* By coincidence, we happened to notate these with the same symbols, but we can follow the example of linear logic and write $\otimes$ for "verificationist conjunction" versus $\&$ for "pragmatist conjunction", and so forth, making it clear that there are two versions of each connective. Then rather than demanding harmony, we can simply accept diversity!

Making this politically-correct slogan mathematically-precise will be the aim of the next few sections. To get the project off the ground we will marry Dummett's analysis with the *judgmental method* (Martin-Löf, 1996; Pfenning and Davies, 2001), employing different judgments for the different kinds of proof objects manipulated above, such as canonical proofs and justified inferences. Moreover, to maintain a perfect duality between verificationist and pragmatist meaning-theories, we will carry out this project for proofs and *refutations,* rather than arbitrary inferences. Thus we define certain canonical proofs and the corresponding refutations they justify, as well as certain canonical refutations and the corresponding proofs (by contradiction) they justify. We will also explain why our adoption of linear logic notation is more than happenstance: the dichotomy between verificationist and pragmatist meaning-theories is precisely one of *polarity.* Using this insight, we will show how to unify the two approaches as fragments of a single, polarized logic, and then explain the connection between the multi-judgment natural deduction we develop for this logic and *focusing strategies* for the classical sequent calculus.

2.2 A judgmental formulation of the verificationist meaning-theory

Above, when we wrote a formula as the conclusion of a rule, we meant that the rule establishes that the formula is true. For example, the rule:

$$\dfrac{A \quad B}{A \wedge B}$$

says that if A and B are both true, then $A \wedge B$ is true. But of course the assertion that a formula is true and the formula itself are different things, and following Martin-Löf we can make this distinction explicit, e.g., by rewriting the rule as:

$$\frac{A \ \mathit{true} \quad B \ \mathit{true}}{A \wedge B \ \mathit{true}}$$

Such a step may seem pedantic when the only judgment one ever makes is A *true*, but for our task of modelling Dummett's analysis we will require a much richer judgmental palette.

We will begin by describing a formulation of the verificationist meaning-theory. Before listing its different judgments, though, we need to explain what these judgments are about: not arbitrary formulas, but "formulas through a verificationist lens", which we say have *positive polarity.* We adopt the notation $\otimes$, 1, $\oplus$, 0, and ${}^{v}\neg$ for the verificationist interpretations of conjunction, truth, disjunction, falsehood, and negation,[4] respectively. These positive connectives may be combined with positive atoms X, Y to build positive formulas P, Q:

$$P, Q ::= X \mid 1 \mid P \otimes Q \mid 0 \mid P \oplus Q \mid {}^{v}\neg P$$

As we explained, the two central concepts of a verificationist meaning-theory are canonical proof and "upwards" justified inference. We will shift terminology a bit from Dummett and refer to the former as *direct* proofs, writing P *triv* for the judgment that P has a direct proof (and so is not only true but "trivial").[5] We write P *false* for the judgment that P is "merely" refutable, i.e., by some justified, but not necessarily direct, train of reasoning.

Rather than immediately giving proof and refutation rules for each connective, we will first axiomatize the inversion principles associated with Dummett's hereditary definition of direct ("canonical") proof. For example, we know that any direct proof of $X \otimes (Y \oplus {}^{v}\neg Q)$ has one of the following two forms:

$$\frac{X \ \mathit{triv} \quad \dfrac{Y \ \mathit{triv}}{Y \oplus {}^{v}\neg Q \ \mathit{triv}}}{X \otimes (Y \oplus {}^{v}\neg Q) \ \mathit{triv}} \qquad \frac{X \ \mathit{triv} \quad \dfrac{\dfrac{Q \ \mathit{false}}{{}^{v}\neg Q \ \mathit{triv}}}{Y \oplus {}^{v}\neg Q \ \mathit{triv}}}{X \otimes (Y \oplus {}^{v}\neg Q) \ \mathit{triv}}$$

In general, any direct proof of P ends in a series of introduction rules, with a fringe of *primitive* premises of the form X *triv* or Q *false*. We call such a list of premises Δ a *linear context,* and write this relationship as $\Delta \Rightarrow P$ *triv*. As shown above, for $P = X \otimes (Y \oplus {}^{v}\neg Q)$ we have $\Delta_1 \Rightarrow P$ *triv* and $\Delta_2 \Rightarrow P$ *triv*, where $\Delta_1 = (X \ \mathit{triv}, Y \ \mathit{triv})$ and $\Delta_2 = (X \ \mathit{triv}, Q \ \mathit{false})$. The intuition is that the set $\{\Delta \mid \Delta \Rightarrow P \ \mathit{triv}\}$ represents all possible decompositions of a direct proof of P, i.e., an inversion principle.

For arbitrary positive formulas, $\Delta \Rightarrow P$ *triv* is axiomatized as follows:

$$\frac{}{X \ \mathit{triv} \Rightarrow X \ \mathit{triv}} \qquad \frac{}{P \ \mathit{false} \Rightarrow {}^{v}\neg P \ \mathit{triv}}$$

$$\frac{}{\cdot \Rightarrow 1 \ \mathit{triv}} \qquad \frac{\Delta_1 \Rightarrow P \ \mathit{triv} \quad \Delta_2 \Rightarrow Q \ \mathit{triv}}{\Delta_1, \Delta_2 \Rightarrow P \otimes Q \ \mathit{triv}}$$

$$\text{(no rule for 0)} \qquad \frac{\Delta \Rightarrow P \ \mathit{triv}}{\Delta \Rightarrow P \oplus Q \ \mathit{triv}} \quad \frac{\Delta \Rightarrow Q \ \mathit{triv}}{\Delta \Rightarrow P \oplus Q \ \mathit{triv}}$$

We can make a few easy observations about $\Delta \Rightarrow P$ *triv*.

Proposition 1 (Subformula property)**.** *(i) if $\Delta \Rightarrow P$ triv, then every formula occurring in Δ is a subformula[6] of P, and moreover (ii) if Q false $\in \Delta$, then Q is a proper subformula of P.*

[4] The notation ${}^{v}\neg$ not only marks this negation as *v*erificationist, but also (foreshadowing) as call-by-*v*alue.

[5] We want to consider direct proofs in a context of assumptions, where it no longer makes sense to call them "canonical" because the Fundamental Assumption fails.

[6] Defined in the standard way, i.e., A is a subformula of B if $A = B$, or if B applies some n-ary connective to formulas $B_1, \ldots, B_n$, and A is a subformula of B_i. Excluding the condition $A = B$ makes A a *proper* subformula of B.

Proposition 2 (Finite support)**.** *For any P, there exist only finitely many Δ such that $\Delta \Rightarrow P\ \mathit{triv}$.*

The relation $\Delta \Rightarrow P\ \mathit{triv}$ describes "necessary components" for a direct proof of P. To actually construct a direct proof of P, we must provide each of these components. This can be expressed as a generic rule for concluding $P\ \mathit{triv}$:

$$\frac{\Delta \Rightarrow P\ \mathit{triv} \quad \Delta}{P\ \mathit{triv}}$$

In general, the proofs of the premises Δ may refer to assumptions, and to make this explicit we use *hypothetical judgments* indexed by a context Γ, rewriting the rule as follows:

$$\frac{\Delta \Rightarrow P\ \mathit{triv} \quad \Gamma \vdash \Delta}{\Gamma \vdash P\ \mathit{triv}}$$

A context Γ is simply a list of linear contexts (we use $\cdot$ for the empty list):

$$\Gamma ::= \cdot \mid \Gamma, \Delta$$

For some primitive hypothesis H (i.e., $X\ \mathit{triv}$ or $P\ \mathit{false}$), we write $H \in \Gamma$ if there exists some $\Delta \in \Gamma$ such that $H \in \Delta$. Now Γ satisfies the premises Δ, written $\Gamma \vdash \Delta$, when it can discharge all atomic premises $X\ \mathit{triv} \in \Delta$, and can provide justified refutations for premises $P\ \mathit{false} \in \Delta$, as expressed by the following rules:

$$\frac{}{\Gamma \vdash \cdot} \qquad \frac{X\ \mathit{triv} \in \Gamma \quad \Gamma \vdash \Delta}{\Gamma \vdash X\ \mathit{triv}, \Delta} \qquad \frac{\Gamma \vdash P\ \mathit{false} \quad \Gamma \vdash \Delta}{\Gamma \vdash P\ \mathit{false}, \Delta}$$

Example 1. The reader can verify that by instantiating P with $X \otimes (Y \oplus {}^{v}\neg Q)$ and working out these definitions, we derive two rules for concluding $\Gamma \vdash X \otimes (Y \oplus {}^{v}\neg Q)\ \mathit{triv}$:

$$\frac{X\ \mathit{triv} \in \Gamma \quad Y\ \mathit{triv} \in \Gamma}{\Gamma \vdash X \otimes (Y \oplus {}^{v}\neg Q)\ \mathit{triv}} \qquad \frac{X\ \mathit{triv} \in \Gamma \quad \Gamma \vdash Q\ \mathit{false}}{\Gamma \vdash X \otimes (Y \oplus {}^{v}\neg Q)\ \mathit{triv}}$$

■

We can similarly give a generic rule for establishing $P\ \mathit{false}$. Intuitively, a justified refutation of P shows how to derive a contradiction from the assumption that we have a direct proof of P. Letting *contra* stand for a separate contradiction judgment, we obtain the following rule:

$$\frac{\forall(\Delta \Rightarrow P\ \mathit{triv}) : \quad \Gamma, \Delta \vdash \mathit{contra}}{\Gamma \vdash P\ \mathit{false}}$$

The form of this rule may appear somewhat unusual, quantifying over derivations $\Delta \Rightarrow P\ \mathit{triv}$. Due to the finite support property for the connectives we are considering, the rule will always simply have a finite list of premises. More abstractly, though, the rule for concluding $P\ \mathit{false}$ may be seen as an *iterated inductive definition* (Martin-Löf, 1971; Buchholz et al., 1981), applying induction on the previously defined relation $\Delta \Rightarrow P\ \mathit{triv}$. Interpreted constructively, the rule demands a *map* from derivations $\Delta \Rightarrow P\ \mathit{triv}$ to derivations $\Gamma, \Delta \vdash \mathit{contra}$. We will exploit this higher-order reading when proving results about the system, and later on in developing the Curry-Howard interpretation.

Example 2. In the case $P = X \otimes (Y \oplus {}^{v}\neg Q)$, we derive the following rule:

$$\frac{\Gamma, X\ \mathit{triv}, Y\ \mathit{triv} \vdash \mathit{contra} \quad \Gamma, X\ \mathit{triv}, Q\ \mathit{false} \vdash \mathit{contra}}{\Gamma \vdash X \otimes (Y \oplus {}^{v}\neg Q)\ \mathit{false}}$$

■

Finally, we describe how to establish $\Gamma \vdash \mathit{contra}$. We have a contradiction if we assumed that P is false, yet can give P a direct proof:

$$\frac{P\ \mathit{false} \in \Gamma \quad \Gamma \vdash P\ \mathit{triv}}{\Gamma \vdash \mathit{contra}}$$

Example 3. Let $\Delta = (X\ \mathit{triv}, X\ \mathit{false})$. The following derivation proves the law of non-contradiction:

Positive formulas $P, Q ::= X \mid 1 \mid P \otimes Q \mid 0 \mid P \oplus Q \mid {}^{v}\neg P$
Linear contexts $\Delta ::= \cdot \mid X\ triv, \Delta \mid P\ false, \Delta$

$\boxed{\Delta \Rightarrow P\ triv}$

$$\frac{}{X\ triv \Rightarrow X\ triv} \qquad \frac{}{P\ false \Rightarrow {}^{v}\neg P\ triv}$$

$$\frac{}{\cdot \Rightarrow 1\ triv} \qquad \frac{\Delta_1 \Rightarrow P\ triv \quad \Delta_2 \Rightarrow Q\ triv}{\Delta_1, \Delta_2 \Rightarrow P \otimes Q\ triv}$$

(no rule for 0) $\qquad \dfrac{\Delta \Rightarrow P\ triv}{\Delta \Rightarrow P \oplus Q\ triv} \quad \dfrac{\Delta \Rightarrow Q\ triv}{\Delta \Rightarrow P \oplus Q\ triv}$

Contexts $\Gamma ::= \cdot \mid \Gamma, \Delta$

$\boxed{\Gamma \vdash P\ triv}$

$$\frac{\Delta \Rightarrow P\ triv \quad \Gamma \vdash \Delta}{\Gamma \vdash P\ triv}$$

$\boxed{\Gamma \vdash P\ false}$

$$\frac{\forall(\Delta \Rightarrow P\ triv) : \quad \Gamma, \Delta \vdash contra}{\Gamma \vdash P\ false}$$

$\boxed{\Gamma \vdash \Delta}$

$$\frac{}{\Gamma \vdash \cdot} \qquad \frac{X\ triv \in \Gamma \quad \Gamma \vdash \Delta}{\Gamma \vdash X\ triv, \Delta} \qquad \frac{\Gamma \vdash P\ false \quad \Gamma \vdash \Delta}{\Gamma \vdash P\ false, \Delta}$$

$\boxed{\Gamma \vdash contra}$

$$\frac{P\ false \in \Gamma \quad \Gamma \vdash P\ triv}{\Gamma \vdash contra}$$

Figure 1: The positive interpretation

$$\dfrac{\dfrac{X\ false \in \Delta \qquad \dfrac{X\ triv \in \Delta}{\Delta \vdash X\ triv}}{\Delta \vdash contra}}{\dfrac{\vdash X \otimes {}^{v}\neg X\ false}{\vdash {}^{v}\neg(X \otimes {}^{v}\neg X)\ triv}}$$

■

This completes our formalization of the verificationist meaning-theory, which we also call the *positive interpretation* of the connectives. The foregoing development is summarized in Figure 1. What more can we say about it?

Well, we explained the sense in which (as Gentzen/Prawitz/Dummett proposed) the introduction rules of natural deduction "define" the connectives through a verificationist meaning-theory. In the presentation here, the rules for $\Delta \Rightarrow P\ triv$ literally define the connectives, in the sense that they are the only rules that mention them! One consequence of this is that we can prove certain properties about the system generically, without having to reason about particular connectives. In particular, two properties that vouchsafe the sanity of the logic are "identity" and "reduction".

Principle 3 (Identity)**.** If $P\ false \in \Gamma$ then $\Gamma \vdash P\ false$.

Principle 4 (Reduction)**.** If $\Gamma \vdash P\ false$ and $\Gamma \vdash P\ triv$ then $\Gamma \vdash contra$.

To prove the identity principle, we must simultaneously prove another "context identity" principle:

Principle 5 (Context identity)**.** $\Gamma, \Delta \vdash \Delta$

Proof (of identity and context identity). We first give the mutually-recursive derivations of these identity principles, and then explain what makes this definition well-founded. The following derivation reduces identity to context identity:

$$\dfrac{\forall(\Delta \Rightarrow P\ \mathit{triv}) : \dfrac{P\ \mathit{false} \in \Gamma \qquad \dfrac{\Delta \Rightarrow P\ \mathit{triv} \quad \Gamma, \Delta \vdash \Delta}{\Gamma, \Delta \vdash P\ \mathit{triv}}}{\Gamma, \Delta \vdash \mathit{contra}}}{\Gamma \vdash P\ \mathit{false}}$$

Both premises $P\ \mathit{false} \in \Gamma$ and $\Delta \Rightarrow P\ \mathit{triv}$ may be discharged, and we are left with the premise $\Gamma, \Delta \vdash \Delta$.

To prove the context identity principle, we first state a trivial lemma that expands the meaning of $\Gamma \vdash \Delta$:

Lemma 6 (Expansion)**.** *$\Gamma \vdash \Delta$ if and only if for all $X\ \mathit{triv} \in \Delta$ we have $X\ \mathit{triv} \in \Gamma$, and for all $Q\ \mathit{false} \in \Delta$ we have $\Gamma \vdash Q\ \mathit{false}$.*

Then $\Gamma, \Delta \vdash \Delta$, since for all $X\ \mathit{triv} \in \Delta$ we have $X\ \mathit{triv} \in \Gamma, \Delta$ by definition, and for all $Q\ \mathit{false} \in \Delta$ we have $\Gamma, \Delta \vdash Q\ \mathit{false}$ by identity.

