Unifying Classical and Intuitionistic Logics for Computational Control

Chuck Liang, Dale Miller

Unifying Classical and Intuitionistic Logics for Computational Control

Chuck Liang
Department of Computer Science
Hofstra University
Hempstead, NY, USA
Chuck.C.Liang@hofstra.edu

Dale Miller
INRIA-Saclay &
LIX, École Polytechnique
Palaiseau, France
Dale.Miller@inria.fr

***Abstract*—We show that control operators and other extensions of the Curry-Howard isomorphism can be achieved without collapsing all of intuitionistic logic into classical logic. For this purpose we introduce a unified propositional logic using polarized formulas. We define a Kripke semantics for this logic. Our proof system extends an intuitionistic system that already allows multiple conclusions. This arrangement reveals a greater range of computational possibilities, including a form of dynamic scoping. We demonstrate the utility of this logic by showing how it can improve the formulation of exception handling in programming languages, including the ability to distinguish between different kinds of exceptions and constraining when an exception can be thrown, thus providing more refined control over computation compared to classical logic. We also describe some significant fragments of this logic and discuss its extension to second-order logic.**

I. Introduction

Since it became clear that the Curry-Howard isomorphism can be extended to classical logic [1], several *constructive* classical systems have been formulated, including $\lambda\mu$-calculus [2] and its variants. However, the gap between intuitionistic logic and classical logic is a large one. Collapsing all of intuitionistic logic into classical logic in order to obtain greater computational ability is not necessarily the best approach. For example, we may wish to obtain a computationally meaningful proof for a formula such as $(A \to B) \vee A$. In classical logic, there is no avoiding the fact that this formula is an instance of the excluded middle, and can be given a trivial proof. Such a situation can occur either with an explicit $\vee$, or implicitly when there are multiple conclusions in proofs. A $\lambda\mu$-term may be obtained, but it does not compute anything meaningful. We wish to have a *logical* way to constrain the interpretation of such a formula, so that B must be *computed,* in the sense that it must be the result of introductions and eliminations and not just the result of weakening, at least *not until certain conditions are met.*

Ideas from classical $\lambda\mu$-calculus have been used to formulate, for example, *exception handling* in programming languages. In particular, *catch* is represented by contraction, which saves a copy of the current continuation, and *throw/raise* is represented by weakening, which discards the current continuation and jumps back to the saved one (see [3]). However, there remains a gap between this form of exception handling and those of practical programming languages such as Java. We list three such gaps below.

- The *type* of the exception is not identified. In Java, for example, one specifies exceptions such as in *catch IOException, catch ArithmeticException,* etc.
- Control over when an exception can be thrown is lacking. One must declare *int p(..) throws IOException* in Java if it is possible for p to throw this kind of exception.
- Exception handling in languages such as Java and ML are *dynamically scoped,* whereas the $\lambda\mu$-calculus has the same variable capturing restrictions as λ-calculus.

We do not claim to have a system as rich as Java or ML but will offer at least some progress in each of these areas. We unify classical and intuitionistic propositional logics into a new logic we call *Polarized Control Logic* (PCL). The polarity of a formula determines the availability of certain structural rules. Depending on the polarity of A and B, $(A \to B) \vee A$ may be interpreted entirely classically, entirely intuitionistically, *or somewhere in between.*

Such a use of polarization is not new, and much of what we have just described may sound like LC [4] and related work on *focusing proofs* [5]. In fact our system will extend the polarization scheme of LC. It is based on the preservation of *intuitionistic implication,* which LC does not contain. PCL will contain not only intuitionistic and classical logics but also other meaningful fragments, including an intermediate logic that admits the law of excluded middle $A \vee \neg A$ but *not* involutive negation $\neg\neg A \to A$.

We will define a form of Kripke models to interpret polarized formulas and cement PCL's status as a new logic with a sequent calculus and proofs of cut elimination, soundness, and completeness. The main proof system for PCL will be a natural deduction system with proof terms. Existing constructive classical systems typically extend Gentzen-style, single-conclusion intuitionistic systems with multiple conclusions. In contrast, the proof system for PCL will extend an intuitionistic system that already has multiple conclusions. This approach will reveal a greater range of computational capabilities, including a dynamically scoped

version of the μ binder, the operator of $\lambda\mu$-calculus that enables a computationally meaningful form of contraction.

II. Polarization and Semantics

We focus on propositional logic in this presentation. Formulas of PCL are build from atomic formulas, the connectives $\wedge$, $\vee$ and $\rightarrow$, and the constants 1, 0 and $\bot$. The two constants for *false* define two forms of negation. We use the abbreviations $\sim A = A \rightarrow 0$, and $\neg A = A \rightarrow \bot$.

A. Polarization

Atomic formulas are polarized *red or green.* Polarization is extended to all formulas as follows:

- $\bot$ is green; 0 and 1 are red.
- $A \wedge B$ is green if A and B are green, otherwise, it is red.
- $A \vee B$ is green if A or B is green, otherwise, it is red.
- $A \rightarrow B$ is green if B is green, otherwise, it is red.

We use the symbol R exclusively to represent red formulas and E to represent green formulas, with frequent reminders of this convention.

The polarization of $\wedge$ and $\vee$ mirrors that of the LC system [4]. However, LC does not contain implication[1]. In particular, the natural correspondence between polarization and focusing (focalization) no longer holds with implication. For these reasons we have named our polarities red and green instead of "positive" and "negative." We also do not insist on an *a priori* relationship of duality between the polarities. For example, $\neg E$ is green even if E is green.

Conceptually, red means intuitionistic and green means classical and we formalize this correspondence below.

B. Kripke Models

We consider only Kripke frames that are (finitely) *rooted:* it is known that intuitionistic propositional models can also be assumed to have this restriction. Such frames are the basis of models of the form $\langle \mathbf{W}, \mathbf{r}, \preceq, \models \rangle$, where $\preceq$ is a transitive, reflexive ordering of the set of possible worlds $\mathbf{W}$ and $\mathbf{r} \in \mathbf{W}$ is the unique root such that $\mathbf{r} \preceq u$ for all $u \in \mathbf{W}$. The binary relation $\models$ monotonically maps elements of $\mathbf{W}$ to sets of atomic formulas. We add another stipulation:

> If q is a world properly above $\mathbf{r}$ ($\mathbf{r} \prec q$), then $q \models e$ holds for all *green* atoms e.

The $\models$ relation is extended to all formulas as follows. Here we use u and v to represent arbitrary worlds.

- $\mathbf{r} \not\models \bot$
- $q \models \bot$ for all $q \succ \mathbf{r}$
- $u \models 1$; $u \not\models 0$
- $u \models A \wedge B$ iff $u \models A$ and $u \models B$
- $u \models A \vee B$ iff $u \models A$ or $u \models B$
- $u \models A \rightarrow B$ iff for all $v \succeq u$, $v \models A$ implies $v \models B$.

Except for green atoms and $\bot$, the $\models$ relation is the same as in Kripke models for intuitionistic logic. The usual monotonicity property holds: if $u \preceq v$ then $u \models A$ implies $v \models A$ for all formulas A.

We shall refer to this version of Kripke models as r-*models.* A formula is considered valid in an r-model if it is valid in all its worlds. A formula is valid in PCL if it is valid in all r-models.