Now, the proof of identity on P appealed to context identity on Δ for $\Delta \Rightarrow P\ \mathit{triv}$. In turn, context identity on Δ appealed back to identity on Q for $Q\ \mathit{false} \in \Delta$. By subformula property (ii), Q must be a proper subformula of P, which makes these derivations well-founded. □

We likewise prove reduction simultaneously with another principle of substitution, which we state generically for any conclusion J (i.e., $P\ \mathit{triv}$, $P\ \mathit{false}$, Δ, or contra).

Principle 7 (Substitution)**.** If $\Gamma, \Delta \vdash J$ and $\Gamma \vdash \Delta$ then $\Gamma \vdash J$.

Proof (of reduction and substitution). Again we first give a proof making free use of mutual self-reference, and then show that it is well-founded.

Reduction immediately reduces to substitution: by inversion on the derivation of $\Gamma \vdash P\ \mathit{triv}$, there must exist some $\Delta \Rightarrow P\ \mathit{triv}$ such that $\Gamma \vdash \Delta$, and by inversion on $\Gamma \vdash P\ \mathit{false}$, we have $\Gamma, \Delta \vdash \mathit{contra}$. Hence $\Gamma \vdash \mathit{contra}$ by substitution.

The proof of substitution uses a side-induction on the derivation of $\Gamma, \Delta \vdash J$. Almost every case (there are six, corresponding to each rule in the bottom half of Figure 1) follows immediately by applying substitution to the premises. The only interesting case is when contra is derived using some hypothesis $Q\ \mathit{false} \in \Delta$, as in the following:

$$\dfrac{Q\ \mathit{false} \in \Delta \quad \Gamma, \Delta \vdash Q\ \mathit{triv}}{\Gamma, \Delta \vdash \mathit{contra}}$$

By the side-induction on the second premise, we have $\Gamma \vdash Q\ \mathit{triv}$. By the expansion lemma applied to assumptions $\Gamma \vdash \Delta$ and $Q\ \mathit{false} \in \Delta$, we have $\Gamma \vdash Q\ \mathit{false}$. Hence $\Gamma \vdash \mathit{contra}$ by reduction.

The well-foundedness argument is the same here as for the identity principles. Namely, the proof of reduction on P appealed to substitution on Δ such that $\Delta \Rightarrow P\ \mathit{false}$, which in turn appealed back to reduction on Q for $Q\ \mathit{false} \in \Delta$, and thus Q must be a proper subformula of P. □

In terms of provability, there is a simple relationship between the formalism we have defined and ordinary intuitionistic logic. Let $|P|$ be the operator that converts a positive formula to an ordinary formula:

$$|X| = X \quad |{}^{v}\neg P| = \neg|P|$$
$$|1| = \mathsf{T} \quad |P \otimes Q| = |P| \wedge |Q| \quad |0| = \mathsf{F} \quad |P \oplus Q| = |P| \vee |Q|$$

We extend this operator to judgments as follows:

$$|P\ \mathit{triv}| = |P| \quad |P\ \mathit{false}| = \neg|P| \quad |\mathit{contra}| = \mathsf{F} \quad |(J_1, \ldots, J_n)| = |J_1| \wedge \ldots \wedge |J_n|$$

And to contexts Γ pointwise. Then we can state the fact that Figure 1 soundly interprets intuitionistic logic:

Proposition 8 (Intuitionistic soundness)**.** *If* $\Gamma \vdash J$ *then* $|\Gamma| \vdash |J|$ *is intuitionistically provable.*

We will not give the proof of this fact, but it should be intuitive, given how we described the different judgments. For example with $J = P\ triv$, soundness says that if a formula has a direct proof, then it has an intuitionistic proof. On the other hand completeness fails, because in some context of assumptions, a formula may be intuitionistically provable without having a direct proof. For example, $X, \neg X \vdash \mathsf{F}$ is intuitionistically provable, but we cannot derive $X\ triv, X\ false \vdash 0\ triv$ (although we can derive $X\ triv, X\ false \vdash contra$). However, completeness does hold for intuitionistic theorems, i.e., if $\vdash |J|$ is intuitionistically provable then $\vdash J$.

In any case, all of these remarks about provability are simply to give the reader a better feel for the formalism. The point of our analysis, after all, is to refine the structure of *proofs,* for which the dividends will appear only later on.

2.3 A dual formulation

We have seen how to formalize the verificationist meaning-theory through the judgmental method, as well as how to prove a few basic results about the system. Now we can obtain a pragmatist interpretation by an almost mechanical dualization. This interpretation is centered on two judgments, $N\ absurd$ and $N\ true$, stating, respectively, that N has a direct refutation (and hence is not only false but "absurd"), or that it (merely) has a justified proof. Here N is a *negative polarity* formula, constructed using the negative connectives $\&$, $\top$, $\parr$, $\bot$, and ${}^{n}\!\neg$ (which are the pragmatist interpretations of conjunction, truth, disjunction, falsehood, and negation,[7] respectively), together with negative atoms $\overline{X}, \overline{Y}$:

$$N, M ::= \overline{X} \mid \top \mid N \& M \mid \bot \mid N \parr M \mid {}^{n}\!\neg N$$

Again, we begin by axiomatizing the inversion properties of direct refutations as a relation $\Delta \Rightarrow N\ absurd$.

$$\frac{}{\overline{X}\ absurd \Rightarrow \overline{X}\ absurd} \qquad \frac{}{N\ true \Rightarrow {}^{n}\!\neg N\ absurd}$$

$$\text{(no rule for } \top) \qquad \frac{\Delta \Rightarrow N\ absurd}{\Delta \Rightarrow N \& M\ absurd} \quad \frac{\Delta \Rightarrow M\ absurd}{\Delta \Rightarrow N \& M\ absurd}$$

$$\frac{}{\cdot \Rightarrow \bot\ absurd} \qquad \frac{\Delta_1 \Rightarrow N\ absurd \quad \Delta_2 \Rightarrow M\ absurd}{\Delta_1, \Delta_2 \Rightarrow N \parr M\ absurd}$$

In prose, we can gloss these rules as follows: a direct refutation of an atomic proposition must be by assumption, while a direct refutation of ${}^{n}\!\neg N$ is just a justified proof of N; a direct refutation of $N \& M$ is a direct refutation of N or of M, while there are no direct refutations of $\top$; a direct refutation of $N \parr M$ is a direct refutation both of N and of M, while there is only one, trivial direct refutation of $\bot$.

Using this relation, we can give a general rule for concluding $N\ absurd$ under some context of assumptions Γ:

$$\frac{\Delta \Rightarrow N\ absurd \quad \Gamma \vdash \Delta}{\Gamma \vdash N\ absurd}$$

where $\Gamma \vdash \Delta$ is defined analogously to before:

$$\frac{}{\Gamma \vdash \cdot} \qquad \frac{\overline{X}\ absurd \in \Gamma \quad \Gamma \vdash \Delta}{\Gamma \vdash \overline{X}\ absurd, \Delta} \qquad \frac{\Gamma \vdash N\ true \quad \Gamma \vdash \Delta}{\Gamma \vdash N\ true, \Delta}$$

Example 4. For $N = \overline{X} \& (\overline{Y} \parr {}^{n}\!\neg M)$ we derive the following rules:

$$\frac{\overline{X}\ absurd \in \Gamma}{\Gamma \vdash \overline{X} \& (\overline{Y} \parr {}^{n}\!\neg M)\ absurd} \qquad \frac{\overline{Y}\ absurd \in \Gamma \quad \Gamma \vdash M\ true}{\Gamma \vdash \overline{X} \& (\overline{Y} \parr {}^{n}\!\neg M)\ absurd}$$

■

[7] The notation ${}^{n}\!\neg$ not only marks this negation as *n*egative, but also (foreshadowing) as call-by-*n*ame.

The notion of justified proof is defined as follows: N is justified if we can derive a contradiction from any direct refutation of N.

$$\frac{\forall(\Delta \Rightarrow N\ \mathit{absurd}):\quad \Gamma, \Delta \vdash \mathit{contra}}{\Gamma \vdash N\ \mathit{true}}$$

EXAMPLE 5. For $N = \overline{X}\&(\overline{Y} \parr^{n} \neg M)$ we have:

$$\frac{\Gamma, \overline{X}\ \mathit{absurd} \vdash \mathit{contra} \quad \Gamma, \overline{Y}\ \mathit{absurd}, M\ \mathit{true} \vdash \mathit{contra}}{\Gamma \vdash \overline{X}\&(\overline{Y} \parr^{n} \neg M)\ \mathit{true}}$$

■

Finally, to establish a contradiction from Γ, we must find some hypothesis $N\ \mathit{true} \in \Gamma$, and show that N is false by giving a direct refutation:

$$\frac{N\ \mathit{true} \in \Gamma \quad \Gamma \vdash N\ \mathit{absurd}}{\Gamma \vdash \mathit{contra}}$$

EXAMPLE 6. Let $\Delta = (\overline{X}\ \mathit{absurd}, \overline{X}\ \mathit{true})$. We prove the law of excluded middle as follows:

$$\dfrac{\dfrac{\overline{X}\ \mathit{true} \in \Delta \quad \dfrac{\overline{X}\ \mathit{absurd} \in \Delta}{\Delta \vdash \overline{X}\ \mathit{absurd}}}{\Delta \vdash \mathit{contra}}}{\vdash \overline{X} \parr^{n} \neg \overline{X}\ \mathit{true}}$$

■

A summary of all these rules is given in Figure 2. From Example 6, it is clear that the judgment $N\ \mathit{true}$ does not correspond to intuitionistic truth. In fact, it corresponds to what Czermak has called "dual intuitionistic" and Goodman "anti-intuitionistic" truth (Czermak, 1977; Goodman, 1981). Conversely, the judgment $N\ \mathit{absurd}$ is more stringent than intuitionistic falsehood, roughly corresponding to what Nelson calls "constructible falsity" (Nelson, 1949). Formally, we can state a simple duality principle between the positive and negative interpretations. We first define $(-)^{\perp}$ as an operator taking positive formulas to negative, and vice versa:[8]

$$\begin{array}{rrrr}
X^{\perp} = \overline{X} & & \overline{X}^{\perp} = X & \\
1^{\perp} = \bot & 0^{\perp} = \top & \top^{\perp} = 0 & \bot^{\perp} = 1 \\
(P \otimes Q)^{\perp} = P^{\perp} \parr Q^{\perp} & & (P \oplus Q)^{\perp} = P^{\perp}\&Q^{\perp} & \\
(N\&M)^{\perp} = N^{\perp} \oplus M^{\perp} & & (N \parr M)^{\perp} = N^{\perp} \otimes M^{\perp} & \\
(\neg^{v} P)^{\perp} = \neg^{n} P^{\perp} & & (\neg^{n} N)^{\perp} = \neg^{v} N^{\perp} &
\end{array}$$

We extend it to judgments with $(P\ \mathit{triv})^{\perp} = P^{\perp}\ \mathit{absurd}$, $(P\ \mathit{false})^{\perp} = P^{\perp}\ \mathit{true}$, $(N\ \mathit{absurd})^{\perp} = N^{\perp}\ \mathit{triv}$, $(N\ \mathit{true})^{\perp} = N^{\perp}\ \mathit{false}$, and $\mathit{contra}^{\perp} = \mathit{contra}$, and finally we extend it to contexts pointwise. Then the following observation is immediate:

Principle 9 (Duality)**.** $\Gamma \vdash J$ iff $\Gamma^{\perp} \vdash J^{\perp}$

Note also that $P^{\perp\perp} = P$ and $N^{\perp\perp} = N$. An immediate corollary is that the negative interpretation satisfies identity and reduction principles analogous to the ones for the positive interpretation.

Principle 10 (Identity)**.** If $N\ \mathit{true} \in \Gamma$ then $\Gamma \vdash N\ \mathit{true}$.

Principle 11 (Reduction)**.** If $\Gamma \vdash N\ \mathit{true}$ and $\Gamma \vdash N\ \mathit{absurd}$ then $\Gamma \vdash \mathit{contra}$.

[8] A technical remark on the atomic case: abstractly, dualization must assign to each positive atom an associated "dual" negative atom, and vice versa. We achieve this with the syntactic trick of adding and removing an overline.

Negative formulas $N, M \;::=\; \overline{X} \mid \top \mid N \& M \mid \bot \mid N \parr M \mid {}^{n}\!\neg N$

Linear contexts $\Delta \;::=\; \cdot \mid \overline{X}\ \mathit{absurd}, \Delta \mid N\ \mathit{true}, \Delta$

$\boxed{\Delta \Rightarrow N\ \mathit{absurd}}$

$$\frac{}{\overline{X}\ \mathit{absurd} \Rightarrow \overline{X}\ \mathit{absurd}} \qquad \frac{}{N\ \mathit{true} \Rightarrow {}^{n}\!\neg N\ \mathit{absurd}}$$

$$\text{(no rule for } \top\text{)} \qquad \frac{\Delta \Rightarrow N\ \mathit{absurd}}{\Delta \Rightarrow N \& M\ \mathit{absurd}} \quad \frac{\Delta \Rightarrow M\ \mathit{absurd}}{\Delta \Rightarrow N \& M\ \mathit{absurd}}$$

$$\frac{}{\cdot \Rightarrow \bot\ \mathit{absurd}} \qquad \frac{\Delta_1 \Rightarrow N\ \mathit{absurd} \quad \Delta_2 \Rightarrow M\ \mathit{absurd}}{\Delta_1, \Delta_2 \Rightarrow N \parr M\ \mathit{absurd}}$$

. . . . . .

Contexts $\Gamma \;::=\; \cdot \mid \Gamma, \Delta$

$\boxed{\Gamma \vdash N\ \mathit{absurd}}$

$$\frac{\Delta \Rightarrow N\ \mathit{absurd} \quad \Gamma \vdash \Delta}{\Gamma \vdash N\ \mathit{absurd}}$$

$\boxed{\Gamma \vdash N\ \mathit{true}}$

$$\frac{\forall (\Delta \Rightarrow N\ \mathit{absurd}) : \quad \Gamma, \Delta \vdash \mathit{contra}}{\Gamma \vdash N\ \mathit{true}}$$

$\boxed{\Gamma \vdash \Delta}$

$$\frac{}{\Gamma \vdash \cdot} \qquad \frac{\overline{X}\ \mathit{absurd} \in \Gamma \quad \Gamma \vdash \Delta}{\Gamma \vdash \overline{X}\ \mathit{absurd}, \Delta} \qquad \frac{\Gamma \vdash N\ \mathit{true} \quad \Gamma \vdash \Delta}{\Gamma \vdash N\ \mathit{true}, \Delta}$$

$\boxed{\Gamma \vdash \mathit{contra}}$

$$\frac{N\ \mathit{true} \in \Gamma \quad \Gamma \vdash N\ \mathit{absurd}}{\Gamma \vdash \mathit{contra}}$$

Figure 2: The negative interpretation

2.4 The unity of duality

Given the formal duality between verificationist and pragmatist interpretations, it seems that a choice of one over the other has, to use Dummett's expression, "no more force than the converse suggestion." Yet, as we have stressed, the two interpretations really define *different* connectives. Rather than making a choice between them, then, why not just combine the two interpretations?

The easiest way of doing this is simply to take the union of the two systems, letting contexts contain a mix of primitive hypotheses X *triv*, P *false*, $\overline{X}$ *absurd*, N *true*, and allowing inference by any of the rules in Figures 1 and 2. A moment's reflection verifies that the combined system is coherent, in particular that the proofs of the various identity, reduction, and substitution principles in Sections 2.2 and 2.3 remain essentially unchanged. Another moment's reflection, however, reveals that this combined system is not very interesting, because of the syntactic separation between positive and negative formulas.

To break this impasse we add a pair of mediating connectives:

$$\begin{array}{lll} P, Q & ::= & \cdots \mid \downarrow N \\ N, M & ::= & \cdots \mid \uparrow P \end{array}$$

The notation is borrowed from Girard (2001), who calls these the "shift" connectives. In terms of our judgmental analysis, the meaning of these connectives may be explained as modalities: $\downarrow$ embeds the weaker (i.e., more permissive) notion of justified proof as a modality of direct proof, while $\uparrow$ embeds the weaker notion of justified refutation as a modality of direct refutation. Formally, we represent this meaning for the shifts with a pair of inversion rules:

$$\frac{}{N \ \mathit{true} \Rightarrow \downarrow N \ \mathit{triv}} \qquad \frac{}{P \ \mathit{false} \Rightarrow \uparrow P \ \mathit{absurd}}$$

The resulting unified logic, which we call *polarized logic,* now allows verificationist and pragmatist interpretations to be related in non-trivial ways, beyond formal duality. Let us first give some intuition for the shift connectives with a few examples.