The semantic characterization of green formulas is consistent with that of $\bot$ and green atoms: $q \models E$ for all green E and worlds $q \succ \mathbf{r}$. The root $\mathbf{r}$ is the only *classically consistent* world. Since all worlds above r force $\neg A$, we have $\mathbf{r} \models A$ if and only if $\mathbf{r} \not\models \neg A$, and thus $A \vee \neg A$ *is valid in all worlds.*

If a formula A is purely red (all subformulas are red), then A *is valid if and only if it is intuitionistically valid.* The root world balances the classical inconsistency of all others above. All worlds are intuitionistically consistent. This means that a red subformula of a green formula will retain its intuitionistic meaning, and vice versa. The r-models are a restricted class of the models that we formulated in [6], but PCL retains intuitionistic implication in a stronger way.

The following properties hold of this semantics.

- $\bot \rightarrow \mathbf{E}$ is valid for green $\mathbf{E}$, $\mathbf{0} \rightarrow \mathbf{A}$ is valid for all $\mathbf{A}$.
- $\mathbf{A} \vee \neg \mathbf{A}$ is valid, but not $\sim A \vee A$ nor $\sim E \vee E$.
- $\neg\neg \mathbf{A} \rightarrow \mathbf{A}$ is **NOT** valid.
- $\neg\neg \mathbf{E} \rightarrow \mathbf{E}$ is valid for green $\mathbf{E}$.
- $\sim\neg \mathbf{A} \rightarrow \mathbf{A}$ is valid for all $\mathbf{A}$.
- If $\neg\neg \mathbf{A}$ is valid then $\mathbf{A}$ is also valid.
- $\neg(\mathbf{A} \wedge \mathbf{B}) \rightarrow (\neg \mathbf{A} \vee \neg \mathbf{B})$
- $((\mathbf{P} \rightarrow \mathbf{Q}) \rightarrow \mathbf{P}) \rightarrow \mathbf{P}$ is valid if $\mathbf{Q}$ is green.

These properties show, respectively, that there are two levels of consistency. The excluded middle is valid with $\neg$, but not with $\sim$. The following countermodel disproves $\neg\neg A \rightarrow A$: let there be a single world $q \succ \mathbf{r}$. For some *red* atom a, define $\mathbf{r} \not\models a$ and $q \not\models a$. Then there is a world q above $\mathbf{r}$ such that $q \models \neg\neg a$ but $q \not\models a$. However, two forms of negation means four of double negation, and one of these gives us the $\mathcal{C}$ control operator. The *admissible rule* stated does not contradict the non-involution of $\neg$. The De Morgan laws are valid with $\neg$. The version of Peirce's formula exemplifies how intuitionistic logic need not be crushed to obtain new computational ability (in this case $call/cc$). If P is red, then the innermost $\rightarrow$ is a classical implication while the other two stay intuitionistic.

The reader may have noticed, from the properties described above, that red atoms have the characteristics of universally quantified second order variables and that the green atoms are existentially quantified. This is indeed true. Although our main focus is on propositional logic, we discuss this further in Section VII.

[1] If $A^{\bot} \vee B$ is used for implication in LC (where $A^{\bot}$ is the De Morgan dual of A), then it is positive only when A is negative and B is positive. In PCL, only the *head* of the implication determines the polarity. This treatment of intuitionistic implication is also different from *LU* and from linear logic. PCL is not a fragment of these systems.

III. Natural Deduction and Proof Terms

The usual route of extending the Curry-Howard isomorphism is to start with a Gentzen-style, single-conclusion intuitionistic system (LJ/NJ), allow multiple conclusions, then add a *stoup,* i.e., some distinguished formula that represents the current formula being deduced. Our approach, which is used because one of our structural rules is *context sensitive,* starts with the multiple-conclusioned version of intuitionistic sequent calculus. This system is found in various sources and is a mirror image of the Beth-Fitting intuitionistic tableaux (and thus also simplifies the proof of semantic completeness). Two inference rules in this system stand out in contrast to Gentzen's LJ, namely

$$\frac{A,\Gamma\vdash B}{\Gamma\vdash A\to B,\Theta}\quad\text{and}\quad\frac{\Gamma\vdash A,B,\Theta}{\Gamma\vdash A\vee B,\Theta}.$$

Contraction and weakening are also allowed on the right-hand side. However, it is easy to see why $(A\to\bot)\vee A$ is still not provable with the *scoping* restriction on $\to$ introduction. We can form a natural deduction system from this sequent calculus as follows.

1) Introduce a stoup on the right-hand side but also allow formulas to freely enter and leave the stoup, so that the new system is obviously equivalent to the stoupless one. In particular, admissible cuts are preserved.
2) Apply cut-elimination to form the elimination rules. The only extra difficulty, compared with the LJ-NJ translation, is in the $\to$ introduction and $\vee$ elimination cases. Fortunately, the *invertibility* of the multiplicative $\vee$-introduction rules (which can be proved by cut-elimination) can be used to combine the right-hand side into a single formula. Alternatively, we can just show that the elimination rules are semantically sound.

We now extend this system with an extra context of formulas on the right-hand side. Sequents of the proof system *NPC* for PCL are of the form $\Gamma\vdash\mathcal{S};\Theta;[\Delta]$, where the stoup $\mathcal{S}$ consists of zero or one formula. An empty stoup means 0, which is intuitionistic *false.* The variables Γ, Θ and Δ range over sets of formulas and the expression A,Γ, where Γ is any set of formulas, represents $\{A\}\cup\Gamma$ and does not preclude the possibility that $A\in\Gamma$. Intuitively, Θ is the intuitionistic multiple-conclusion context while $[\Delta]$ represents the "locked" context which is only accessible once a green formula appears in Θ.

The semantic interpretation of the sequent $\Gamma\vdash\mathcal{S};\Theta;[\Delta]$ is as for the formula $(\neg\check{\Delta}\wedge\hat{\Gamma})\to(\mathcal{S}\vee\check{\Theta})$. Here, $\hat{\Gamma}$ is the $\wedge$-conjunction of formulas in Γ, with an empty Γ representing 1, while $\check{\Theta}$ (and $\check{\Delta}$) is the $\vee$-disjunction of formulas in Θ, with an empty Θ, and an empty stoup, representing 0.

Except for the stoup formula, all other formulas in a sequent are indexed by variables. We assume that variables indexing formulas in Γ (λ variables) are distinguishable from those in Θ and Δ (μ variables) and that the usual variable conventions are applied. NPC is found in Figure 1. We prefer to associate a proof term with an entire subproof, and not just the stoup formula.

The following admissible rule indicates that contraction and weakening are available in Γ, Θ and $[\Delta]$:

$$\frac{s:C^u,C^v,\Gamma\vdash\mathcal{F};A^x,A^y,\Theta;[B^d,B^e,\Delta]}{s[u/v,x/y,d/e]:C^u,\Gamma\Gamma'\vdash\mathcal{F};A^x,\Theta\Theta';[B^d,\Delta\Delta']}$$

This is the uninteresting form of contraction and weakening and we usually use them implicitly. With these structural rules available, we are free to choose between multiplicative and additive treatments of contexts. We prefer multiplicative for its convenience when writing proofs.

A formula A is provable if $\vdash A;;[\,]$ is provable.

The proof terms for $Open$ and Esc are those found in $\lambda\mu$-calculus. However, here *they remain intuitionistic.* The μ-binder is still important since it allows for structural reduction (rule $(\mu d.s)t\longrightarrow\mu d.s\{[d]wt/d[w]\}$ remains valid), but with μ alone the reduction would be shallow since it cannot survive past a λ binder. Formulas can be *locked* in the boxed context $[\Delta]$ prior to an $\to$ introduction (regarding proofs bottom-up), which can be *unlocked* only if a green formula E is found outside of the box. In NPC the separation between intuitionistic and non-intuitionistic proofs is indicated by the presence of ?, without which ! would also be useless (with apologies to linear logic: there is no direct connection). The ! operators replaces μ as the most meaningful operator since $Lock$ enables contraction (B^x may persist in Θ) between two distinct contexts.