Example 7. On the left we give the derived rule for concluding $\downarrow N$ *triv*, while on the right we show *one* way of deriving $\uparrow P$ *true*:

$$\frac{\Gamma \vdash N \ \mathit{true}}{\Gamma \vdash \downarrow N \ \mathit{triv}} \qquad \frac{\dfrac{\Gamma \vdash P \ \mathit{triv}}{\Gamma, P \ \mathit{false} \vdash \mathit{contra}}}{\Gamma \vdash \uparrow P \ \mathit{true}}$$

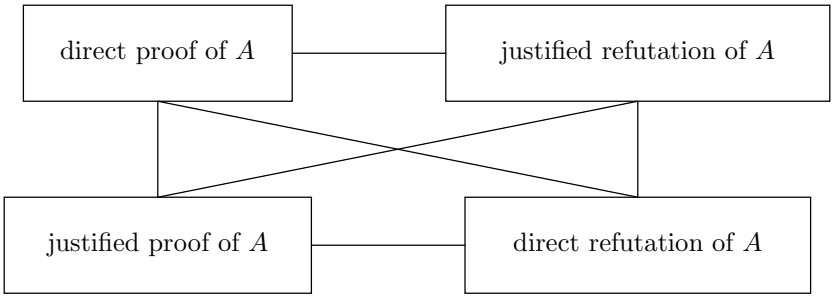
The double line in the second derivation represents the following line of inference: weaken $\Gamma \vdash P$ *triv* to Γ, P *false* $\vdash P$ *triv*, then derive Γ, P *false* $\vdash$ *contra* by applying the hypothesis P *false*. ■

Example 8. Let $\Delta = (P \ \mathit{false}, {}^{v}\neg P \ \mathit{false})$. Starting from $\Delta \vdash P$ *false* (by the identity principle), we derive $\vdash \uparrow P \otimes \uparrow {}^{v}\neg P$ *true*.

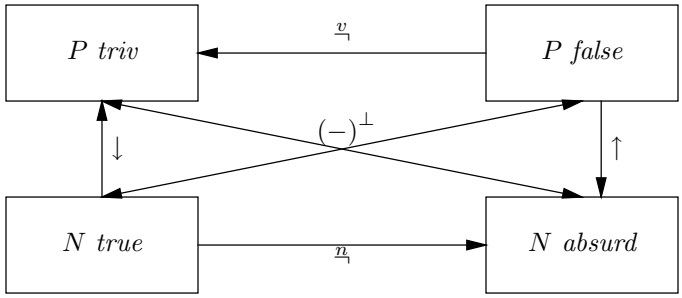
$$\frac{{}^{v}\neg P \ \mathit{false} \in \Delta \qquad \dfrac{\Delta \vdash P \ \mathit{false}}{\Delta \vdash {}^{v}\neg P \ \mathit{triv}}}{\dfrac{\Delta \vdash \mathit{contra}}{\vdash \uparrow P \otimes \uparrow {}^{v}\neg P \ \mathit{true}}}$$

■

In the Introduction, we alluded to the idea that Dummett's analysis represents a "square of opposition" between different kinds of inference:



Using polarized logic, we can replace the boxes with polarized judgments, and annotate the edges with connectives:



The diagram "commutes", in the following sense.

Definition 12. For two positive formulas P, Q, we say that $P \leq Q$ if for all $\Delta \Rightarrow P$ *triv*, there exists $\Delta' \Rightarrow Q$ *triv* such that $\Delta \vdash \Delta'$. For two negative formulas we say $M \leq N$ if for all $\Delta \Rightarrow N$ *absurd*, there exists $\Delta' \Rightarrow M$ *absurd* such that $\Delta \vdash \Delta'$. We write $P \equiv Q$ if $P \leq Q$ and $Q \leq P$, and $M \equiv N$ if $M \leq N$ and $N \leq M$.

Proposition 13. *$\leq$ is a partial order (and hence $\equiv$ an equivalence relation).*

Proof. Reflexivity holds by context identity, and transitivity by substitution. □

Proposition 14. *Let P, Q, R be positive formulas such that $P \leq Q$. Let L, M, N be negative formulas such that $M \leq N$. Then the following hold:*

1. $P \otimes R \leq Q \otimes R$ *and* $R \otimes P \leq R \otimes Q$
2. $P \oplus R \leq Q \oplus R$ *and* $R \oplus P \leq R \oplus Q$
3. $Q^{\perp} \leq P^{\perp}$
4. $\overset{v}{\neg} Q \leq \overset{v}{\neg} P$
5. $\uparrow P \leq \uparrow Q$
6. $L \& M \leq L \& N$ *and* $M \& L \leq N \& L$
7. $L \parr M \leq L \parr N$ *and* $M \parr L \leq N \parr L$
8. $\overset{n}{\neg} N \leq \overset{n}{\neg} M$

9. $N^\perp \leq M^\perp$

10. $\downarrow M \leq \downarrow N$

Proof. (1–3) are immediate. (4) and (5) are both by the following derivation:

$$\dfrac{\forall(\Delta \Rightarrow P\ \mathit{triv}) : \quad \dfrac{\dfrac{\Delta' \Rightarrow Q\ \mathit{triv} \quad \Delta \vdash \Delta'}{\mathit{Q\ false}, \Delta \vdash \mathit{contra}}}{\mathit{Q\ false} \vdash P\ \mathit{false}}$$

where the double line represents the following line of inference: we have $\Delta' \Rightarrow Q\ \mathit{triv}$ and $\Delta \vdash \Delta'$ (for some Δ') by the assumption $P \leq Q$, and hence $\Delta \vdash Q\ \mathit{triv}$, which we can weaken to $Q\ \mathit{false}, \Delta \vdash Q\ \mathit{triv}$, and obtain $Q\ \mathit{false}, \Delta \vdash \mathit{contra}$ by applying $Q\ \mathit{false}$. (6–10) are dual to (1–5). □

Now we can relate the shifts, negations, and the dualizing operator. Note that the definition of $(-)^\perp$ extends to the shifts by $(\downarrow N)^\perp = \uparrow N^\perp$, $(\uparrow P)^\perp = \downarrow P^\perp$.[9]

Proposition 15. *The following equivalences hold:*

1. $\overset{v}{\neg} P \equiv \downarrow P^\perp$

2. $\overset{n}{\neg} N \equiv \uparrow N^\perp$

3. $\downarrow\uparrow P \equiv \overset{v}{\neg}\overset{v}{\neg} P$

4. $\uparrow\downarrow N \equiv \overset{n}{\neg}\overset{n}{\neg} N$

Proof. (1) and (2) are immediate. For (3), we have $\overset{v}{\neg}\overset{v}{\neg} P \equiv \Downarrow(\overset{v}{\neg} P)^\perp \equiv \Downarrow(\downarrow P^\perp)^\perp = \downarrow\uparrow P^{\perp\perp} = \downarrow\uparrow P$, and similarly for (4). □

Hopefully we have at least somewhat demystified the meaning of Girard's shift connectives. After we develop the Curry-Howard interpretation of polarized logic, the shifts will become even more concrete. There, we will see that they model control effects: $\downarrow N$ is the type of a *suspended* expression of type N, while $\uparrow P$ the type of a *captured* continuation accepting type P. But we hold off on defining the programming language for just a while longer, in order to explain the relationship of this unified logic of proofs and refutations to Andreoli's notion of "focusing" strategies for sequent calculi.

2.5 Focusing the classical sequent calculus

The technique of *focusing* was originally invented as a way of guiding bottom-up proof search in linear logic (Andreoli, 1992). In general, bottom-up search works by reading each rule as a suggestion, "To prove the conclusion, try proving the premises." Starting from a goal sequent, one looks for a proof by provisionally invoking rules to obtain a new set of subgoals until, hopefully, all goals can be discharged. Unfortunately, at each stage of the search there are often multiple rules that could be applied to the goal, and a priori we have no way of knowing which will lead to a successful proof. Therefore we must try many different interleavings of rules, creating a combinatorial explosion of the search space, which for linear logic turns out to be intractable.

Andreoli's observation was two-fold. First, about half of the rules of linear logic are *invertible,* i.e., their conclusion implies their premise. In particular, the right-rules for the negative connectives $\&, \parr, \top, \perp$ are

[9] Rather than defining $(-)^\perp$ as an operator on formulas and judgments, we can alternatively define it as a first-class connective by adding inversion rules:

$$\dfrac{\Delta \Rightarrow P\ \mathit{triv}}{\Delta \Rightarrow P^\perp\ \mathit{absurd}} \qquad \dfrac{\Delta \Rightarrow N\ \mathit{absurd}}{\Delta \Rightarrow N^\perp\ \mathit{triv}}$$

Then $(\downarrow N)^\perp \equiv \uparrow N^\perp$ and $(\uparrow P)^\perp \equiv \downarrow P^\perp$ are equivalences rather than equalities.

invertible, as are the left-rules for the positive connectives $\otimes, \oplus, 1, 0$.[10] These rules can be applied greedily, in an arbitrary order, preserving the provability of the goal. Second, and less intuitively, one can also chain the application of non-invertible rules by "focusing" on some formula and applying a sequence of non-invertible rules. In particular, while a formula is in focus one need not attempt to decompose any other formula. Combining these observations, one can describe a bottom-up *focusing proof search* procedure for linear logic as follows:

1. Greedily decompose positive connectives on the left and negative connectives on the right by applying invertible rules, until the goal sequent becomes *stable,* i.e., has only negative formulas and positive atoms on the left, and positive formulas and negative atoms on the right.

2. Focus on some formula in a stable sequent, and attempt to prove the goal using the rule(s) associated with the formula's principal connective. Keep focus on the formula's subformulas and repeat this step, until reaching a *polarity mismatch,* i.e., a positive formula in left focus, or a negative formula in right focus.

3. On a polarity mismatch, *blur* the sequent (i.e., lose focus) and go back to stage (1).

Finally, this description may be refined with one additional observation (Andreoli, 2001): the particular sequence of left and right rules occurring within stages (1) and (2) is unimportant, and one can instead consider each stage as a single step. By this view, stage (1) applies a single, invertible rule for decomposing a "compound connective", with a set of stable sequents as premises. Stage (2) applies one rule from a *set* of derived rules for introducing the compound connective in focus—the choice of which rule to apply is part of the non-determinism of proof search (the additional source of non-determinism being the transition from (1) to (2), i.e., the choice of formula to focus on).

From the point of view of proof search, the crucial fact is that focusing is not only sound (obvious, since it is just a restriction of ordinary bottom-up search) but also *complete:* if a linear logic sequent is provable, then it has a focusing proof. To say that focusing is merely a "speed-up" of proof search is not to do it justice, however. Soon after Andreoli's work, Girard (1993) proposed that polarity (i.e., the focusing behavior of connectives) was a general phenomenon that could be used to study logic in a unified setting. In particular, he explained that the connectives of classical logic have ambiguous polarity, but that different proofs can be isolated with different *polarizations* of the connectives. The connectives of intuitionistic logic are likewise ambiguous for the most part, but on the other hand, some of its characteristic features such as the disjunction and existence properties can be explained by those connectives having positive polarity. More recently, Girard (2001) has taken focusing proofs as the starting point for a general study of the meaning of logical rules.

But how does this relate to *Dummett's* analysis of the meaning of logical rules? To give away the punchline, our multi-judgment reconstruction of Dummett's analysis may alternatively be seen as a focusing analysis of the classical sequent calculus. We now describe this correspondence.

Given a context of hypotheses Γ, we define a pair of multisets of polarized formulas $(\Lambda_\Gamma, \Theta_\Gamma)$ (omitting the subscript when clear from context, no pun intended):

$$\begin{array}{rcl} \Lambda & = & \{X \mid X \ \mathit{triv} \in \Gamma\} \cup \{N \mid N \ \mathit{true} \in \Gamma\} \\ \Theta & = & \{\overline{X} \mid \overline{X} \ \mathit{absurd} \in \Gamma\} \cup \{P \mid P \ \mathit{false} \in \Gamma\} \end{array}$$

Now, we define five kinds of sequents—four styles of *focused* sequents (which have a single polarized formula in focus on either the left or right), as well as *unfocused* sequents:

$$\begin{array}{cc} \Lambda \to \Theta \mid P & P \mid \Lambda \to \Theta \\ \Lambda \to \Theta \mid N & N \mid \Lambda \to \Theta \\ \multicolumn{2}{c}{\Lambda \to \Theta} \end{array}$$

Hypothetical judgments of polarized logic can be rewritten as focused and unfocused sequents, as follows:

[10] Andreoli considers a one-sided sequent calculus for linear logic, and rather than negative vs. positive polarity he uses the terms "asynchronous" vs. "synchronous".

$$\begin{array}{rcl}
\Gamma \vdash P\ \mathit{triv} & \leftrightarrow & \Lambda \to \Theta \mid P \\
\Gamma \vdash P\ \mathit{false} & \leftrightarrow & P \mid \Lambda \to \Theta \\
\Gamma \vdash N\ \mathit{true} & \leftrightarrow & \Lambda \to \Theta \mid N \\
\Gamma \vdash N\ \mathit{absurd} & \leftrightarrow & N \mid \Lambda \to \Theta \\
\Gamma \vdash \mathit{contra} & \leftrightarrow & \Lambda \to \Theta
\end{array}$$

The judgment $\Gamma \vdash \Delta$ is mapped to a set of sequents (and assertions "$X \in \Lambda$" or "$\overline{X} \in \Theta$") by applying the expansion lemma. Observe that "truthy" hypotheses ($X\ \mathit{triv}$ and $N\ \mathit{true}$) become formulas on the *left* side of the sequent, but "truthy" conclusions ($P\ \mathit{triv}$ and $N\ \mathit{true}$) become formulas in *right*-focus. Likewise, "falsish" hypotheses become formulas on the right side of the sequent, while "falsish" conclusions become formulas in left-focus. We illustrate with a few examples.

Example 9. In Example 1, we found two rules for proving $X \otimes (Y \oplus {}^{v}\neg Q)$ directly:

$$\frac{X\ \mathit{triv} \in \Gamma \quad Y\ \mathit{triv} \in \Gamma}{\Gamma \vdash X \otimes (Y \oplus {}^{v}\neg Q)\ \mathit{triv}} \qquad \frac{X\ \mathit{triv} \in \Gamma \quad \Gamma \vdash Q\ \mathit{false}}{\Gamma \vdash X \otimes (Y \oplus {}^{v}\neg Q)\ \mathit{triv}}$$

Via the above translation, these become rules for deriving the formula in right-focus:

$$\frac{X \in \Lambda \quad Y \in \Lambda}{\Lambda \to \Theta \mid X \otimes (Y \oplus {}^{v}\neg Q)} \qquad \frac{X \in \Lambda \quad Q \mid \Lambda \to \Theta}{\Lambda \to \Theta \mid X \otimes (Y \oplus {}^{v}\neg Q)}$$

■

Example 10. In Example 2, we gave the rule for refuting $X \otimes (Y \oplus {}^{v}\neg Q)$:

$$\frac{\Gamma, X\ \mathit{triv}, Y\ \mathit{triv} \vdash \mathit{contra} \quad \Gamma, X\ \mathit{triv}, Q\ \mathit{false} \vdash \mathit{contra}}{\Gamma \vdash X \otimes (Y \oplus {}^{v}\neg Q)\ \mathit{false}}$$

After translation this becomes a rule for decomposing the formula in left-focus:

$$\frac{X, Y, \Lambda \to \Theta \quad X, \Lambda \to \Theta, Q}{X \otimes (Y \oplus {}^{v}\neg Q) \mid \Lambda \to \Theta}$$

■

Example 11. We had two rules for establishing contradiction:

$$\frac{P\ \mathit{false} \in \Gamma \quad \Gamma \vdash P\ \mathit{triv}}{\Gamma \vdash \mathit{contra}} \qquad \frac{N\ \mathit{true} \in \Gamma \quad \Gamma \vdash N\ \mathit{absurd}}{\Gamma \vdash \mathit{contra}}$$

After translation these are read as proof-search rules for "choosing" a focus on the right or left of the sequent:

$$\frac{P \in \Theta \quad \Lambda \to \Theta \mid P}{\Lambda \to \Theta} \qquad \frac{N \in \Lambda \quad N \mid \Lambda \to \Theta}{\Lambda \to \Theta}$$

■

Now, extending the definition of Section 2.2, we define $|-|$ as the operator that "forgets" polarity, converting a polarized formula to an ordinary propositional formula.