$Lock$ and $Unlock$ can be used to prove $\neg A\vee A$: since $\bot$ is green, it can unlock the copy of A saved inside $[\,]$. A proof of Peirce's formula $((P\to Q)\to P)\to P$, where Q *is green* and P is red, is found in Figure 2. Some steps moving formulas in and out of the stoup were combined.

Choosing a multiple-conclusion intuitionistic proof system as starting point does imply some non-determinism: the formula in the stoup is not always computed by introductions and eliminations. But this form of non-determinism does not survive past a β-redex (it also does not survive a $\vee$-elimination), so it is limited in form. It is still possible to derive typed λ-terms from purely intuitionistic proofs in NPC. In classical $\lambda\mu$ calculus, the μ binder can be considered as a constant of type $\neg\neg A\to A$, which has no equivalent λ-term. However, when seen in an intuitionistic context, μ has type $(0\vee A)\to A$ (and the contraction form of μ would be $A\vee A\to A$). It is ! that is non-intuitionistic. Recall that formulas B inside $[\,]$ has the meaning of $\neg B$ on the left-hand side. This means !, as a constant, has type $(\neg B\to B)\to B$ (assuming $B^x\in\Theta$), which is a version of Peirce's formula.

We can also translate classical $\lambda\mu$-terms into "$\lambda\mu$!-terms." Each instance of a classical $\mu d.s$ translates to $\mu d.!d.s$, and $[d]s$ translates to $?d.[d]s$. Classical $\lambda\mu$-terms thus translated are typable in NPC because we can choose green formulas

$$\frac{s : \Gamma \vdash ; A^x, \Theta; [\Delta]}{\mu x.s : \Gamma \vdash A; \Theta; [\Delta]}\ Open \qquad \frac{s : \Gamma \vdash A; \Theta; [\Delta]}{[d]s : \Gamma \vdash ; A^d, \Theta; [\Delta]}\ Esc \qquad \frac{s : \Gamma \vdash \mathcal{F}; \Theta; [B^x, \Delta]}{!x.s : \Gamma \vdash \mathcal{F}; \Theta, B^x; [\Delta]}\ Lock \qquad \frac{s : \Gamma \vdash E; \Theta, B^x; [\Delta]}{?x.s : \Gamma \vdash E; \Theta; [B^x, \Delta]}\ Unlock$$

$$\frac{s : A^x, \Gamma \vdash B; ; [\Delta]}{\lambda x.s : \Gamma \vdash A \to B; \Theta; [\Delta]} \to I \qquad \frac{s : \Gamma \vdash A \to B; \Theta; [\Delta] \quad t : \Gamma' \vdash A; \Theta'; [\Delta']}{(s\ t) : \Gamma\Gamma' \vdash B; \Theta\Theta'; [\Delta\Delta']} \to E$$

$$\frac{u : \Gamma \vdash A; \Theta; [\Delta] \quad v : \Gamma' \vdash B; \Theta'; [\Delta']}{(u, v) : \Gamma\Gamma' \vdash A \wedge B; \Theta\Theta'; [\Delta\Delta']} \wedge I \qquad \frac{s : \Gamma \vdash A \wedge B; \Theta; [\Delta]}{\pi_\ell(s) : \Gamma \vdash A; \Theta; [\Delta]} \wedge E_1 \qquad \frac{s : \Gamma \vdash A \wedge B; \Theta; [\Delta]}{\pi_r(s) : \Gamma \vdash B; \Theta; [\Delta]} \wedge E_2$$

$$\frac{s : \Gamma \vdash A; B^d, \Theta; [\Delta]}{\omega^\ell d.s : \Gamma \vdash A \vee B; \Theta; [\Delta]} \vee I_1 \qquad \frac{s : \Gamma \vdash B; A^d, \Theta; [\Delta]}{\omega^r d.s : \Gamma \vdash A \vee B; \Theta; [\Delta]} \vee I_2$$

$$\frac{s : \Gamma_1 \vdash A \vee B; \Theta; [\Delta_1] \quad u : A^x, \Gamma_2 \vdash C; ; [\Delta_2] \quad v : B^y, \Gamma_3 \vdash C; ; [\Delta_3]}{(\lambda x.u, \lambda y.v)s : \Gamma_1\Gamma_2\Gamma_3 \vdash C; \Theta; [\Delta_1\Delta_2\Delta_3]} \vee E$$

$$\frac{}{x : A^x, \Gamma \vdash A; \Theta; [\Delta]}\ Id \qquad \frac{s : \Gamma \vdash 0; \Theta; [\Delta]}{\mathcal{A}(s) : \Gamma \vdash \mathcal{F}; \Theta; [\Delta]}\ 0\ E \qquad \frac{s : \Gamma \vdash \bot; \Theta; [\Delta]}{\mathcal{B}(s) : \Gamma \vdash E; \Theta; [\Delta]} \bot E \qquad \frac{}{exit : \Gamma \vdash 1; \Theta; [\Delta]}\ 1I$$

Figure 1. The Natural Deduction System NPC; E in $Unlock$ and $\bot E$ must be green

$$\dfrac{\dfrac{}{x : ((P \to Q) \to P)^x \vdash (P \to Q) \to P; ; [\,]}\ Id \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{y : ((P \to Q) \to P)^x, P^y \vdash P; Q^e; [\,]}\ Id}{\mu e.[d]y : ((P \to Q) \to P)^x, P^y \vdash Q; P^d; [\,]}\ Open,\ Esc}{?d.\mu e.[d]y : ((P \to Q) \to P)^x, P^y \vdash Q; ; [P^d]}\ Unlock}{\lambda y.?d.\mu e.[d]y : ((P \to Q) \to P)^x \vdash P \to Q; P^d; [P^d]} \to I}{x\ (\lambda y.?d.\mu e.[d]y) : ((P \to Q) \to P)^x \vdash P; P^d; [P^d]} \to E}{!d.(x\ (\lambda y.?d.\mu e.[d]y)) : ((P \to Q) \to P)^x \vdash P; P^d; [\,]}\ Lock}{\mu d.[d]!d.(x\ (\lambda y.?d.\mu e.[d]y)) : ((P \to Q) \to P)^x \vdash P; ; [\,]}\ Open,\ Esc}{\lambda x.\mu d.[d]!d.(x\ (\lambda y.?d.\mu e.[d]y)) : \ \vdash ((P \to Q) \to P) \to P; ; [\,]} \to I$$

Figure 2. NPC proof of Peirce's formula with P red and Q green

to type them, such as $\neg\neg E \to E$[2]. Choosing only green formulas renders the guard on $Unlock$ meaningless, leaving us with classical logic.

The ! operator is *not* a binder in the sense of λ and μ: x *remains free in* $!x.s$ (and in $?x.s$). However, $!x.s$ does represent the start of a kind of *scope,* represented by s, in which x can be considered to have been *pushed on to an alternate, global stack.* Since x remains free, we would obviously be able to emulate a form of *capturing substitution* with respect to !. In particular, we may have $!x.\lambda y.s$ (typed by $A \to B$ in the stoup), and thus a valid reduction rule for ! is: $(!x.s)\ t \longrightarrow\ !x.(s\ t)$. Such a rule will "capture" free occurrences of x in t inside the scope of !. This is clearly the behavior of dynamic scoping. Nested occurrences of $!x$ may not seem necessary when building a cut-free proof, but they may certainly appear in proofs with cuts (β redexes), which represent programs.

To be fair, it is also possible to give such an interpretation to $?x.s$, and even to $[d]s$. At least the following special cases of μ also do not need to bind their variables:

$$\frac{s : \Gamma \vdash ; A^d, \Theta; [\Delta]}{\mu' d.s : \Gamma \vdash A; A^d, \Theta; [\Delta]} \qquad \frac{s : \Gamma \vdash ; A^d, \Theta; [A^d, \Delta]}{\mu'' d.s : \Gamma \vdash A; \Theta; [A^d, \Delta]}$$

The μ' operator is not needed in classical $\lambda\mu$-calculus because it can be replaced by weakening, but it is a *possible* interpretation of μ, which also leads to some dynamic binding behavior. However, μ' cannot implement *contraction,* and therefore does not have the computational power of !. In contrast, since $B^x \in \Theta$ is possible in the $Lock$ rule, ! can be interpreted as a dynamic binder *and* implements contraction. It is also not just the kind of contraction that renames variables. This comparison shows that the dynamic behavior of ! is not just the result of a fortuitous interpretation. The multiple-conclusion intuitionistic context, when paired with the [ ] context, thus revealed a new computational capability.

Since μ' and μ'' can extend the effectiveness of ! as a dynamic binder, we shall admit them as well.

A. An Abstract Machine

We do not give a complete set of reduction/rewrite rules for NPC proof terms because an effective computational

[2]Not all versions of classical $\lambda\mu$ calculus can be translated: the original $\lambda\mu$ allowed proofs of theorems to contain free variables, which is not possible in NPC.

interpretation of these terms requires a specific evaluation strategy. In the original $\lambda\mu$ calculus, certain valid reductions were also excluded to preserve confluence. The problem is exacerbated by the non-binding behavior of !: a specific reduction strategy is required for determinism and for type soundness (there are situations where *at most* one of several possible reductions may take place). We define such a strategy in the form of an abstract machine. This machine is not the only way to evaluate NPC terms but it demonstrates their computational effectiveness.

Krivine's machine for λ-terms, which computes weak head normal forms, was extended to $\lambda\mu$ by adding an additional environment for saved continuations [7], [8]. We also require a *global* environment to realize the dynamic binding behavior of !. Most of the following definitions are inherited from the $\lambda\mu$ machine. A machine state is defined inductively as $St; [GE]$, where St is a *stack* that consists of a list of *closures.* A closure $\langle t, CE, LE\rangle$ consists of a $\lambda\mu!$-term t, a *closure environment* CE, and a (local) *stack environment* LE. A closure environment consists of a list of bindings (x, C) where x is a (λ) variable and C is a closure. A stack environment consists of a list of bindings (d, S) where d is a (μ) variable and S is a stack. $[GE]$ is the global stack environment. Conceptually, a stack represents a sequence of cuts. A closure represents a subproof with pending cuts suspended in closure and stack environments. Each transition of the machine represents a cut reduction step.

We use nil for the empty list, :: for *Cons*, and assume that $CE(x)$, $LE(x)$ and $GE(x)$ return the first (from left to right) closure or stack bound to x in a closure or stack environment. The transitions of the machine are found in Figure 3. Only the implicational fragment is shown.

The highlighted rules represent divergence from classical $\lambda\mu$. The machine interprets μ as the non-binding μ'' when possible, which means that all nested $!x$ will behave dynamically. Only the outermost μx will bind statically.

B. Multiplicative Disjunction

It is quite natural to represent a multiplicative $\vee$-introduction, in the presence of a stoup, using binders: here, ω^ℓ and ω^r. Additive disjunction is subsumed in this representation by $\omega^{\ell/r} d.s$ where d is not free is s. In such a case $\omega^{\ell/r}$ naturally degrade to the usual *injection* operators. But non-vacuous multiplicative disjunction has the potential for greater computational content. In the $\vee$-elimination rule, both implicit λ-terms are needed in the reduction. The principal reduction rules associated with $\vee$-elimination are

$$\begin{array}{rcl}(u, v)\ (\omega^\ell d.t) & \longrightarrow & \mu d.[d](u\ t\{[d](v\ w)/[d]w\}) \\ (u, v)\ (\omega^r d.t) & \longrightarrow & \mu d.[d](v\ t\{[d](u\ w)/[d]w\})\end{array}$$

Both u and v are involved in the reduction. In the terminology of $\lambda\mu$-calculus, one is *logical* while the other one is *structural,* and is applied to subproofs where the alternate disjunct enters the stoup. Since both u and v are required, an extra copy of C will be found in the conclusion, and thus a contraction, represented by $\mu d.[d]\ldots$ is required at the end.

The abstract machine can be generalized to accommodate $\vee$-elimination with stack environment entries of the form $(d, (C, S))$ where C is an extra closure, which is placed in front of the closure for t in the rule for $[d]t$.

Our $\omega^{\ell/r}$-binders are similar to the ones of [9]. While their $\vee$-introduction rule is also multiplicative, their $\vee$-elimination is entirely different from ours.

IV. Extending NPC for Exception Handling

We choose exception handling as the main example of the enriched computations that PCL can formulate because it is a subject that has been studied in the literature on $\lambda\mu$ calculus and related systems. The use of multiple-conclusion proofs to represent exception handling was formulated in [10], and further studied in [3], [11], where $\mu d.[d]s$ is interpreted as *catch,* and $\mu z.[d]w$ where z is free in w is considered *throw* (or *raise*). Our goal here is not to resolve every issue associated with exception handling but to demonstrate the additional expressiveness that PCL can contribute to formulating this kind of computation.

In $\lambda\mu$ calculus, μ has the same variable capturing constraints as λ, which means that it implements statically scoped exception handling. However, exceptions are handled dynamically in practical programming languages. The ! operator represents contraction and binds dynamically. However, it does not always save the *current* continuation. We can designate the special instance of Lock that should represent *catch* as follows

$$\frac{s : \Gamma \vdash A; \Theta; [A^d, \Delta]}{!d.s : \Gamma \vdash A; \Theta, A^d; [\Delta]}\ catch$$

Even when not in the restricted form above, ! can still serve as a kind of exception handler if one is willing to stretch the meaning of "current continuation." The *catch* must take place in the context of an outer μ binder, but it need not be a trivial context. When the context is trivial, i.e., in the form $\mu d.[d]!d.s$, then the *catch* becomes a static one. Without !, μ alone would only implement a shallow form of *catch.*

A dynamically scoped *catch* can also be implemented using a $\mu'' d$ in conjunction with !:

$$\frac{s : \Gamma \vdash A; \Theta, A^d; [A^d, \Delta]}{\mu'' d.[d]!d.s : \Gamma \vdash A; \Theta; [A^d, \Delta]}\ catch$$

When this *catch* is executed on the abstract machine of Figure 3, which interprets μ as μ'' when possible, any previously saved continuation with label d will be overridden.