Definition 16. $|P|$ and $|N|$ are defined as follows:[11]

$$\begin{array}{rr}
|1| = |\top| = \mathsf{T} & |0| = |\bot| = \mathsf{F} \\
|P \otimes Q| = |P| \wedge |Q| & |N \& M| = |N| \wedge |M| \\
|P \oplus Q| = |P| \vee |Q| & |N \parr M| = |N| \vee |M| \\
|{}^{v}\neg P| = \neg |P| & |{}^{n}\neg N| = \neg |N| \\
|{\downarrow}N| = |N| & |{\uparrow}P| = |P| \\
|X| = X & |\overline{X}| = \overline{X}
\end{array}$$

[11] Observe we do *not* define $|\overline{X}| = X$. Polarity may be thought of as a fixed partitioning of the set of atoms, which does not go away even if we forget about it.

Logical judgment	Typing judgment	Meaning of typing judgment
$\Gamma \vdash P\ \mathit{triv}$	$\Gamma \vdash V \stackrel{\text{val}}{:} P$	value V has type P
$\Gamma \vdash P\ \mathit{false}$	$\Gamma \vdash K \stackrel{\text{cnt}}{:} P$	CBV continuation K accepts type P
$\Gamma \vdash N\ \mathit{absurd}$	$\Gamma \vdash C \stackrel{\text{cov}}{:} N$	"covalue" C accepts type N
$\Gamma \vdash N\ \mathit{true}$	$\Gamma \vdash E \stackrel{\text{exp}}{:} N$	expression E has type N
$\Gamma \vdash \Delta$	$\Gamma \vdash \sigma \stackrel{\text{sub}}{:} \Delta$	σ is a well-typed substitution for Δ
$\Gamma \vdash \mathit{contra}$	$\Gamma \vdash S \stackrel{\text{stm}}{:} \#$	statement S is well-typed

Figure 3: Curry-Howard interpretation of polarized logic

In other words, $|-|$ collapses oppositely polarized conjunctions, disjunctions, and negations, and erases the shift operators.[12] For any unpolarized formula, we can consider its *polarizations,* i.e., the inverse image of $|-|$. Clearly, $|-|$ is not injective—any formula can be given at least two polarizations, one positive and one negative, and in fact there are infinitely many, corresponding to arbitrary compositions of the shift operators.

The completeness of focusing can now be stated as follows:

If $|\Lambda| \to |\Theta|$ is classically provable, then $\Lambda \to \Theta$ has a focusing proof.

We will wait until Section 3.5 to prove the completeness theorem, so that we can annotate it with terms from the Curry-Howard interpretation (developed below), and thus better illustrate the computational content of completeness. Intuitively, though, we should already be able to see that focusing proofs for different polarizations of classical theorems correspond to different kinds of double-negation translations. For instance, suppose A is true classically, and that P is one possible polarization ($|P| = A$). Then completeness says that $\cdot \to P$ has a focusing proof—but by the correspondence we established above, this is just alternative notation for the hypothetical judgment $P\ \mathit{false} \vdash \mathit{contra}$.

In Section 2.2, we explained how the positive fragment of polarized logic soundly and completely interprets intuitionistic logic (or to be precise, its theorems). Likewise, in Section 2.3 we alluded to a similar relationship between the negative fragment and Czermak's dual intuitionistic logic. And now we have just claimed that polarized logic soundly and completely interprets the classical sequent calculus. Thus much in the spirit of Girard's **LU** (Girard, 1993), polarized logic serves as a bridge between different logical traditions.

But we care not only about provability, but also about proofs. In the next section, we explore the structure of proofs in polarized logic by giving it a Curry-Howard interpretation. We will find that various concepts from the tradition of *programming languages* emerge out of this analysis. For example, *pattern-typing* is revealed as a natural analogue of the logical inversion judgments. Moreover, we will explain how polarization explicitly reflects *evaluation order* at the level of types, and thus gives a logical foundation for studying operationally-sensitive typing phenomena.

3 The logic of values and continuations (and their duals)

We saw how Dummett's analysis of the justification of logical laws through alternative "meaning-theories" could be formalized through the judgmental method as *polarized logic,* a rich setting for describing different notions of proof and refutation. Now, we will explain how the judgments of polarized logic each correspond precisely to natural programming language constructs. A summary of this correspondence is given in Figure 3, defining a programming language which we call the *Calculus of Unity,* or **CU**. We explain it in detail, beginning with the positive fragment.

3.1 The call-by-value fragment

Recall that the central objects of a verificationist meaning-theory are direct proofs and the corresponding inferences they justify. In Section 2.2, we represented these concepts by judgments $P\ \mathit{triv}$ ("P has a direct

[12] If $(-)^{\perp}$ were defined as a first-class connective (see Footnote 9), we would likewise collapse it to negation.

proof") and P *false* ("P has a justified refutation"). The core idea behind the Curry-Howard isomorphism is that

$$\begin{array}{rcl} \text{direct proofs} & \cong & \text{values} \\ \text{justified refutations} & \cong & \text{call-by-value continuations} \end{array}$$

Correspondingly, the two logical judgments will become typing judgments $V \stackrel{\text{val}}{:} P$ ("value V has type P") and $K \stackrel{\text{cnt}}{:} P$ ("call-by-value continuation K accepts type P").

But again recall that we did not give rules for the logical judgments directly, but instead stated them in terms of a relation $\Delta \Rightarrow P$ *triv*, which holds whenever a direct proof of P could have Δ as its fringe of primitive premises. So what do different derivations of $\Delta \Rightarrow P$ look like? Abstractly, they describe the shape or the *pattern* that a direct proof of P takes, up to the point where it reaches primitive premises. Let us label these primitive hypotheses in Δ with variables: *atomic value variables* $x \stackrel{\text{val}}{:} X$ and *continuation variables* $\overline{u} \stackrel{\text{cnt}}{:} P$. Then we can interpret $\Delta \Rightarrow P$ *triv* as *pattern-typing:*

$$\frac{}{x \stackrel{\text{val}}{:} X \Rightarrow x \stackrel{\text{val}}{:} X} \qquad \frac{}{\overline{u} \stackrel{\text{cnt}}{:} P \Rightarrow \overline{u} \stackrel{\text{val}}{:} \stackrel{v}{\neg} P}$$

$$\frac{}{\cdot \Rightarrow () \stackrel{\text{val}}{:} 1} \qquad \frac{\Delta_1 \Rightarrow p_1 \stackrel{\text{val}}{:} P \quad \Delta_2 \Rightarrow p_2 \stackrel{\text{val}}{:} Q}{\Delta_1, \Delta_2 \Rightarrow (p_1, p_2) \stackrel{\text{val}}{:} P \otimes Q}$$

$$\text{(no rule for 0)} \qquad \frac{\Delta \Rightarrow p \stackrel{\text{val}}{:} P}{\Delta \Rightarrow \mathsf{inl}(p) \stackrel{\text{val}}{:} P \oplus Q} \qquad \frac{\Delta \Rightarrow p \stackrel{\text{val}}{:} Q}{\Delta \Rightarrow \mathsf{inr}(p) \stackrel{\text{val}}{:} P \oplus Q}$$

With this interpretation, observe that some of the formal characteristics of the logical axiomatization are mapped to quite familiar (to programmers) features of pattern-matching. For example, the linearity of Δ corresponds to the usual restriction that *patterns cannot bind a variable more than once.* The fact that P *false* $\Rightarrow \stackrel{v}{\neg} P$ *triv* is a base case corresponds to the restriction that *one cannot pattern-match on continuations.* Pattern-typing satisfies a few additional, simple properties. We say that p is a P*-pattern* if there exists some Δ such that $\Delta \Rightarrow p \stackrel{\text{val}}{:} P$

Proposition 17. *For any P, there exist only finitely many P-patterns.*

Proposition 18. *If p is a P-pattern, then there exists a unique Δ such that $\Delta \Rightarrow p \stackrel{val}{:} P$.*

These properties will play a role analogous to that of the finite support property in Section 2.2, i.e., not a very significant one.[13] What *is* crucial about derivations of $\Delta \Rightarrow p \stackrel{\text{val}}{:} P$ is (as for their logical counterparts) simply their inductive definition, and hence our ability to meaningfully quantify over them.

With pattern-typing defined, we can give the rules for type-checking values and continuations. The single rule for establishing P *triv* in context was

$$\frac{\Delta \Rightarrow P \text{ } triv \quad \Gamma \vdash \Delta}{\Gamma \vdash P \text{ } triv}$$

Now, the first premise is annotated with a pattern p. The second premise is annotated with an *explicit substitution* σ, as we will describe shortly. By pairing these two objects, we form a value:

$$\frac{\Delta \Rightarrow p \stackrel{\text{val}}{:} P \quad \Gamma \vdash \sigma \stackrel{\text{sub}}{:} \Delta}{\Gamma \vdash [\sigma] p \stackrel{\text{val}}{:} P}$$

An explicit substitution is a list of term-for-variable replacements. It is a well-typed substitution for Δ if all the terms have the appropriate types. Within the positive fragment, Δ contains only hypotheses $x \stackrel{\text{val}}{:} X$ and $\overline{u} \stackrel{\text{cnt}}{:} P$, so that the following substitution-checking rules suffice:

$$\frac{}{\Gamma \vdash \cdot \stackrel{\text{sub}}{:} \cdot} \qquad \frac{y \stackrel{\text{val}}{:} X \in \Gamma \quad \Gamma \vdash \sigma \stackrel{\text{sub}}{:} \Delta}{\Gamma \vdash (y/x, \sigma) \stackrel{\text{sub}}{:} (x \stackrel{\text{val}}{:} X, \Delta)} \qquad \frac{\Gamma \vdash K \stackrel{\text{cnt}}{:} P \quad \Gamma \vdash \sigma \stackrel{\text{sub}}{:} \Delta}{\Gamma \vdash (K/\overline{u}, \sigma) \stackrel{\text{sub}}{:} (\overline{u} \stackrel{\text{cnt}}{:} P, \Delta)}$$

[13] Indeed, one can consider introducing pattern-typing rules that break one or both properties, as we will discuss in Section 4.

The explicit substitution notation for values may seem strange, but it is really interchangeable with a traditional abstract syntax tree notation. Indeed, if V is a tree, the equation $V = [\sigma]p$ can be seen as "factoring" V into a rooted subtree p together with a fringe σ. This factorization is unique (given the grammar of patterns), so without danger we will use either notation.

EXAMPLE 12. Let $K \stackrel{\text{cnt}}{:} P$ be a well-typed continuation (in some context Γ). Then we have $V \stackrel{\text{val}}{:} (\stackrel{v}{\neg} P) \otimes (\stackrel{v}{\neg} P)$ where $V = [K/\overline{u}_1, K/\overline{u}_2](\overline{u}_1, \overline{u}_2)$, which we can also simply write as (K, K). ■

Moving onward, the single rule for establishing P *false* in context was

$$\frac{\forall(\Delta \Rightarrow P \ \mathit{triv}) : \quad \Gamma, \Delta \vdash \mathit{contra}}{\Gamma \vdash P \ \mathit{false}}$$

Again, each derivation $\Delta \Rightarrow P \ \mathit{triv}$ is annotated with a pattern p, while (as we describe below) proofs of contradiction are annotated with executable *statements.* Thus we can interpret continuations accepting type P as *maps from P-patterns to well-typed statements.* Or to be more precise, we interpret untyped continuations by *partial* maps from patterns to statements, and (letting ϕ range over such mappings) check that they accept type P as follows:

$$\frac{\forall(\Delta \Rightarrow p \stackrel{\text{val}}{:} P) : \quad \Gamma, \Delta \vdash \phi(p) \stackrel{\text{stm}}{:} \#}{\Gamma \vdash (\lambda\phi) \stackrel{\text{cnt}}{:} P}$$

As for its logical counterpart, this typing rule deserves a bit of explanation. Abstractly, the syntax of a continuation simply specifies, for any pattern, (at most) a single statement to execute, with variables bound by the pattern. The typing rule checks that for any P-pattern p, in any context Δ (actually unique by Proposition 18) assigning types to the variables bound by p, there is a statement associated with p, and that it is well-typed. Adopting the concrete syntax of, say, ML, we could specify such a continuation by a finite list of branches $p_1 \mapsto S_1 \mid \cdots \mid p_n \mapsto S_n$. Indeed, Proposition 17 ensures that we can always do this for well-typed continuations. However, as we work with **CU**, we will find that it is quite convenient to apply this more abstract, higher-order syntax for continuations.

Finally, we give the rule for forming well-typed statements, obtained by annotating the rule for contradiction:

$$\frac{P \ \mathit{false} \in \Gamma \quad \Gamma \vdash P \ \mathit{triv}}{\Gamma \vdash \mathit{contra}} \quad \rightsquigarrow \quad \frac{\overline{u} \stackrel{\text{cnt}}{:} P \in \Gamma \quad \Gamma \vdash V \stackrel{\text{val}}{:} P}{\Gamma \vdash \overline{u}\ V \stackrel{\text{stm}}{:} \#}$$

The statement $\overline{u}\ V$ has the intuitive reading, "Pass the value V to the continuation variable $\overline{u}$".

EXAMPLE 13. Let $2 = 1 \oplus 1$ be the type of booleans, and let t and f be names for the patterns $\mathsf{inl}()$ and $\mathsf{inr}()$, respectively. We treat t and f both as 2-patterns and as closed values of type 2. Now, we define partial maps *not* and *xor* from patterns to statements, respectively implementing the "not" and "exclusive-or" boolean operators in continuation-passing-style (CPS):

$$\begin{aligned} \mathit{not}\ (\mathsf{t}, \overline{u}) &= \overline{u}\ \mathsf{f} \\ \mathit{not}\ (\mathsf{f}, \overline{u}) &= \overline{u}\ \mathsf{t} \end{aligned}$$

$$\begin{aligned} \mathit{xor}\ ((\mathsf{t}, \mathsf{t}), \overline{u}) &= \overline{u}\ \mathsf{f} \\ \mathit{xor}\ ((\mathsf{t}, \mathsf{f}), \overline{u}) &= \overline{u}\ \mathsf{t} \\ \mathit{xor}\ ((\mathsf{f}, \mathsf{t}), \overline{u}) &= \overline{u}\ \mathsf{t} \\ \mathit{xor}\ ((\mathsf{f}, \mathsf{f}), \overline{u}) &= \overline{u}\ \mathsf{f} \end{aligned}$$

Then $\mathsf{not} = (\lambda \mathit{not})$ and $\mathsf{xor} = (\lambda \mathit{xor})$ are, respectively, continuations accepting types $2 \otimes \stackrel{v}{\neg} 2$ and $(2 \otimes 2) \otimes \stackrel{v}{\neg} 2$, as the reader can verify. ■

What about the properties we proved in Section 2.2 to demonstrate the coherence of the logical rules, namely the identity and reduction principles? Rather than translating the proofs of those properties, we will instead *internalize* the principles in the language, providing additional ways of forming continuations and statements:

$$\frac{\overline{u} \stackrel{\text{cnt}}{:} P \in \Gamma}{\Gamma \vdash \overline{u} \stackrel{\text{cnt}}{:} P} \qquad \frac{\Gamma \vdash K \stackrel{\text{cnt}}{:} P \quad \Gamma \vdash V \stackrel{\text{val}}{:} P}{\Gamma \vdash K\ V \stackrel{\text{stm}}{:} \#}$$

Identity forms a continuation by coercing a continuation variable, while reduction forms a statement by pairing a continuation to a value (with the intuitive reading: "Pass value V to continuation K"). The following example illustrates both uses.