The representation of *throw* can only be explained alongside other refinements. These refinements can be applied to both the static and the dynamic versions of *catch.*

In designing a type system for a programming language, we can use red types for data values (integers, strings,

$$\langle x, CE, LE\rangle :: S; [GE] \longrightarrow CE(x) :: S; [GE]$$
$$\langle (u\ v), CE, LE\rangle :: S; [GE] \longrightarrow \langle u, CE, LE\rangle :: \langle v, CE, LE\rangle :: S; [GE]$$
$$\langle \lambda \mathbf{x.t}, \mathbf{CE}, \mathbf{LE}\rangle :: \mathbf{C} :: \mathbf{S}; [\mathbf{GE}] \longrightarrow \langle \mathbf{t}, (\mathbf{x}, \mathbf{C}) :: \mathbf{CE}, \mathbf{nil}\rangle :: \mathbf{S}; [\mathbf{GE}]$$
$$\langle \mu d.t, CE, LE\rangle :: S; [GE] \longrightarrow \langle t, CE, (d, S) :: LE\rangle :: S; [GE]$$
$$\langle [d]t, CE, LE\rangle :: S; [GE] \longrightarrow \langle t, CE, LE\rangle :: LE(d); [GE]$$
$$\langle !\mathbf{d.t}, \mathbf{CE}, \mathbf{LE}\rangle :: \mathbf{S}; [\mathbf{GE}] \longrightarrow \langle \mathbf{t}, \mathbf{CE}, \mathbf{LE}\rangle :: \mathbf{S}; [(\mathbf{d}, \mathbf{LE(d)}) :: \mathbf{GE}]$$
$$\langle ?\mathbf{d.t}, \mathbf{CE}, \mathbf{LE}\rangle :: \mathbf{S}; [\mathbf{GE}] \longrightarrow \langle \mathbf{t}, \mathbf{CE}, (\mathbf{d}, \mathbf{GE(d)}) :: \mathbf{LE}\rangle :: \mathbf{S}; [\mathbf{GE}]$$

Figure 3. An Abstract Machine For $\lambda\mu$!

etc), while green types will represent *errors.* In existing formulations of exception handling using classical logic, a *throw* can occur at any point in the program and different types of exceptions are not distinguished. We wish to enforce stronger constraints for exception raising. PCL has the capability to implement this kind of constraint, but in NPC as given, all locked formulas are placed inside the same box. Under this interpretation, any error can trigger any kind of exception. We need to modify NPC as follows.

First, enumerate all green formulas as E_0, E_1, E_2, ..., with $E_0 = \bot$.

Next, sequents are now $\Gamma \vdash \mathcal{F}; \Theta; [\Delta_{i_1}]^{i_1} \dots [\Delta_{i_n}]^{i_n}$, where $i_1 \dots i_n$ are distinct non-negative integers. The boxes are considered unordered (placed inside a set of boxes). The new versions of Lock and Unlock are found in Figure 4. The restricted form of the right premise of $Unlock_i$ suffices for our purposes. The introduction rules are adjusted in the obvious way: the contents of each $[\ ]^i$ that appears in a premise are merged in the conclusion.

The meaning of a formula A inside $[\ ]^i$ is now $A \to E_i$ on the left-hand side. The semantics easily assure us that these rules remain sound (because $(B \to E_i) \vee B$ is valid). They are also complete since we can still choose to lock all formulas in $[\ ]^0$. Sequents are still finite since only a finite number of boxes would be needed in a proof. When interpreted as a constant, $!_i$ (in the form that implements contraction) would be of type $((P \to E_i) \to P) \to P$ (the proof of this formula is an η-expansion of $!_i$ if $!_i$ is interpreted as a constant: this is to be expected.)

We shall refer to this version of NPC as NPC^n.

A procedure that could throw an exception will have a type of the form $B \to (R \vee \sim\sim E_i)$ where E_i is the type of the exception. The double intuitionistic negation is used so that the disjunction is red, and so the implication remain red, i.e., intuitionistic, without which it would be possible for E_i to escape scope just as in a classical logic setting. Using an explicit $\vee$ to represent an exception type may seem contrary to the very idea of exception handling, in which errors need not be handled locally. However, our $\vee$ is *multiplicative,* with an introduction rule that is also *invertible.* The following term can be used to invert any $A \vee \sim\sim E_i$ to leave only A in the stoup, which then can be cut with an $A \to B$ without interference from $\sim\sim E_i$:

$$\lambda x.(\lambda u.u, \lambda v.\mu z.[d]v)x :\vdash (A \vee B) \to A; B^d; .$$

In this context, the sequence of rules that constitutes throwing an exception is as follows (with some details elided):

$$\dfrac{\dfrac{x :\sim E_i^x \vdash \sim E_i; ;}{\dfrac{\sim E_i^x, \Gamma \vdash 0; [A^d, \Delta]^i \dots}{\dfrac{\Gamma \vdash \sim\sim E_i; R^z, \Theta; [A^d, \Delta]^i \dots}{\Gamma \vdash R; \sim\sim E_i^y, \Theta; [A^d, \Delta]^i \dots}\ \to I}}\ \to E_i \qquad \dfrac{\dfrac{s : \Gamma \vdash A, E_i^e, A^d; [\Delta]^i \dots}{\Gamma \vdash E_i; A^d; [\Delta]^i \dots}}{\Gamma \vdash E_i; ; [A^d, \Delta]^i \dots}\ Unlock}$$

Here, x and e are vacuous in s (weakened) and *catch* is similarly extended to $catch_i$. The abstract machine of Figure 3 can be modified by tagging each entry in $[GE]$ with an index (i.e., with dynamic type information). Figure 5 illustrates the execution of an exception throwing procedure, along with the integral $throw_i$ rule. Note that the procedure labeled g cannot throw an E_i-type exception because of the scoping constraint of *intuitionistic* $\to$ introduction.

Another refinement to note: in terms of Java, $E_0 = \bot$ is the superclass *Exception* of all exceptions. Other exceptions can be represented by green atoms and superclasses by conjunctions of green atoms. Thus a *catch IOException* can also catch a *SocketIOException,* etc.

To be fair to classical logic, we should note that it is also possible to give a logical interpretation to the boxes $[\ ]^i$. Reserve a set of of unique atoms $q_1, q_2, \dots$ The meaning of a formula P inside $[\ ]^i$ would be $P \to q_i$ on the left-hand side (but $[\ ]^0$ for $\bot$ cannot be emulated). Then $Lock$ is admissible because $((P \to q_i) \to P) \to P$ is classically provable, and $Unlock$ becomes $\to$ elimination, which is possible when q_i appears on the right-hand side. However, what cannot be changed is that classical implication is equivalent to a disjunction. A sequent with $(A \to B \vee q_1),\ (C \to D \vee q_2)$ is equivalent to $(A \to B \vee q_1 \vee q_2),\ (C \to D \vee q_1),\ q_2$ along with numerous other possible interpretations. Thus, such an effort in classical logic is bound to be fruitless.

This example illustrates the value of retaining *intuitionistic* implication while allowing for classical reasoning. PCL

$$\frac{s : \Gamma \vdash \mathcal{F}; \Theta; [B^x, \Delta]^i \ldots}{!_i x.s : \Gamma \vdash \mathcal{F}; \Theta, B^x; [\Delta]^i \ldots}\ Lock_i \qquad \frac{s : \Gamma \vdash A; \Theta, B^d; [\Delta]^i \ldots \qquad t : \Gamma' \vdash E_i \to A; ;}{?_i x.(t\ s) : \Gamma\Gamma' \vdash A; \Theta; [B^d, \Delta]^i \ldots}\ Unlock_i$$

Figure 4. Lock and Unlock with multiple boxes

$$\frac{s : \Gamma \vdash A; E_i^e, A^d; [\Delta]^i \ldots}{\mu z.[y]\lambda x.(x\ ?d.\mu e.[d]s) : \Gamma \vdash R; \sim\sim E_i^y, \Theta; [A^d, \Delta]^i \ldots}\ throw_i$$

$$\frac{\dfrac{\dfrac{C, \Gamma \vdash A; ; [K^d]^i}{g : \Gamma \vdash C \to A; \sim\sim E_i^x; [K^d]^i} \to I \quad \dfrac{\Gamma \vdash K; E_i^x, K^d; [\ ]^i}{\Gamma \vdash C; \sim\sim E_i^x; [K^d]^i}\ throw_i}{\Gamma \vdash A; \sim\sim E_i^x; [K^d]^i} \to E}{\Gamma \vdash A \vee \sim\sim E_i; ; [K^d]^i} \vee I$$

Figure 5. *throw* E_i and sample usage with saved continuation K

is not a strengthening of intuitionistic logic or weakening of classical logic: it is a unified logic.