EXAMPLE 14. We define a higher-order function nc, which checks that a unary boolean operator is non-constant. nc is defined in CPS, as a continuation accepting type $\stackrel{v}{\neg}(2 \otimes \stackrel{v}{\neg}2) \otimes \stackrel{v}{\neg}2$:

$$nc\ (\overline{u}, \overline{u}') = \overline{u}\ (\mathsf{t}, \lambda b_1.\overline{u}\ (\mathsf{f}, \lambda b_2.\mathsf{xor}\ ((b_1, b_2), \overline{u}')))$$
$$\mathsf{nc} = (\lambda nc)$$

Note that in the definition of nc, b_1 and b_2 are meta-variables quantifying over 2-patterns. The reduction principle was used to pass a value to xor (which is not a variable), while the identity principle was used to coerce the variable $\overline{u}'$ into a continuation (and then into a value, passed to xor along with (b_1, b_2)). Also observe that the definition of nc chooses an explicit evaluation order: first evaluating $\overline{u}$ on t and storing the result in b_1, then evaluating $\overline{u}$ on f and storing the result in b_2, and finally computing the exclusive-or of (b_1, b_2). ■

Borrowing terminology from the λ-calculus, we could say that terms that do *not* apply the identity principle are "η-long", while terms that do not apply the reduction principle are "β-reduced". Internalizing identity and reduction thus corresponds to allowing terms that are not η-long and β-reduced.

On the other hand, we choose not to internalize the principles of context identity and substitution as typing rules, instead building the associated terms explicitly. Context identity is formulated as follows:

Proposition 19 (Context identity)**.** *For any linear context Δ, we can build $\circ_\Delta$, the substitution which maps all variables in Δ to themselves. Then for all Γ we have $\Gamma, \Delta \vdash \circ_\Delta \stackrel{sub}{:} \Delta$.*

The context identity principle allows us to extend the notational convention introduced in Example 13.

Notation. If p is a P-pattern $\Delta \Rightarrow p \stackrel{\text{val}}{:} P$, we can also write p for the value $[\circ_\Delta]p$. Note that for all Γ we have $\Gamma, \Delta \vdash p \stackrel{\text{val}}{:} P$.

Later, when we define the operational semantics of **CU**, the substitution principle will become an explicit substitution operation on terms. First, though, let us finish describing the rest of the language. The positive fragment is summarized in Figure 4.

3.2 The call-by-name fragment

Defining the negative fragment of **CU** is easy, simply by dualizing the positive fragment. Understanding what it means is a bit more tricky though! The rough idea is that values are replaced by *covalues,* which represent a sort of continuation in canonical form. In particular, unlike call-by-value continuations, it *is* sensible to pattern-match against covalues—and this is how lazy *expressions* are defined. We will present the typing rules without much comment, and then try to provide more examples and intuitions.

Δ now contains *atomic covalue variables* $\overline{x} \stackrel{\text{cov}}{:} \overline{X}$ and *expression variables* $u \stackrel{\text{exp}}{:} N$, and $\Delta \Rightarrow N$ *absurd* is interpreted as *copattern-typing:*

$$\frac{}{\overline{x} \stackrel{\text{cov}}{:} \overline{X} \Rightarrow \overline{x} \stackrel{\text{cov}}{:} \overline{X}} \qquad \frac{}{u \stackrel{\text{exp}}{:} N \Rightarrow u \stackrel{\text{cov}}{:} \stackrel{n}{\neg} N}$$

$$\text{(no rule for } \top\text{)} \qquad \frac{\Delta \Rightarrow \overline{p} \stackrel{\text{cov}}{:} N}{\Delta \Rightarrow \mathsf{fst}(\overline{p}) \stackrel{\text{cov}}{:} N \& M} \qquad \frac{\Delta \Rightarrow \overline{p} \stackrel{\text{cov}}{:} M}{\Delta \Rightarrow \mathsf{snd}(\overline{p}) \stackrel{\text{cov}}{:} N \& M}$$

$$\frac{}{\cdot \Rightarrow [] \stackrel{\text{cov}}{:} \bot} \qquad \frac{\Delta_1 \Rightarrow \overline{p}_1 \stackrel{\text{cov}}{:} N \quad \Delta_2 \Rightarrow \overline{p}_2 \stackrel{\text{cov}}{:} M}{\Delta_1, \Delta_2 \Rightarrow [\overline{p}_1, \overline{p}_2] \stackrel{\text{cov}}{:} N \parr M}$$

A covalue is constructed by pairing a copattern with an explicit substitution:

$$\frac{\Delta \Rightarrow \overline{p} \stackrel{\text{cov}}{:} N \quad \Gamma \vdash \sigma \stackrel{\text{sub}}{:} \Delta}{\Gamma \vdash [\sigma]\overline{p} \stackrel{\text{cov}}{:} N}$$

Linear contexts $\quad \Delta \quad ::= \cdot \mid x \stackrel{\text{val}}{:} X, \Delta \mid \overline{u} \stackrel{\text{cnt}}{:} P, \Delta$

$\boxed{\Delta \Rightarrow p \stackrel{\text{val}}{:} P}$

$$\frac{}{x \stackrel{\text{val}}{:} X \Rightarrow x \stackrel{\text{val}}{:} X} \qquad \frac{}{\overline{u} \stackrel{\text{cnt}}{:} P \Rightarrow \overline{u} \stackrel{\text{val}}{:} \neg P}$$

$$\frac{}{\cdot \Rightarrow () \stackrel{\text{val}}{:} 1} \qquad \frac{\Delta_1 \Rightarrow p_1 \stackrel{\text{val}}{:} P \quad \Delta_2 \Rightarrow p_2 \stackrel{\text{val}}{:} Q}{\Delta_1, \Delta_2 \Rightarrow (p_1, p_2) \stackrel{\text{val}}{:} P \otimes Q}$$

$$\text{(no rule for 0)} \qquad \frac{\Delta \Rightarrow p \stackrel{\text{val}}{:} P}{\Delta \Rightarrow \mathsf{inl}(p) \stackrel{\text{val}}{:} P \oplus Q} \quad \frac{\Delta \Rightarrow p \stackrel{\text{val}}{:} Q}{\Delta \Rightarrow \mathsf{inr}(p) \stackrel{\text{val}}{:} P \oplus Q}$$

Contexts $\quad \Gamma \quad ::= \quad \cdot \mid \Gamma, \Delta$

$\boxed{\Gamma \vdash V \stackrel{\text{val}}{:} P}$

$$\frac{\Delta \Rightarrow p \stackrel{\text{val}}{:} P \quad \Gamma \vdash \sigma \stackrel{\text{sub}}{:} \Delta}{\Gamma \vdash [\sigma] p \stackrel{\text{val}}{:} P}$$

$\boxed{\Gamma \vdash K \stackrel{\text{cnt}}{:} P}$

$$\frac{\forall (\Delta \Rightarrow p \stackrel{\text{val}}{:} P) : \quad \Gamma, \Delta \vdash \phi(p) \stackrel{\text{stm}}{:} \#}{\Gamma \vdash (\lambda \phi) \stackrel{\text{cnt}}{:} P} \qquad \frac{\overline{u} \stackrel{\text{cnt}}{:} P \in \Gamma}{\Gamma \vdash \overline{u} \stackrel{\text{cnt}}{:} P}$$

$\boxed{\Gamma \vdash \sigma \stackrel{\text{sub}}{:} \Delta}$

$$\frac{}{\Gamma \vdash \cdot \stackrel{\text{sub}}{:} \cdot} \qquad \frac{y \stackrel{\text{val}}{:} X \in \Gamma \quad \Gamma \vdash \sigma \stackrel{\text{sub}}{:} \Delta}{\Gamma \vdash (y/x, \sigma) \stackrel{\text{sub}}{:} (x \stackrel{\text{val}}{:} X, \Delta)} \qquad \frac{\Gamma \vdash K \stackrel{\text{cnt}}{:} P \quad \Gamma \vdash \sigma \stackrel{\text{sub}}{:} \Delta}{\Gamma \vdash (K/\overline{u}, \sigma) \stackrel{\text{sub}}{:} (\overline{u} \stackrel{\text{cnt}}{:} P, \Delta)}$$

$\boxed{\Gamma \vdash S \stackrel{\text{stm}}{:} \#}$

$$\frac{\overline{u} \stackrel{\text{cnt}}{:} P \in \Gamma \quad \Gamma \vdash V \stackrel{\text{val}}{:} P}{\Gamma \vdash \overline{u}\ V \stackrel{\text{stm}}{:} \#} \qquad \frac{\Gamma \vdash K \stackrel{\text{cnt}}{:} P \quad \Gamma \vdash V \stackrel{\text{val}}{:} P}{\Gamma \vdash K\ V \stackrel{\text{stm}}{:} \#}$$

Figure 4: **CU** type system (positive fragment)

where explicit substitutions are defined as before, but with additional rules for satisfying negative hypotheses:

$$\frac{\overline{y} \overset{\text{cov}}{:} \overline{X} \in \Gamma \quad \Gamma \vdash \sigma \overset{\text{sub}}{:} \Delta}{\Gamma \vdash (\overline{y}/\overline{x}, \sigma) \overset{\text{sub}}{:} (\overline{x} \overset{\text{cov}}{:} \overline{X}, \Delta)} \qquad \frac{\Gamma \vdash E \overset{\text{exp}}{:} N \quad \Gamma \vdash \sigma \overset{\text{sub}}{:} \Delta}{\Gamma \vdash (E/u, \sigma) \overset{\text{sub}}{:} (u \overset{\text{exp}}{:} N, \Delta)}$$

Expressions are defined by pattern-matching against covalues. Letting ψ ranges over partial maps from copatterns to statements, we write the expression typing rule as:

$$\frac{\forall (\Delta \Rightarrow \overline{p} \overset{\text{cov}}{:} N) : \quad \Gamma, \Delta \vdash \psi(\overline{p}) \overset{\text{stm}}{:} \#}{\Gamma \vdash (\mu\psi) \overset{\text{exp}}{:} N}$$

This μ can be seen as a generalization of μ in the $\lambda\mu$-calculus (Parigot, 1992), as we explain in Section 3.3 below. The negative fragment introduces an additional form of statement, which passes a covalue to an expression variable:

$$\frac{u \overset{\text{exp}}{:} N \in \Gamma \quad \Gamma \vdash C \overset{\text{cov}}{:} N}{\Gamma \vdash u\ C \overset{\text{stm}}{:} \#}$$

Finally, again we internalize the identity and reduction principles:

$$\frac{u \overset{\text{exp}}{:} N \in \Gamma}{\Gamma \vdash u \overset{\text{exp}}{:} N} \qquad \frac{\Gamma \vdash E \overset{\text{exp}}{:} N \quad \Gamma \vdash C \overset{\text{cov}}{:} N}{\Gamma \vdash E\ C \overset{\text{stm}}{:} \#}$$

The negative fragment is summarized in Figure 5.

EXAMPLE 15. Let $E_1 \overset{\text{exp}}{:} N$ and $E_2 \overset{\text{exp}}{:} M$ be expressions. Define the partial map *pair* from copatterns to statements as follows ($\overline{p}_1$ and $\overline{p}_2$ are meta-variables ranging over all N-copatterns and M-copatterns, respectively):

$$\begin{aligned} \mathit{pair}\ \mathsf{fst}(\overline{p}_1) &= E_1\ \overline{p}_2 \\ \mathit{pair}\ \mathsf{snd}(\overline{p}_2) &= E_2\ \overline{p}_1 \end{aligned}$$

Then $E = (\mu \mathit{pair})$ is an expression of type $N \& M$ (reader: verify this fact!). Intuitively, E is a *lazy* pair because, so to speak, it waits for its continuation to make a "decision" about which component to project before evaluating E_1 or E_2. As we formalize in the operational semantics below, the statement $E\ \mathsf{fst}(C)$ always evaluates to $E_1\ C$, and $E\ \mathsf{snd}(C)$ always to $E_2\ C$, regardless of the definitions of E_1 and E_2 (even if, say, $M = \bot$). ■

EXAMPLE 16. We define the "first projection" map π_1 in CPS by $\pi_1\ [u, \overline{p}] = u\ \mathsf{fst}(\overline{p})$. Then $(\mu\pi_1)$ is an expression of type $\overset{n}{\neg}(N \& M) \parr N$. ■

As Examples 13 and 16 suggest, in general we can interpret the call-by-value function space $P \overset{v}{\to} Q$ via CPS transformation by continuations accepting type $P \otimes \overset{v}{\neg} Q$ (or values of type $\overset{v}{\neg}(P \otimes \overset{v}{\neg} Q)$), and the call-by-name function space $N \overset{n}{\to} M$ by expressions of type $\overset{n}{\neg} N \parr M$. These typings of Plotkin's CPS transformations are analogous to several others that appear in the literature (Plotkin, 1975; Streicher and Reus, 1998; Selinger, 2001; Wadler, 2003; Laurent, 2005). However, they are only a starting point for the analysis of real call-by-value and call-by-name languages. If we want to model languages such as ML and Haskell, we can get a much closer interpretation if we do not confine ourselves to the positive or negative fragments, instead using all of **CU**. For this we need the shift operators.

Linear contexts $\quad \Delta \quad ::= \cdots \mid \overline{x} \overset{\text{cov}}{:} \overline{X}, \Delta \mid u \overset{\text{exp}}{:} N, \Delta$

$\boxed{\Delta \Rightarrow \overline{p} \overset{\text{cov}}{:} N}$

$$\frac{}{\overline{x} \overset{\text{cov}}{:} \overline{X} \Rightarrow \overline{x} \overset{\text{cov}}{:} \overline{X}} \qquad \frac{}{u \overset{\text{exp}}{:} N \Rightarrow u \overset{\text{cov}}{:} \pi N}$$

$$\text{(no rule for } \top) \qquad \frac{\Delta \Rightarrow \overline{p} \overset{\text{cov}}{:} N}{\Delta \Rightarrow \mathsf{fst}(\overline{p}) \overset{\text{cov}}{:} N \& M} \qquad \frac{\Delta \Rightarrow \overline{p} \overset{\text{cov}}{:} M}{\Delta \Rightarrow \mathsf{snd}(\overline{p}) \overset{\text{cov}}{:} N \& M}$$

$$\frac{}{\cdot \Rightarrow [] \overset{\text{cov}}{:} \bot} \qquad \frac{\Delta_1 \Rightarrow \overline{p}_1 \overset{\text{cov}}{:} N \quad \Delta_2 \Rightarrow \overline{p}_2 \overset{\text{cov}}{:} M}{\Delta_1, \Delta_2 \Rightarrow [\overline{p}_1, \overline{p}_2] \overset{\text{cov}}{:} N \otimes M}$$

$\boxed{\Gamma \vdash C \overset{\text{cov}}{:} N}$

$$\frac{\Delta \Rightarrow \overline{p} \overset{\text{cov}}{:} N \quad \Gamma \vdash \sigma \overset{\text{sub}}{:} \Delta}{\Gamma \vdash [\sigma]\overline{p} \overset{\text{cov}}{:} N}$$

$\boxed{\Gamma \vdash E \overset{\text{exp}}{:} P}$

$$\frac{\forall (\Delta \Rightarrow \overline{p} \overset{\text{cov}}{:} N) : \quad \Gamma, \Delta \vdash \psi(\overline{p}) \overset{\text{stm}}{:} \#}{\Gamma \vdash (\mu\psi) \overset{\text{exp}}{:} N} \qquad \frac{u \overset{\text{exp}}{:} N \in \Gamma}{\Gamma \vdash u \overset{\text{exp}}{:} N}$$

$\boxed{\Gamma \vdash \sigma \overset{\text{sub}}{:} \Delta}$

$$\frac{}{\Gamma \vdash \cdot \overset{\text{sub}}{:} \cdot} \qquad \frac{\overline{y} \overset{\text{cov}}{:} \overline{X} \in \Gamma \quad \Gamma \vdash \sigma \overset{\text{sub}}{:} \Delta}{\Gamma \vdash (\overline{y}/\overline{x}, \sigma) \overset{\text{sub}}{:} (\overline{x} \overset{\text{cov}}{:} \overline{X}, \Delta)} \qquad \frac{\Gamma \vdash E \overset{\text{exp}}{:} N \quad \Gamma \vdash \sigma \overset{\text{sub}}{:} \Delta}{\Gamma \vdash (E/u, \sigma) \overset{\text{sub}}{:} (u \overset{\text{exp}}{:} N, \Delta)}$$

$\boxed{\Gamma \vdash S \overset{\text{stm}}{:} \#}$

$$\frac{u \overset{\text{exp}}{:} N \in \Gamma \quad \Gamma \vdash C \overset{\text{cov}}{:} N}{\Gamma \vdash u\ C \overset{\text{stm}}{:} \#} \qquad \frac{\Gamma \vdash E \overset{\text{exp}}{:} N \quad \Gamma \vdash C \overset{\text{cov}}{:} N}{\Gamma \vdash E\ C \overset{\text{stm}}{:} \#}$$

Figure 5: **CU** type system (negative fragment)

$$\frac{}{u \overset{\text{exp}}{:} N \Rightarrow u \overset{\text{val}}{:} \downarrow N} \qquad \frac{}{\overline{u} \overset{\text{cnt}}{:} P \Rightarrow \overline{u} \overset{\text{cov}}{:} \uparrow P}$$

Figure 6: **CU** type system (shift operators)

3.3 The shifts: modelling control

To extend **CU** with the shift operators (Figure 6), we simply include additional (co)pattern-typing rules:

$$\overline{u \overset{\text{exp}}{:} N \Rightarrow u \overset{\text{val}}{:} \downarrow N} \qquad \overline{\overline{u} \overset{\text{cnt}}{:} P \Rightarrow \overline{u} \overset{\text{cov}}{:} \uparrow P}$$

In Section 2.4, we interpreted the shift operators as modalities, which safely embed the lax notion of justified proof into the more stringent notion of direct proof, and likewise justified refutation into direct refutation. They now have a very concrete operational reading: an expression $E \overset{\text{exp}}{:} N$ can be *suspended* and treated as a value $E \overset{\text{val}}{:} \downarrow N$, while a continuation $K \overset{\text{cnt}}{:} P$ can be *captured* and treated as a covalue $K \overset{\text{cov}}{:} \uparrow P$.