V. Sequent Calculus, Cut Elimination, Soundness, and Completeness

In this section, we establish some basic properties so that PCL can be called logic.

We begin with a sequent calculus, *LPC,* with the subformula property and is suitable for proving completeness. Sequents are of the form $\Gamma \vdash \Theta; [\Delta]$. The stoup is folded into the right-side context as there are no proof terms. In Figure 6, e represents $\bot$ or a green atom, a represents an arbitrary atom, and A, B represent arbitrary formulas.

From the subformula property of this sequent calculus, it is plainly obvious that if a formula is purely red, then it can only have an intuitionistic proof: even a failed partial proof would be intuitionistic once useless $Lock$s are discarded.

A formula with green subformulas may still have an intuitionistic proof if only intuitionistic rules are applied. This particular proof system reveals an important property.

Proposition 1: If a formula A is provable with an atom b colored red, then A is also provable with b colored green. Clearly this holds since the presence of a green atom can only lead to more proofs.

The correctness of other proof systems for PCL follow from the correctness of LPC once the following properties are established.

A. Cut Elimination

In a polarized system, it is important to understand the effect of polarization on cut elimination. The restriction to a green atom or $\bot$ in the LPC $Unlock$ rule (and the $\bot L$ rule) can, in fact, be relaxed to allow all green formulas. We used the restriction for two reasons. First, technically speaking, "sequent calculus" rules should only depend on the top-level structure of formulas. The second reason concerns cut elimination. Consider the following scenario involving the relaxed form of $Unlock$, labeled here as $Unlock'$:

$$\frac{\dfrac{\Gamma \vdash E, \Theta, B; [\Delta]}{\Gamma \vdash E, \Theta; [B, \Delta]}\ Unlock' \qquad E, \Gamma \vdash R, \Theta; [\Delta]}{\Gamma \vdash R, \Theta; [B, \Delta]}\ cut$$

Here, E is an arbitrary green formula and R is a red formula. The $Unlock'$ rule is also semantically sound and should not destroy cut-elimination. In a natural deduction setting, the above situation is normalized by simple substitution. However, cut-elimination in sequent calculus requires us to permute the cut upwards, until E in both subproofs are introduced, forming a "key case." But if we permuted the cut up to the premise of $Unlock'$, we would get $\Gamma \vdash R, \Theta, B; [\Delta]$. Since R is red, the $Unlock'$ may not be applicable beneath. By restricting to a green atom or $\bot$, we can permute the cut upwards until the *right* subproof reaches Id at which point the cut is reduced by substitution just as in natural deduction. If the cut formula is $\bot$, we similarly permute the cut up the right subproof until it reaches $\bot L$, at which point the formula on the right-hand side of the conclusion of $\bot L$ is *green,* which means we can now permute the cut above the $Unlock'$.

This is an example of how polarity information is used to control cut elimination, in a manner not unlike what is found in focused proof systems. However, that is not the principal use of polarization in PCL.

The relaxed form of $Unlock$ is used in NPC proofs, so we establish its equivalence with the restricted version.

Lemma 2: In a cut-free proof, every instance of $Unlock'$ can be replaced by $Unlock$.

The proof consists of permutation arguments, depending on the rule above $Unlock'$.

Cut elimination is proved for LPC in two forms:

$$\frac{\Gamma \vdash A, \Theta; [\Delta] \qquad A, \Gamma' \vdash \Theta'; [\Delta']}{\Gamma\Gamma' \vdash \Theta\Theta'; [\Delta\Delta']}\ cut$$

$$\frac{\Gamma \vdash \Theta; [A, \Delta] \qquad A, \Gamma' \vdash B, \Theta'; [\Delta']}{\Gamma\Gamma' \vdash \Theta\Theta'; [B, \Delta\Delta']}\ cut_2$$

$$\frac{\Gamma \vdash \Theta; [B, \Delta]}{\Gamma \vdash \Theta, B; [\Delta]}\ Lock \qquad \frac{\Gamma \vdash e, \Theta, B; [\Delta]}{\Gamma \vdash e, \Theta; [B, \Delta]}\ Unlock \qquad \frac{A, \Gamma \vdash B; [\Delta]}{\Gamma \vdash A \to B, \Theta; [\Delta]} \to R \qquad \frac{\Gamma \vdash A, B, \Theta; [\Delta]}{\Gamma \vdash A \vee B, \Theta; [\Delta]} \vee R$$

$$\frac{\Gamma \vdash A, \Theta; [\Delta] \quad \Gamma \vdash B, \Theta; [\Delta]}{\Gamma \vdash A \wedge B, \Theta; [\Delta]} \wedge R \qquad \frac{\Gamma \vdash A, \Theta; [\Delta] \quad B, \Gamma \vdash \Theta; [\Delta]}{A \to B, \Gamma \vdash \Theta; [\Delta]} \to L \qquad \frac{A, \Gamma \vdash \Theta; [\Delta] \quad B, \Gamma \vdash \Theta; [\Delta]}{A \vee B, \Gamma \vdash \Theta; [\Delta]} \vee L$$

$$\frac{A, B, \Gamma \vdash \Theta; [\Delta]}{A \wedge B, \Gamma \vdash \Theta; [\Delta]} \wedge L \qquad \frac{}{0, \Gamma \vdash \Theta; [\Delta]}\ 0L \qquad \frac{}{\bot, \Gamma \vdash e, \Theta; [\Delta]} \bot L \qquad \frac{}{\Gamma \vdash 1, \Theta; [\Delta]}\ 1R \qquad \frac{}{a, \Gamma \vdash a, \Theta; [\Delta]}\ Id$$

Figure 6. Sequent Calculus LPC; (e is a green atom or $\bot$)

Instances of cut_2 reduce to cut; they represent structural reduction. A crucial case of cut elimination occurs when cut is permuted above $Lock$, which can be a contraction: here cut_2 is applied to the *copy* of the cut formula inside $[\,]$. The inductive measure of the proof is the lexicographical ordering consisting of the size of the cut formula, followed by the number of $Lock$ rules above the cut, then the heights of subproofs. The rest of the proof is fairly standard.

Theorem 3: Both cut and cut_2 are admissible in LPC.

B. Soundness and Completeness

Soundness of LPC is proved by induction on proofs. Most important are the three non-intuitionistic rules. $Lock$ is sound because $\neg A \vee A$ is always valid, while $Unlock$ and $\bot L$ are sound because $\bot \to E$ is valid for green E. The other rules are all intuitionistically valid.