Example 17. Control operators such as callcc have long been associated with Curry-Howard interpretations of classical logic (Griffin, 1990). In *polarized* logic, we can distinguish between values of type P, which do not have control effects, and expressions of type $\uparrow P$, which can. For example, ordinarily callcc is typed using Pierce's Law, but in polarized logic we have the following derived rule:

$$\frac{\Gamma, \overline{u} \overset{\text{cnt}}{:} P \vdash E \overset{\text{exp}}{:} \uparrow P}{\Gamma \vdash \mathsf{callcc}\ \overline{u}.E \overset{\text{exp}}{:} \uparrow P}$$

where $\mathsf{callcc}\ \overline{u}.E$ is syntactic sugar for $\mu\overline{u}.E\ \overline{u}$. Similarly, instead of double-negation elimination (which types Parigot's μ and Felleisen's $\mathcal{C}$) we have:

$$\frac{\Gamma, \overline{u} \overset{\text{cnt}}{:} P \vdash S \overset{\text{stm}}{:} \#}{\Gamma \vdash \mu\overline{u}.S \overset{\text{exp}}{:} \uparrow P}$$

As a special case of the latter, we can lift any value $V \overset{\text{exp}}{:} P$ to an expression $\mu\overline{u}.\overline{u}\ V \overset{\text{exp}}{:} \uparrow P$ (cf. the second derivation in Example 7). ■

In general, we can use expressions of type $\uparrow P$ to model "possibly effectful" call-by-value terms. The usual introduction and elimination rules of the lambda-calculus, which operate not only on values but on arbitrary terms, can be recovered in **CU** as derived rules *at shifted type.* For example, we can give a pair-formation rule:

$$\frac{\Gamma \vdash E_1 \overset{\text{exp}}{:} \uparrow P \quad \Gamma \vdash E_2 \overset{\text{exp}}{:} \uparrow Q}{\Gamma \vdash (E_1, E_2) \overset{\text{exp}}{:} \uparrow(P \otimes Q)}$$

where (E_1, E_2) is syntactic sugar for the left-to-right evaluation:

$$(E_1, E_2) = \mu\overline{u}.E_1\ (\lambda p_1.E_2\ (\lambda p_2.\overline{u}\ (p_1, p_2)))$$

One may wonder how this analysis compares with Moggi's notion of computational monad (Moggi, 1991). Categorically, the relationship may be understood as that between a monad and an adjunction.[14] In other words, the shift operators decompose the monad as $\bigcirc = \downarrow\uparrow$. Operationally, we can think of the relationship between expressions $E \overset{\text{exp}}{:} \uparrow P$ and values of monadic type $V \overset{\text{val}}{:} \downarrow\uparrow P$, the key difference being that V is a *suspended* expression—it is passed to a continuation which may evaluate it arbitrarily many times, whereas E has control over its own evaluation.

We can likewise use the shift operators to give a better encoding of lazy evaluation. As Filinski (1989) observed (cf. §2.5.3), what are typically thought of as lazy sums (e.g., in Haskell) are emphatically *not* the duals of strict products. Instead, what is usually meant is a form of "lazy pattern-matching", where a sum is eagerly reduced down to a tag indicating the correct branch, but no further. Filinski shows how to simulate this behavior in a language similar to the negative fragment of **CU**, effectively by encoding sums as expressions of type ${}^{n}\neg{}^{n}M \otimes {}^{n}\neg{}^{n}N$. But since "lazy" sums really combine lazy and eager evaluation, we can give a more direct encoding using the shift operators. The type $\uparrow(\downarrow M \oplus \downarrow N)$ can be read directly, as that of an expression ($\uparrow$) computing a tagged ($\oplus$), suspended ($\downarrow$) expression.[15]

[14] We have $\uparrow \dashv \downarrow$ in the partial order of Section 2.4, since $\uparrow P \leq N$ iff $P \leq \downarrow N$.

[15] For similar reasons, $\top$ does not really correspond to the type () in Haskell, since it is possible to force evaluation of the latter by pattern-matching. Instead, a better encoding is $\uparrow 1$.

ML	**CU**
$V : \tau$	$V \stackrel{\text{val}}{:} P$
$E : \tau$	$E \stackrel{\text{exp}}{:} \uparrow P$
$\tau_1 * \tau_2$	$P \otimes Q$
`unit`	1
$\tau_1 + \tau_2$	$P \oplus Q$
`void`	0
$\tau \to$ `void`	$\stackrel{v}{\neg} P$
$\tau_1 \to \tau_2$	$\stackrel{v}{\neg}(P \otimes \stackrel{v}{\neg} Q)$

Haskell	**CU**
$E : \tau$	$E \stackrel{\text{exp}}{:} N$
(τ_1, τ_2)	$N \& M$
$()$	$\uparrow 1$
`Either` τ_1 τ_2	$\uparrow(\downarrow N \oplus \downarrow M)$
`Void`	$\bot$
$\tau \to$ `Void`	$\stackrel{n}{\neg} N$
$\tau_1 \to \tau_2$	$\stackrel{n}{\neg} N \otimes M$

Figure 7: Polarity Pocket Dictionary® (for recreational use only!)

$\boxed{S \rightsquigarrow S'}$

$$(\lambda\phi)([\sigma]p) \rightsquigarrow [\sigma]\phi(p) \qquad (\mu\psi)([\sigma]\overline{p}) \rightsquigarrow [\sigma]\psi(\overline{p})$$

$\boxed{[\sigma]t}$

$$[\sigma]v = \begin{cases} t & (t/v) \in \sigma \\ v & v \notin \mathrm{dom}(\sigma) \end{cases}$$

$$[\sigma]([\sigma']p) = [[\sigma]\sigma']p \qquad [\sigma]([\sigma']\overline{p}) = [[\sigma]\sigma']\overline{p}$$

$$[\sigma](\lambda\phi) = \lambda p.[\sigma]\phi(p) \quad [\sigma](\mu\psi) = \mu\overline{p}.[\sigma]\psi(\overline{p})$$

$$[\sigma]\cdot = \cdot \quad [\sigma](t/v, \sigma') = ([\sigma]t/v, [\sigma]\sigma')$$

$$[\sigma](K\ V) = ([\sigma]K\ [\sigma]V) \quad [\sigma](E\ C) = ([\sigma]E\ [\sigma]C)$$

Figure 8: **CU** operational semantics

Summarizing these analyses, in Figure 7 we publish a "pocket dictionary", which gives a rough translation from judgments and types of ML and Haskell to those of **CU**. These translations are provided only for intuition, and do not capture all the properties of the source languages (e.g., the purity of Haskell). Establishing a formal correspondence is far beyond the scope of this paper. We do postulate, though, that the ML entries are fairly accurate.

3.4 Operational semantics and type safety

By adapting the proofs of the substitution and reduction principles in Section 2.2, we now give **CU** a traditional small-step operational semantics.

Notation. We use t to stand for an arbitrary term of **CU**, i.e., a value V, continuation K, covalue C, expression E, statement S, or substitution σ. Similarly, we use v to stand for an arbitrary variable x, $\overline{x}$, u, or $\overline{u}$. We use $t : J$ to stand for an arbitrary typing judgment, i.e., $V \stackrel{\text{val}}{:} P$, $K \stackrel{\text{cnt}}{:} P$, $C \stackrel{\text{cov}}{:} N$, $E \stackrel{\text{exp}}{:} P$, $S \stackrel{\text{stm}}{:} \#$, or $\sigma \stackrel{\text{sub}}{:} \Delta$.

Given a substitution σ and term t, we let $[\sigma]t$ stand for the usual simultaneous, capture-avoiding substitution. $[\sigma]t$ is defined in the obvious way (see Figure 8), but it is worth mentioning one of the higher-order cases: applying σ to a continuation $(\lambda\phi)$ builds a new map from patterns to statements, by post-composing ϕ with σ, i.e., $[\sigma](\lambda\phi) = \lambda p.[\sigma]\phi(p)$.

Lemma 20 (Substitution). *If* $\Gamma, \Delta \vdash t : J$ *and* $\Gamma \vdash \sigma \overset{sub}{\vdots} \Delta$*, then* $\Gamma \vdash [\sigma]t : J$.

Proof. Immediate by induction on the derivation of $\Gamma, \Delta \vdash t : J$. The proof is essentially identical to the proof in Section 2.2, except that in the case where $t = \overline{u}\ V$ and $\overline{u} \in \mathrm{dom}(\sigma)$ (or dually, $t = u\ C$ and $u \in \mathrm{dom}(\sigma)$), we need not appeal to a separate induction hypothesis, but instead can directly apply the typing rule internalizing the reduction principle. □

The operational semantics can now be specified as a reduction relation $S \rightsquigarrow S'$ on statements. Indeed, it is exceptionally easy to describe:

$$(\lambda\phi)([\sigma]p) \rightsquigarrow [\sigma]\phi(p) \qquad (\mu\psi)([\sigma]\overline{p}) \rightsquigarrow [\sigma]\psi(\overline{p})$$

Of course, our ability to state the reduction relation so concisely relies on the duality between the higher-order representation of continuations and the explicit substitution notation for values (ditto for expressions and covalues). Note that since ϕ and ψ are partial maps, these reductions can only be applied when $\phi(p)$ and $\psi(\overline{p})$ are defined.

Example 18. Recall the definitions of nc and not from Examples 13 and 14. Let $\overline{u} \overset{\mathrm{cnt}}{\vdots} 2$ be a "top-level" boolean continuation. Then $\mathsf{nc}\ (\mathsf{not}, \overline{u})$ evaluates to $\overline{u}\ \mathsf{t}$, as the following calculation verifies.

$\mathsf{nc}\ (\mathsf{not}, \overline{u})$
$\rightsquigarrow \mathsf{not}\ (\mathsf{t}, \lambda b_1.\mathsf{not}\ (\mathsf{f}, \lambda b_2.\mathsf{xor}\ ((b_1, b_2), \overline{u})))$
$\rightsquigarrow (\lambda b_1.\mathsf{not}\ (\mathsf{f}, \lambda b_2.\mathsf{xor}\ ((b_1, b_2), \overline{u})))\ \mathsf{f}$
$\rightsquigarrow \mathsf{not}\ (\mathsf{f}, \lambda b_2.\mathsf{xor}\ ((\mathsf{f}, b_2), \overline{u}))$
$\rightsquigarrow (\lambda b_2.\mathsf{xor}\ ((\mathsf{f}, b_2), \overline{u}))\ \mathsf{t}$
$\rightsquigarrow \mathsf{xor}\ ((\mathsf{f}, \mathsf{t}), \overline{u})$
$\rightsquigarrow \overline{u}\ \mathsf{t}$

■

That this example used a "top-level" continuation variable illustrates a slightly counterintuitive aspect of the pure Curry-Howard interpretation of polarized logic: there are no closed, well-typed statements. Indeed, this is due to the consistency of logic! For well-typed statements correspond to proofs of contradiction, and a closed well-typed statement would be a proof of a contradiction from no assumptions. Although not disastrous, this is somewhat inconvenient because it suggests we have to consider evaluation of open statements for the semantics to be non-vacuous. However, a conceptually simpler alternative presents itself: adding a single closed, well-typed statement. We call this done since it represents termination, and type it as follows:

$$\frac{}{\Gamma \vdash \mathsf{done} \overset{\mathrm{stm}}{\vdots} \#}$$

Of course, this rule is not "logical"—it makes the Curry-Howard interpretation inconsistent.[16] But from an operational perspective the rule is quite mundane. Indeed, since we interpret statements as effectful computations, morally their definition *should* be open-ended, as it would be in a realistic programming language. In Section 4 below we consider extending **CU** with additional effects—the statement done is simply the most basic. We can now prove a (non-vacuous!) type safety theorem for evaluation of closed statements.

Lemma 21 (Progress). *If* $\vdash S \overset{stm}{\vdots} \#$ *then either* $S = \mathsf{done}$ *or else there exists an* S' *such that* $S \rightsquigarrow S'$.

Proof. By inversion on the typing derivation, either $S = \mathsf{done}$ (and we're done!) or $S = (\lambda\phi)([\sigma]p)$ or $S = (\mu\psi)([\sigma]\overline{p})$. These two cases are entirely symmetric, so we only examine the former. Since S is well-typed, there exists some P such that $\vdash (\lambda\phi) \overset{\mathrm{cnt}}{\vdots} P$ and $\vdash [\sigma]p \overset{\mathrm{val}}{\vdots} P$. Inverting $\vdash [\sigma]p \overset{\mathrm{val}}{\vdots} P$, we establish that p is a P-pattern, and inverting $\vdash (\lambda\phi) \overset{\mathrm{cnt}}{\vdots} P$ we derive that $\phi(p)$ is defined. Hence $(\lambda\phi)([\sigma]p) \rightsquigarrow [\sigma]\phi(p)$. □

[16] The reader familiar with ludics may note the similarity between done and "daimon" (Girard, 2001). Girard introduced the daimon rule to get around the problem of the "empty pivot", essentially what we have just described.

$$\frac{}{X, \Lambda \to^c \Theta, X}\ \mathit{init} \qquad \frac{\Lambda \to^c \Theta, A \quad A, \Lambda \to^c \Theta}{\Lambda \to^c \Theta}\ \mathit{cut}$$

$$\frac{A, A \wedge B, \Lambda \to^c \Theta}{A \wedge B, \Lambda \to^c \Theta}\ \wedge L_1 \qquad \frac{B, A \wedge B, \Lambda \to^c \Theta}{A \wedge B, \Lambda \to^c \Theta}\ \wedge L_2 \qquad \frac{\Lambda \to^c \Theta, A \wedge B, A \quad \Lambda \to^c \Theta, A \wedge B, B}{\Lambda \to^c \Theta, A \wedge B}\ \wedge R$$

$$\frac{A, A \vee B, \Lambda \to^c \Theta \quad B, A \vee B, \Lambda \to^c \Theta}{A \vee B, \Lambda \to^c \Theta}\ \vee L \qquad \frac{\Lambda \to^c \Theta, A \vee B, A}{\Lambda \to^c \Theta, A \vee B}\ \vee R_1 \qquad \frac{\Lambda \to^c \Theta, A \vee B, B}{\Lambda \to^c \Theta, A \vee B}\ \vee R_2$$

$$\frac{\neg A, \Lambda \to^c \Theta, A}{\neg A, \Lambda \to^c \Theta}\ \neg L \qquad \frac{A, \Lambda \to^c \Theta, \neg A}{\Lambda \to^c \Theta, \neg A}\ \neg R$$

Figure 9: Classical sequent calculus

Lemma 22 (Preservation)**.** *If* $\Gamma \vdash S \overset{stm}{\cdot\!\cdot} \#$ *and* $S \rightsquigarrow S'$ *then* $\Gamma \vdash S \overset{stm}{\cdot\!\cdot} \#$.