Completeness is proved by following the traditional strategy of Kripke, Smullyan and Fitting (especially the presentation of Fitting [12].) A sequent is considered *consistent* if it is not provable. Given an unprovable formula A, we can show, using cut elimination, that $A \vee \bot$ is also unprovable. We then construct a countermodel as a saturated tableau of consistent sequents starting with $\vdash A, \bot; [\,]$. All sequents will have the characteristics of Hintikka sets. However, the *root* sequent will also be *maximally consistent* with respect to $\bot$ ($\Gamma \vdash \bot; [\,]$ is not provable): enumerate all subformulas B of A and their negations $\neg B$ and insert each into Γ ($[\,]$ is absorbed into Γ because of its semantic meaning) if it keeps the sequent consistent with respect to both A and $\bot$. Since $B \vee \neg B$ is provable, by the admissibility of cut it follows that *exactly one* of B or $\neg B$ will be inserted into Γ in any such saturation. Thus whenever another formula is properly added to Γ by $\to R$, forming a new possible world, *it will always become $\bot$-inconsistent.* All such worlds force $\bot$, but not Γ, which represents the root.

Theorem 4: A formula is valid in PCL if and only if it is provable in LPC.

Another basic property we can establish from cut elimination is that (propositional) PCL is *decidable,* following traditional arguments.

VI. Core Fragments of PCL

It is easy to identify the purely intuitionistic and purely classical fragments of PCL.

Intuitionistic Logic: Color all atoms red, and do not use $\bot$. All formulas are red.

Classical Logic: Color all atoms green, and use $\bot$ instead of 0. In this extreme, $\neg$ becomes involutive.

LC-style classical logic: Color all atoms red, but use $\bot$ instead of 0. Furthermore, restrict all uses of $\to$ to $\neg a$ where a is atomic. All formulas are in *negation normal form.* Define a syntactic-level operation (i.e, not a new connective) $A^\bot$ that represents the De Morgan dual of A: $1^\bot = \bot$, $\bot^\bot = 1$, $a^\bot = \neg a$, $(\neg a)^\bot = a$, $(A \vee B)^\bot = A^\bot \wedge B^\bot$, and $(A \wedge B)^\bot = A^\bot \vee B^\bot$. Then $A^{\bot\bot}$ is syntactically identical to A. Red formulas are "positive" and green ones are "negative." Notice that A and $A^\bot$ are always of different polarities. In fact, it is valid to consider 1 as either green or red; we designated it red to preserve this LC-type duality. It is possible to construct a fully focused proof system that takes advantage of the perfect dualities of this fragment, but that has already been done [13].

Intuitionistic Control Logic: The purpose of PCL was far more than to support intuitionistic and classical logics as independent fragments. If we colored all atoms red, but do not use 0, we get an intermediate logic with the excluded middle $A \vee \neg A$ but without involutive negation $\neg\neg A \to A$. If we further allowed the use of both $\bot$ and 0, then we get a more expressive logic where both $call/cc$ and the $\mathcal{C}$ control operators can be obtained. We regard this as an important fragment of PCL, and refer to it as *intuitionistic control logic* (ICL).

In the ICL fragment, *formulas need not be considered polarized.* Rather, the formula $\bot$ is given unique treatment. Key to the validity of the unpolarized interpretation is Lemma 2, which showed that only green atoms and $\bot$ need to be given a special role in proofs, in which case the polarity of non-atomic formulas lose their significance.

We further distill ICL down to its implicational fragment to present a compact natural deduction system *NJC* with terms (Figure 7). NJC sequents are of the form $\Gamma \vdash A; [\Delta]$, so it extends the single conclusioned NJ.

The stoup is always non-empty in NJC. The Esc rule of NJC combines the roles of Esc and $Unlock$ in LPC. The rule Con (for contraction) and the γ binder replaces μ. However, Con *keeps the stoup locked.* The formula inside the stoup must be computed by intuitionistic introductions and eliminations, at least until $\bot$ appears. NJC preserves NJ-

$$\frac{t : A^x, \Gamma \vdash B;\ [\Delta]}{(\lambda x.t) : \Gamma \vdash A \to B;\ [\Delta]}\ {\to}I \qquad \frac{t : \Gamma \vdash A \to B;\ [\Delta] \quad s : \Gamma' \vdash A;\ [\Delta']}{(t\ s) : \Gamma\Gamma' \vdash B;\ [\Delta\Delta']}\ {\to}E$$

$$\frac{s : \Gamma \vdash 0;\ [\Delta]}{\mathit{abort}\ s : \Gamma \vdash A;\ [\Delta]}\ 0E \qquad \frac{}{\mathit{exit} : \Gamma \vdash 1;\ [\Delta]}\ \top I \qquad \frac{}{x : A^x, \Gamma \vdash A;\ [\Delta]}\ Id$$

$$\frac{t : \Gamma \vdash A;\ [\Delta]}{[d]t : \Gamma \vdash \bot;\ [A^d, \Delta]}\ \mathit{Esc} \qquad \frac{u : \Gamma \vdash A;\ [A^d, \Delta]}{\gamma d.u : \Gamma \vdash A;\ [\Delta]}\ Con$$

Figure 7. NJC Proof System for Intuitionistic Control Logic

style intuitionistic provability as much as possible. Weakening is only present as a result of 0-elimination, which is intuitionistically valid.

The normalization rules of "$\lambda\gamma$-calculus" consist of three reduction rules followed by three renaming rules:

$$\begin{array}{rcl}
(\lambda x.s)t & \longrightarrow & s[t/x] \\
(\gamma d.s)t & \longrightarrow & \gamma d.(s\{[d](wt)/[d]w\}t) \\
\mathit{abort}(s)t & \longrightarrow & \mathit{abort}(s) \\
\gamma a.s & \longrightarrow & s, \text{ when } a \text{ is not free in } s \\
\gamma a.\gamma b.s & \longrightarrow & \gamma a.s[a/b] \\
[d]\gamma a.s & \longrightarrow & [d]s[d/a].
\end{array}$$

We have proved that this system is confluent and strongly normalizing following traditional techniques. Of course, as in $\lambda\mu$ calculus, confluence is possible because certain other reduction rules were excluded.

The following NJC proof shows that the $\mathcal{C}$ control operator [14] can be obtained.

$$\dfrac{\dfrac{x :\sim\neg A^x \vdash\sim\neg A;[\ ] \quad \dfrac{\dfrac{\dfrac{}{y :\sim\neg A^x, A^y \vdash A;[\ ]}\ Id}{[d]y :\sim\neg A^x, A^y \vdash \bot;\ [A^d]}\ Esc}{\lambda y.[d]y :\sim\neg A^x \vdash\sim\neg A;\ [A^d]}\ {\to}I}{x\ \lambda y.[d]y :\sim\neg A^x \vdash 0;\ [A^d]}\ {\to}E}{\dfrac{\mathit{abort}\ (x\ \lambda y.[d]y) :\sim\neg A^x \vdash A;\ [A^d]}{\gamma d.\mathit{abort}\ (x\ \lambda y.[d]y) :\sim\neg A^x \vdash A;\ [\ ]}\ Con}\ 0E}{\mathcal{C} = \lambda x.\gamma d.\mathit{abort}\ (x\ \lambda y.[d]y) : \vdash\sim\neg A \to A;\ [\ ]}\ {\to}I$$

Compared to the original $\lambda\mu$ calculus, there is no free variable in the $\mathcal{C}$ operator. A similar proof, of $(\neg P \to P) \to P$, derives the *call/cc* operator as $\lambda x.\gamma d.(x\ \lambda y.[d]y)$. Thus, NJC already captures the essential computational capabilities of $\lambda\mu$-calculus without destroying intuitionistic logic.

There is no known translation of PCL to intuitionistic logic, or to linear logic. We have reasons to consider such translations unlikely. Any double negation translation of PCL must allow $A \vee \neg A$ but deny $\neg\neg A \to A$. Linear logic also includes both classical and intuitionistic logics, but not every intermediate logic. One might be tempted to equate green formulas with linear logic formulas $?A$: but consider Peirce's formula $((P \to Q) \to P) \to P$ translated as $!(!(!P \multimap ?Q) \multimap P) \multimap P$. This formula is *still not provable.* The challenge is to provide a translation that preserves provability while keeping at least the outer implication intuitionistic (equivalent to $!A \multimap B$)[3]. However, it is simple to translate PCL into ICL: for each green PCL atom e, reserve a unique ICL atom e', and translate e as $\neg e'$. For example, $\neg\neg e \to e$ becomes $\neg\neg\neg e' \to \neg e'$, which is minimally valid. Thus, ICL forms a core fragment of PCL.

Nevertheless, PCL is significantly more expressive. It embeds classical logic more clearly than ICL. Having green formulas other than $\bot$ means more expressive type systems. It also means that restricting the context outside of $[\ ]$ to a single formula becomes impractical. The combination of $Lock$ and $Open$ into a single Con rule in NJC hides potential computational capabilities.

VII. Continuing Work: Second Order PCL

One problem that has faced polarized logics has been how to assign polarities to second order formulas, specifically to propositional variables that are bound by $\forall$ and $\exists$. One might consider two versions of each quantifier, which restricts also the polarity of formulas that can instantiate them. Another approach might be to keep bound variables unpolarized. None of these approaches seem satisfactory. Proposition 1, however, shows that provability is preserved if some red atoms were colored green instead. In PCL, one can prove a formula such as $b \to b$ with b considered red, so of course b can be replaced by *any* formula, red or green. On the other hand, one can show that there exists a formula $\neg\neg b \to b$ that is provable only if b is considered green. The following important attributes form the basis of a *second order propositional PCL:*

- In $\forall X.P$, the free occurrences of X in P are red;
- In $\exists X.P$, the free occurrences of X in P are green.
- $\forall X.P$ is green if P is green, otherwise it is red.
- $\exists X.P$ is green if P is green, otherwise, it is red.

The inference rules for $\forall$ and $\exists$, in the context of LPC, are as follows

$$\frac{\Gamma \vdash A[B/X], \Theta; [\Delta]}{\Gamma \vdash \exists X.A, \Theta; [\Delta]}\ \exists R \qquad \frac{A, \Gamma \vdash \Theta; [\Delta]}{\exists Y.A, \Gamma \vdash \Theta; [\Delta]}\ \exists L,\ Y \text{ green}$$

$$\frac{\Gamma \vdash A; [\Delta]}{\Gamma \vdash \forall Y.A, \Theta; [\Delta]}\ \forall R,\ Y \text{ red} \qquad \frac{A[B/X], \Gamma \vdash \Theta; [\Delta]}{\forall X.A, \Gamma \vdash \Theta; [\Delta]}\ \forall L$$

[3]Furthermore, the translation should define valid synthetic connectives (i.e., preserve focus in the introduction rules).

The usual restrictions on variable occurrences apply. Note that just because $\exists$-quantified variables are green does not mean that they cannot be instantiated with red formulas in $\exists R$. If one can find an existential witness that is red, then so much the better. We can still universally quantify over green formulas using $\forall X.(\bot \to X) \to P$: although $\bot \to X$ is not technically green, it implies equivalence with green formulas. No complementary form exists, however, to restrict $\exists$ to quantify over only red formulas. That would mean *requiring something to be anything:* a self-contradiction.

Polarization has been used elsewhere as a reflection of *duality.* The polarities of PCL, however, define *two levels of provability.* Nevertheless, a duality between the second-order $\forall$ and $\exists$ introduction rules is revealed. The natural role of polarization in second order PCL is possible because we did not insist on enforcing dualities when assigning polarities to formulas, because we did not insist on an involutive form of negation for all formulas.

VIII. Conclusion

The computational content of classical logic depends on how classical proofs are structured. PCL can still be considered classical logic in that every PCL proof is also a classical proof and, with the right polarity assignments, every classically provable formula has a PCL proof.

There are multiple approaches to controlling classical proofs. By the manner that polarities are assigned to formulas, PCL appears similar to LC. The similarity hides a much deeper distinction. The polarization of LC corresponds to dualities found in linear logic. This notion of polarization is principally concerned with the structure of proofs, with how it controls cut elimination. Focused proofs further advance this use of polarization, and we have contributed to focusing in some of our own work. Although polarization in PCL also impacts cut elimination as we have shown, it is mainly concerned not with the structure of proofs, but with *provability.* It is characterized by when a formula can be considered valid. Focusing in PCL is an issue that is orthogonal to its polarization.

In contrast to LC and focusing, the PCL approach to controlling proofs use restrictions on inference rules that are required for *soundness,* relative to a semantics of validity. The two approaches are not necessarily incompatible, and we hope to combine them in future work.

Acknowledgments. The authors wish to thank Stéphane Lengrand and the anonymous reviewers for helpful comments.

References

[1] T. Griffin, "The formulae-as-types notion of control," in *17th Annual ACM Symp. on Principles of Programming Languages*, 1990, pp. 47–57.

[2] M. Parigot, "$\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction," in *LPAR: Logic Programming and Automated Reasoning, International Conference*, ser. LNCS, vol. 624. Springer, 1992, pp. 190–201.

[3] T. Crolard, "A confluent lambda-calculus with a catch/throw mechanism," *Journal of Functional Programming*, vol. 9, no. 6, pp. 625–647, 1999.

[4] J.-Y. Girard, "A new constructive logic: classical logic," *Math. Structures in Comp. Science*, vol. 1, pp. 255–296, 1991.

[5] J.-M. Andreoli, "Logic programming with focusing proofs in linear logic," *J. of Logic and Computation*, vol. 2, no. 3, pp. 297–347, 1992.

[6] C. Liang and D. Miller, "Kripke semantics and proof systems for combining intuitionistic logic and classical logic," *Annals of Pure and Applied Logic*, vol. 164, no. 2, pp. 86–111, Feb. 2013.

[7] P. de Groote, "An environment machine for the lambda-mu calculus," *Mathematical Structures in Computer Science*, vol. 8, pp. 637–669, 1998.

[8] T. Streicher and B. Reus, "Classical logic, continuation semantics and abstract machines," *Journal of Functional Programming*, vol. 8, no. 6, pp. 543–572, 1998.

[9] E. Ritter, D. Pym, and L. Wallen, "Proof-terms for classical and intuitionistic resolution," *Journal of Logic and Computation*, vol. 10, no. 2, pp. 173–207, 2000.

[10] H. Nakano, "A constructive formalization of the catch and throw mechanism," in *Symposium on Logic in Computer Science*. IEEE, 1992, pp. 82–89.

[11] H. Herbelin, "An intuitionistic logic that proves Markov's principle," in *Symposium on Logic in Computer Science*. IEEE, 2010, pp. 50–56.

[12] M. C. Fitting, *Intuitionistic Logic Model Theory and Forcing*. North-Holland, 1969.

[13] C. Liang and D. Miller, "Focusing and polarization in linear, intuitionistic, and classical logics," *Theoretical Computer Science*, vol. 410, no. 46, pp. 4747–4768, 2009.

[14] M. Felleisen, D. Friedman, E. Kohlbecker, and B. Duba, "A syntactic theory of sequential control," *Theoretical Computer Science*, vol. 52, no. 3, pp. 205–237, 1987.