Proof. By inversion on the reduction relation, we have either $S = (\lambda\phi)([\sigma]p) \rightsquigarrow [\sigma]\phi(p) = S'$ or $S = (\mu\psi)([\sigma]\overline{p}) \rightsquigarrow [\sigma]\psi(\overline{p}) = S'$. Again these cases are entirely symmetric. Now we invert the typing derivation $\Gamma \vdash (\lambda\phi)([\sigma]p) \overset{\text{stm}}{\cdot\!\cdot} \#$ to obtain that for some P and Δ such that $\Delta \Rightarrow p \overset{\text{val}}{\cdot\!\cdot} P$, we have $\Gamma \vdash \sigma \overset{\text{sub}}{\cdot\!\cdot} \Delta$ and $\Gamma, \Delta \vdash \phi(p) \overset{\text{stm}}{\cdot\!\cdot} \#$. Then $\Gamma \vdash [\sigma]\phi(p) \overset{\text{stm}}{\cdot\!\cdot} \#$ holds by substitution. □

Corollary 23 (Type safety)**.** *If* S *is a closed, well-typed statement, then either* $S = \mathsf{done}$*, or* S *reduces to a closed, well-typed statement.*

3.5 Postlude: a computational interpretation of focusing completeness

In this section we make good on the promise of Section 2.5 by proving the *focusing completeness* theorem, i.e., that any sequent provable in classical sequent is provable (after polarization) in polarized logic. In order to better illustrate the completeness proof's computational content, we will annotate derivations with terms of **CU**. As an axiomatization of the sequent calculus we will use Kleene's G_3 presentation, given in Figure 9 (Kleene, 1952; Troelstra and Schwichtenberg, 1996). We omit the units, since they do not reveal anything not already illustrated by conjunction and disjunction.

3.5.1 Preliminaries

To state the focusing completeness theorem in full generality, we will have to extend our notion of valid polarized contexts. When giving the rules for polarized logic, we disallowed contexts containing hypotheses P *triv* or N *absurd* except where P and N are atomic. We now relax this restriction.

Definition 24. An *inductive context* is either an ordinary context, or else an inductive context together with a non-primitive hypothesis P *triv* or N *absurd*. The meaning of a hypothetical judgment under an inductive context is expanded as follows: $\Gamma, P\ \mathit{triv} \vdash J$ iff for all $\Delta \Rightarrow P\ \mathit{triv}$, we have $\Gamma, \Delta \vdash J$ (and analogously for $\Gamma, N\ \mathit{absurd} \vdash J$).

This convention is extended to typing judgments by writing P val and N cov instead of P *triv* and N *absurd*, and allowing "inductive terms".

Definition 25. An *inductive term* is either an ordinary term, or else a partial map from (co)patterns to inductive terms. The meaning of a typing judgment under an inductive context is expanded as follows: $\Gamma, P\,\text{val} \vdash t : J$ iff for all $\Delta \Rightarrow p \overset{\text{val}}{\cdot\!\cdot} P$, we have $\Gamma, \Delta \vdash t(p) : J$ (and analogously for $\Gamma, N\,\text{cov} \vdash t : J$).

Proposition 26. *If* $\Gamma, P\,\mathrm{val} \vdash S \mathrel{\overset{stm}{:}} \#$ *then* $\Gamma \vdash (\lambda p.S(p)) \mathrel{\overset{cnt}{:}} P$.

Now, we adapt the sequent notation defined in Section 2.5 to **CU**. For any inductive context Γ, the pair of contexts (Λ, Θ) is constructed by placing hypotheses $x \mathrel{\overset{\mathrm{val}}{:}} X$, $\overline{u} \mathrel{\overset{\mathrm{cnt}}{:}} P$, and $P\,\mathrm{val}$ in Λ, and hypotheses $\overline{x} \mathrel{\overset{\mathrm{val}}{:}} \overline{X}$, $u \mathrel{\overset{\mathrm{exp}}{:}} N$, and $N\,\mathrm{cov}$ in Θ. We define alternative syntax for the typing judgments:

$$\begin{array}{rcl}
\Gamma \vdash V \mathrel{\overset{\mathrm{val}}{:}} P & \leftrightarrow & V : (\Lambda \to \Theta \mid P) \\
\Gamma \vdash K \mathrel{\overset{\mathrm{cnt}}{:}} P & \leftrightarrow & K : (P \mid \Lambda \to \Theta) \\
\Gamma \vdash E \mathrel{\overset{\mathrm{exp}}{:}} N & \leftrightarrow & E : (\Lambda \to \Theta \mid N) \\
\Gamma \vdash C \mathrel{\overset{\mathrm{cov}}{:}} N & \leftrightarrow & C : (N \mid \Lambda \to \Theta) \\
\Gamma \vdash S \mathrel{\overset{\mathrm{stm}}{:}} \# & \leftrightarrow & S : (\Lambda \to \Theta)
\end{array}$$

We likewise extend the erasure operator $|-|$ (which converts a polarized formula to an unpolarized formula) to labelled hypotheses as follows:

$$\begin{array}{lll}
|x \mathrel{\overset{\mathrm{val}}{:}} X| = X & |\overline{u} \mathrel{\overset{\mathrm{cnt}}{:}} P| = |P| & |P\,\mathrm{val}| = |P| \\
|\overline{x} \mathrel{\overset{\mathrm{val}}{:}} \overline{X}| = \overline{X} & |u \mathrel{\overset{\mathrm{exp}}{:}} N| = |N| & |N\,\mathrm{cov}| = |N|
\end{array}$$

Finally, we call terms of **CU** *pure* if they do not use done. Clearly, we want to restrict our attention to pure terms in the proof of completeness.

3.5.2 Completeness

Theorem 27 (Focusing Completeness)**.** *If* $|\Lambda| \to^c |\Theta|$ *then there exists a pure* S *such that* $S : (\Lambda \to \Theta)$.

Proof. By induction on the classical derivation, with a side induction on the formulas in Λ and Θ. In all we have to consider each of the ten rules with either positive or negative polarization of the principal connective/atom/cut formula (i.e., 20 cases), plus the possible occurrence of the shift operators on either side of the polarized sequent (i.e., four cases). Fortunately, the positive and negative cases are entirely symmetric, and indeed all cases are simply routine computations. Below we illustrate a few of the positive cases.

Case: (*init*)
Take $S = \overline{u}\ x : (x \mathrel{\overset{\mathrm{val}}{:}} X, \Lambda \to \Theta, \overline{u} \mathrel{\overset{\mathrm{cnt}}{:}} X)$.

Case: (*cut*)
By the induction hypothesis we have:

$$S_1 : (\Lambda \to \Theta, \overline{u} \mathrel{\overset{\mathrm{cnt}}{:}} P) \quad \text{and} \quad S_2 : (P\,\mathrm{val}, \Lambda \to \Theta)$$

Take $S = [(\lambda p.S_2(p))/\overline{u}]S_1$. By Proposition 26 and the substitution lemma, we have $S : (\Lambda \to \Theta)$.

Case: ($\wedge R$)
By the i.h. we have:

$$S_1 : (\Lambda \to \Theta, \overline{u} \mathrel{\overset{\mathrm{cnt}}{:}} P \otimes Q, \overline{u}_1 \mathrel{\overset{\mathrm{cnt}}{:}} P) \ \text{ and } \ S_2 : (\Lambda \to \Theta, \overline{u} \mathrel{\overset{\mathrm{cnt}}{:}} P \otimes Q, \overline{u}_2 \mathrel{\overset{\mathrm{cnt}}{:}} Q)$$

We define S by:

$$S = [(\lambda p_1.[(\lambda p_2.\overline{u}\ (p_1, p_2))/\overline{u}_2]S_2)/\overline{u}_1]S_1$$

The computational gloss of this proof term: "Begin by executing S_1, until possibly a call to $\overline{u}_1$ is made—if so, remember the argument p_1 and execute S_2 until (possibly) a call to $\overline{u}_2$ is made with argument p_2. Now we have $p_1 \mathrel{\overset{\mathrm{val}}{:}} P$ and $p_2 \mathrel{\overset{\mathrm{val}}{:}} Q$, so we can finish by throwing (p_1, p_2) to $\overline{u}$". Note the choice of left-to-right evaluation order here is arbitrary. We could as well have taken

$$S = [(\lambda p_2.[(\lambda p_1.\overline{u}\ (p_1, p_2))/\overline{u}_1]S_1)/\overline{u}_2]S_2$$

Case: $(\neg R)$
By the i.h., we have $S_1 : (P\,\mathrm{val}, \Lambda \to \Theta, \overline{u} \stackrel{\mathrm{cnt}}{:} \stackrel{v}{\neg} P)$. Then take $S = \overline{u}\ (\lambda p.S_1(p))$.

Case: $(\vee L)$
We have $S_1 : (P\,\mathrm{val}, P \oplus Q\,\mathrm{val}, \Lambda \to \Theta)$ and $S_2 : (Q\,\mathrm{val}, P \oplus Q\,\mathrm{val}, \Lambda \to \Theta)$. We define S inductively on $(P \oplus Q)$-patterns, by $S(\mathsf{inl}(p_1)) = S_1(p_1)(\mathsf{inl}(p_1))$ and $S(\mathsf{inr}(p_2)) = S_2(p_2)(\mathsf{inr}(p_2))$.

Case: $(\downarrow N)$
Suppose we have $|\Lambda| \to^c |\Theta|, |\downarrow N|$ (and know nothing about the last rule applied). Then by definition $|\Lambda| \to^c |\Theta|, |N|$, and by the i.h. there exists a $S_1 : (\Lambda \to \Theta, N\,\mathrm{cov})$. Therefore we have $\overline{u}\ (\mu\overline{p}.S_1(\overline{p})) : (\Lambda \to \Theta, \overline{u} \stackrel{\mathrm{cnt}}{:} \downarrow N)$.

□

Letting $\vdash^c$ stand for classical truth, the following are immediate corollaries of focusing completeness.

Corollary 28 ("Glivenko's Theorem"). *if $\vdash^c |P|$ then there is a pure V such that $\vdash V \stackrel{val}{:} \downarrow\uparrow P$.*

Proof. Let $V = \mu\overline{u}.S$, where $S : (\cdot \to \overline{u} \stackrel{\mathrm{cnt}}{:} P)$. □

Corollary 29 (Dual intuitionistic truth = classical truth). *if $\vdash^c |N|$ then there is a pure E such that $\vdash E \stackrel{exp}{:} N$.*

Proof. Let $E = \mu\overline{p}.S(\overline{p})$, where $S : (\cdot \to N\,\mathrm{cov})$. □

4 Extending the language: preliminary results

We have *derived* the language **CU** through a Curry-Howard interpretation of polarized logic, and illustrated some of its expressive power to encode strict and lazy evaluation at the level of types. From this logical core, the language is amenable to extension in several different directions. Roughly, these classify as introducing new *effects* or new *types.* We give a quick sketch of both possibilities in order to point the way forward, although a thorough examination is not possible here.

Non-termination. Adding recursion is straightforward. For example, the following typing rule and reduction rule allow building recursive expressions:

$$\frac{\Gamma, u \stackrel{\mathrm{exp}}{:} N \vdash E \stackrel{\mathrm{exp}}{:} N}{\Gamma \vdash \mathsf{fix}\ u.E \stackrel{\mathrm{exp}}{:} N} \qquad (\mathsf{fix}\ u.E)\ C \rightsquigarrow ([(\mathsf{fix}\ u.E)/u]E)\ C$$

Non-determinism. We can add simple non-deterministic evaluation:

$$\frac{\Gamma \vdash S_1 \stackrel{\mathrm{stm}}{:} \#\quad \Gamma \vdash S_2 \stackrel{\mathrm{stm}}{:} \#}{\Gamma \vdash S_1 \parallel S_2 \stackrel{\mathrm{stm}}{:} \#} \qquad \begin{array}{l} S_1 \parallel S_2 \rightsquigarrow S_1 \\ S_1 \parallel S_2 \rightsquigarrow S_2 \end{array}$$

Both of these extensions introduce new effects into the language without breaking its essential properties. In particular, we can rely on the fact that these kinds of extensions do not change the *structure* of values and covalues (although they introduce new expressions and statements that can be *suspended* within values and covalues). Thus type safety extends in a modular way: the proof of progress requires one additional case for each new typing rule, and the proof of preservation one additional case for each new reduction rule.

Recursive types. Perhaps surprisingly, recursive types can be added with just a single pattern-typing rule:

$$\frac{\Delta \Rightarrow p \stackrel{\mathrm{val}}{:} [(\mathsf{rec}\ X.P)/X]P}{\Delta \Rightarrow \mathsf{fold}(p) \stackrel{\mathrm{val}}{:} \mathsf{rec}\ X.P}$$

Alternately, we can give patterns for specific recursive datatypes, for example natural numbers or infinite boolean streams:

$$\frac{}{\cdot \Rightarrow \mathsf{z} \mathrel{\vdots}^{\mathrm{val}} \mathbb{N}} \qquad \frac{\Delta \Rightarrow p \mathrel{\vdots}^{\mathrm{val}} \mathbb{N}}{\Delta \Rightarrow \mathsf{s}(p) \mathrel{\vdots}^{\mathrm{val}} \mathbb{N}}$$

$$\frac{}{u \mathrel{\vdots}^{\mathrm{exp}} \uparrow\mathsf{Stream} \Rightarrow \mathsf{cons}(\mathsf{t}, u) \mathrel{\vdots}^{\mathrm{val}} \mathsf{Stream}} \qquad \frac{}{u \mathrel{\vdots}^{\mathrm{exp}} \uparrow\mathsf{Stream} \Rightarrow \mathsf{cons}(\mathsf{f}, u) \mathrel{\vdots}^{\mathrm{val}} \mathsf{Stream}}$$

Since the proof of type safety is parametric in the pattern-typing judgment, we do not even have to check any additional cases! However, we can no longer rely on there being only finitely many patterns of any type. Type safety indeed did not rely on this fact. But it raises a question about how to interpret the higher-order rules,[17] which we will not attempt to answer here.

Example 19. We define an expression $\mathsf{bit} \mathrel{\vdots}^{\mathrm{exp}} \uparrow 2$ computing a non-deterministic boolean, and then $\mathsf{bits} \mathrel{\vdots}^{\mathrm{exp}} \uparrow\mathsf{Stream}$ as an infinite lazy stream of booleans:

$\mathsf{bit} = \mu\overline{u}.(\overline{u}\ \mathsf{t}) \parallel (\overline{u}\ \mathsf{f})$
$\mathsf{bits} = \mathsf{fix}\ u.\mu\overline{u}.\mathsf{bit}\ (\lambda b.\overline{u}\ \mathsf{cons}(b, u))$

■

Union and intersection types. As described in the introduction, part of the motivation for studying evaluation order from a logical perspective was to explain what we call *operationally-sensitive typing phenomena,* the differing behavior of unions and intersections under call-by-value and call-by-name evaluation. For example, Davies and Pfenning (2000) discovered that intersection introduction requires a value restriction under call-by-value with effects, constructing a counterexample to type safety analogous to the original bug in the implementation of ML polymorphism. They also discovered that the standard subtyping distributivity rule $(A \to B) \cap (A \to C) \leq A \to (B \cap C)$ (Barendregt et al., 1983) was unsound in that setting. Although union and intersection types are beyond the scope of this paper, our preliminary studies suggest that operationally-sensitive typing phenomena do indeed have a logical explanation. To give a preview of the results, just like products and sums, intersections and unions come in two polarities—one positive, one negative—and are defined purely through their (co)pattern-typing rules. Then, as in Section 3.3, we can attempt to *derive* rules for expression-typing at shifted type. We find that the usual union introduction rules are derivable:

$$\frac{\Gamma \vdash E \mathrel{\vdots}^{\mathrm{exp}} \uparrow P}{\Gamma \vdash E \mathrel{\vdots}^{\mathrm{exp}} \uparrow(P \cup Q)} \qquad \frac{\Gamma \vdash E \mathrel{\vdots}^{\mathrm{exp}} \uparrow Q}{\Gamma \vdash E \mathrel{\vdots}^{\mathrm{exp}} \uparrow(P \cup Q)}$$

But the unrestricted intersection introduction rule is *not:*

$$\frac{\Gamma \vdash E \mathrel{\vdots}^{\mathrm{exp}} \uparrow P \quad \Gamma \vdash E \mathrel{\vdots}^{\mathrm{exp}} \uparrow Q}{\Gamma \vdash E \mathrel{\vdots}^{\mathrm{exp}} \uparrow(P \cap Q)}\ ??$$

Likewise, we define subtyping by *generalizing* the identity principle: P is a subtype of Q if by any hypothesis $\overline{u} \mathrel{\vdots}^{\mathrm{cnt}} Q$ can be transformed by the identity coercion into a continuation of type P. Again, using this derived notion of subtyping we can explain the failure of unsound distributivity laws such as $\uparrow P \cap \uparrow Q \leq \uparrow(P \cap Q)$, while simultaneously validating sound principles such as ${}^{v}\!\downarrow P \cap {}^{v}\!\downarrow Q \leq {}^{v}\!\downarrow(P \cup Q)$. Polarity is important here! As duals to these principles, we *can* derive $\downarrow N \cap \downarrow M \leq \downarrow\!\downarrow(N \cap M)$ but not ${}^{n}\!\downarrow N \cap {}^{n}\!\downarrow M \leq {}^{n}\!\downarrow(N \cup M)$.

5 Related Work

This work builds on ideas developed by many different people in different areas of research. We present a brief survey below.

Refinement types and effects. Refinement types are practically-motivated extensions to type systems for existing languages, used to capture more program invariants (Freeman and Pfenning, 1991). Although this research drew from very old theoretical foundations on intersection types (Barendregt et al., 1983), those foundations were found to be shaky in this more operational setting: Davies and Pfenning (2000)

[17] For example, the derived type-checking rule for $\mathbb{N}$-continuations is a variant of the ω-rule (Buchholz et al., 1981).

found examples showing the necessity of a value restriction on intersection introduction, as well as the unsoundness of the standard subtyping distributivity rule for function types. The design of a refinement system with union types (Dunfield and Pfenning, 2004) uncovered further dissonance with prior theoretical studies (Barbanera et al., 1995).

Duality of computation. Beginning with Filinski's master's thesis (Filinski, 1989), a line of work has explored a duality between call-by-value and call-by-name evaluation in the presence of first-class continuations. Filinski was inspired by categorical duality; logical studies of this duality have been done largely in the setting of classical logic, particularly based upon the $\lambda\mu$-calculus (Parigot, 1992). For example, Curien and Herbelin (2000) define a programming language with control operators as a Curry-Howard interpretation of a sequent calculus for implicational classical logic, which Wadler (2003) extends to a "dual calculus" for propositional classical logic including conjunction, disjunction, negation, and implication defined by De Morgan translation. Both papers analyze the duality between call-by-value and call-by-name as one of alternative cut-reduction strategies—without explicitly using polarity to encode these strategies at the level of types. On the other hand, Curien and Herbelin (2000) come very close to defining dual focusing calculi as "well-behaved subsyntaxes" (cf. §5), as does Shan (2005) in an extension of Wadler's calculus. In even more recent work, Dyckhoff and Lengrand (2006) define a positive focusing system and use the implication fragment to encode call-by-value lambda calculus.

From a domain-theoretic perspective, Streicher and Reus (1998) give an interpretation of call-by-name based on *negative domains* and continuation semantics, an extension of earlier work with Lafont et al. (1993), which was itself inspired by Girard's polarized deconstruction of classical logic (see below). More "unified" (both positive and negative) perspectives are provided by Selinger's *control categories*, as well as Paul-Blain Levy's call-by-push-value language (Selinger, 2001; Levy, 2001). The latter maintains separate syntactic categories of "value types" and "computation types", which may be seen as a polarity distinction—a similar distinction has also been proposed by Filinski (2007). Recently, Levy (2006) defined a "jumbo" λ-calculus with pattern-matching constructs—essentially for their inversion properties—arguing that this is necessary to obtain the right type isomorphisms.

Polarity and focusing. The discovery of linear logic (Girard, 1987) and its crisp symmetries led to some early attempts at explaining strict vs. lazy evaluation within linear logic, for example by Abramsky (1993). Focusing (Andreoli, 1992) greatly improved the understanding of these symmetries, and sparked interest in the idea of polarity as a way of taming misbehaved logics (Girard, 1991, 1993). It was reconstructed in the "colour protocol" of Danos et al. (1997) for studying normalization of proofs, and used by Watkins et al. (2002) to define a dependent type theory with only canonical forms (and hence no need for $\beta\eta$ conversion rules). Recently the theory of polarity has been developed in greater depth, both in ludics (Girard, 2001) and in Olivier Laurent's dissertation (Laurent, 2002). Laurent (2005) pursues an analysis in various ways technically similar to ours, defining dual translations of classical logic into polarized linear logic, and then studying the different type isomorphisms induced by η-expansion in a purely logical setting.

Constructive negation and dual intuitionistic logic. As mentioned in Section 2.3, the notion of constructive refutation is due to Nelson (1949). The system defined in that paper (via a realizability interpretation) seems to correspond to the fragment of polarized logic with only the judgments *triv* and *absurd*, and negation defined as the $(-)^{\perp}$ connective. Dual intuitionistic logic (corresponding to the negative fragment of polarized logic) was first studied by Czermak (1977) and Goodman (1981) (the latter called it "anti-intuitionistic" logic). Again inspired by Girard's **LU**, Bellin and Biasi (2004) present a polarized logic combining intuitionistic and dual intuitionistic logic. Their analysis has a similar philosophical motivation as ours, viewing the two halves of the system as corresponding to dual, verificationist and pragmatist meaning-theories.

The logic of logic. While we have traced our judgmental reconstruction of polarity to Dummett's especially insightful account in "The Logical Basis of Metaphysics", many others have considered similar questions about the meaning and the justification of the logical laws. We gave some historical context for Dummett's analysis in Section 2.1, particularly his debt to Prawitz. Prawitz's inversion principle has seen a burst of renewed interest, following Jan von Plato's analysis of natural deduction and the framework of "general elimination rules" (Plato, 2001). While Dummett bases his notion of proof-theoretic justification on

Prawitz's, we saw that a major difference was his hereditary definition of canonical proof. This is what makes Dummett's analysis so closely related to focusing and pattern-matching. Coquand (1992) also draws this explicit connection to Dummett's work in the conclusion to a short paper on pattern-matching in dependent type theory, writing (without further elaboration), "From a proof-theoretic viewpoint, our treatment can be characterized as fixing the meaning of a logical constant by its introduction rules."

Of course another famous investigation of the justification of the logical laws is Martin-Löf's 1983 Siena Lectures (Martin-Löf, 1996), while Girard's recent monograph "Locus Solum" is an almost infamous one (Girard, 2001). Though we have profited greatly from the ideas developed in both these investigations, we have gone back to Dummett's not only because it was an early exposition of polarity, but so as to suggest his work as a surprising link between them.

Acknowledgments

This work would not have been possible without the many thoughtful suggestions and criticisms of my advisor Frank Pfenning. I have also greatly benefited from discussions with Steve Awodey, Peter Hancock, Bob Harper, Neel Krishnaswami, Peter Lee, William Lovas, Paul-André Melliès, Jason Reed, Rob Simmons, and Kevin Watkins. Thierry Coquand's paper pointed me towards Dummett's lectures, and he also suggested contacting Peter Hancock, who kindly provided me information about his work with Per Martin-Löf, as well as a copy of (Martin-Löf, 1976). Peter Dybjer also tracked down and mailed me a copy of (Hancock and Martin-Löf, 1975). Finally, I would like to gratefully acknowledge the back-breaking labor of the anonymous referees, in helping me to improve previous versions of this paper.

References

Abramsky, S., 1993. Computational interpretations of linear logic. Theoretical Computer Science 111 (1–2), 3–57.

Andreoli, J.-M., 1992. Logic programming with focusing proofs in linear logic. Journal of Logic and Computation 2 (3), 297–347.

Andreoli, J.-M., 2001. Focussing and proof construction. Annals of Pure and Applied Logic 107 (1), 131–163.

Barbanera, F., Dezani-Ciancaglini, M., De'Liguoro, U., 1995. Intersection and union types: syntax and semantics. Information and Compution 119 (2), 202–230.

Barendregt, H., Coppo, M., Dezani-Ciancaglini, M., 1983. A filter lambda model and the completeness of type assignment. The Journal of Symbolic Logic 48 (4), 931–940.

Bellin, G., Biasi, C., 2004. Towards a logic for pragmatics: Assertions and conjectures. Journal of Logic and Computation 14 (4), 473–506.

Buchholz, W., Feferman, S., Pohlers, W., Sieg, W., 1981. Iterated Inductive Definitions and Subsystems of Analysis: Recent Proof-Theoretical Studies. Springer-Verlag.

Coquand, T., 1992. Pattern matching with dependent types. In: Proceedings of the Workshop on Types for Proofs and Programs. Båstad, Sweden, pp. 71–83.

Curien, P.-L., Herbelin, H., 2000. The duality of computation. In: ICFP '00: Proceedings of the SIGPLAN International Conference on Functional Programming. pp. 233–243.

Czermak, J., 1977. A remark on Gentzen's calculus of sequents. Notre Dame Journal of Formal Logic 18, 471–474.

Danos, V., Joinet, J.-B., Schellinx, H., 1997. A new deconstructive logic: Linear logic. The Journal of Symbolic Logic 62 (3), 755–807.

Davies, R., Pfenning, F., 2000. Intersection types and computational effects. In: ICFP '00: Proceedings of the SIGPLAN International Conference on Functional Programming. pp. 198–208.

Dummett, M., 1991. The Logical Basis of Metaphysics. The William James Lectures, 1976. Harvard University Press, Cambridge, Massachusetts.

Dunfield, J., Pfenning, F., 2004. Tridirectional typechecking. In: POPL '04: Proceedings of SIGPLAN-SIGACT Symposium on Principles of Programming Languages. pp. 281–292.

Dyckhoff, R., Lengrand, S., 2006. LJQ: A strongly focused calculus for intuitionistic logic. In: Proceedings of the Second Conference on Computability in Europe.

Filinski, A., 1989. Declarative continuations and categorical duality. Master's thesis, University of Copenhagen, Computer Science Department.

Filinski, A., 2007. On the relations between monadic semantics. Theoretical Computer Science 375 (1-3), 41–75.

Freeman, T., Pfenning, F., 1991. Refinement types for ML. In: PLDI '91: Proceedings of the SIGPLAN Conference on Programming Language Design and Implementation. pp. 268–277.

Gentzen, G., 1935. Untersuchungen über das logische Schließen. Mathematische Zeitschrift 39, 176–210, 405–431, English translation in M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pages 68–131, North-Holland, 1969.

Girard, J.-Y., 1987. Linear logic. Theoretical Computer Science 50 (1), 1–101.

Girard, J.-Y., 1991. A new constructive logic: Classical logic. Mathematical Structures in Computer Science 1, 255–296.

Girard, J.-Y., 1993. On the unity of logic. Annals of pure and applied logic 59 (3), 201–217.

Girard, J.-Y., 2001. Locus solum: From the rules of logic to the logic of rules. Mathematical Structures in Computer Science 11 (3), 301–506.

Goodman, N. D., 1981. The logic of contradiction. Zeitschrift für mathematische Logik und Grundlagen der Mathematik 27 (2), 119–126.

Griffin, T. G., 1990. The formulae-as-types notion of control. In: POPL '90: Proceedings of the SIGPLAN-SIGACT Symposium on Principles of Programming Languages. pp. 47–57.

Hancock, P., Martin-Löf, P., 1975. Syntax and semantics of the language of primitive recursive functions. Tech. Rep. 3, University of Stockholm, Stockholm, Sweden.

Harper, B., Lillibridge, M., 1991. ML with callcc is unsound. Post to TYPES mailing list, July 8, 1991.

Heyting, A., 1974. Mathematische Grundlagenforschung, Intuitionismus, Beweistheorie. Springer, Berlin.

Kleene, S. C., 1952. Introduction to Metamathematics. Van Nostrand, Princeton, NJ.

Kolmogorov, A. N., 1932. Zur Deutung der intuitionistischen Logik. Mathematischen Zeitschrift 35, 58–65.

Lafont, Y., Reus, B., Streicher, T., 1993. Continuation semantics or expressing implication by negation. Tech. Rep. 93-21, University of Munich.

Laurent, O., Mar. 2002. Etude de la polarisation en logique. Thèse de doctorat, Université Aix-Marseille II.

Laurent, O., Oct. 2005. Classical isomorphisms of types. Mathematical Structures in Computer Science 15 (5), 969–1004.

Levy, P. B., 2001. Call-by-push-value. Ph.D. thesis, Queen Mary, University of London.

Levy, P. B., 2006. Jumbo λ-calculus. In: Proceedings of the 33rd International Colloquium on Automata, Languages and Programming, Venice, 2006. Vol. 4052 of Lecture Notes in Computer Science.

Martin-Löf, P., 1971. *Hauptsatz* for the intuitionistic theory of iterated inductive definitions. In: Fenstad, J. E. (Ed.), Proceedings of the Second Scandinavian Logic Symposium. North Holland, Amsterdam, pp. 179–216.

Martin-Löf, P., 1976. Letter to Michael Dummett, dated 5 March, 1976, including lecture notes transcribed by Peter Hancock. Copy received from Peter Hancock.

Martin-Löf, P., 1996. On the meanings of the logical constants and the justifications of the logical laws. Nordic Journal of Philosophical Logic 1 (1), 11–60.

Milner, R., Tofte, M., Harper, R., MacQueen, D., 1997. The Definition of Standard ML, Revised edition. MIT Press.

Moggi, E., 1991. Notions of computation and monads. Information and Compution 93 (1), 55–92.

Nelson, D., 1949. Constructible falsity. Journal of Symbolic Logic 14 (1), 16–26.

Parigot, M., 1992. $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. In: LPAR '92: Proceedings of the International Conference on Logic Programming and Automated Reasoning. pp. 190–201.

Pfenning, F., Davies, R., 2001. A judgmental reconstruction of modal logic. Mathematical Structures in Computer Science 11 (4), 511–540.

Plato, J. v., 2001. Natural deduction with general elimination rules. Archive for Mathematical Logic 40 (7).

Plotkin, G. D., 1975. Call-by-name, call-by-value and the λ-calculus. Theoretical Computer Science 1, 125–159.

Prawitz, D., 1974. On the idea of a general proof theory. Synthese 27, 63–77.

Reynolds, J. C., 1972. Definitional interpreters for higher-order programming languages. In: ACM '72: Proceedings of the ACM annual conference. pp. 717–740.

Selinger, P., 2001. Control categories and duality: on the categorical semantics of the lambda-mu calculus. Mathematical Structures in Computer Science 11 (2), 207–260.

Shan, C., 2005. A computational interpretation of classical S4 modal logic. In: IMLA '05: Intuitionistic Modal Logics and Applications Workshop.

Streicher, T., Reus, B., Nov. 1998. Classical logic, continuation semantics and abstract machines. Journal of Functional Programming 8 (6), 543–572.

Troelstra, A. S., Schwichtenberg, H., 1996. Basic Proof Theory. Vol. Cambridge Tracts in Theoretical Computer Science 43. Cambridge University Press.

Wadler, P., 2003. Call-by-value is dual to call-by-name. In: ICFP '03: Proceedings of the SIGPLAN International Conference on Functional Programming. pp. 189–201.

Watkins, K., Cervesato, I., Pfenning, F., Walker, D., 2002. A concurrent logical framework I: Judgments and properties. Tech. Rep. CMU-CS-02-101, Department of Computer Science, Carnegie Mellon University, revised May 2003.

Wittgenstein, L., 1974. Philosophical Grammar. Blackwell, Oxford.

Wright, A. K., 1995. Simple imperative polymorphism. Lisp and Symbolic Computation 8 (4), 343–355.