Technical Report No. 608

A Proof-Theoretic Foundation of Abortive Continuations (Extended version)

by

Zena M. Ariola
Hugo Herbelin
Amr Sabry

February 2005

Computer Science Department
Indiana University
Bloomington, Indiana 47405-7104

A Proof-Theoretic Foundation of Abortive Continuations (Extended version)*

Zena M. Ariola† Hugo Herbelin Amr Sabry‡
University of Oregon INRIA-Futurs Indiana University

February 25, 2005

Abstract

We give an analysis of various classical axioms and characterize a notion of minimal classical logic that enforces Peirce's law without enforcing Ex Falso Quodlibet. We show that a "natural" implementation of this logic is Parigot's classical natural deduction. We then move on to the computational side and emphasize that Parigot's $\lambda\mu$ corresponds to minimal classical logic. A continuation constant must be added to $\lambda\mu$ to get full classical logic. We then map the extended $\lambda\mu$ to a new theory of control, $\lambda_{\mathcal{C}^-\mathsf{tp}}$, which extends Felleisen's reduction theory. The new theory $\lambda_{\mathcal{C}^-\mathsf{tp}}$ distinguishes between aborting and throwing to a continuation and is in correspondence with a refined version of Prawitz's natural deduction system.

1 Introduction

Traditionally, classical logic is defined by extending intuitionistic logic with either Peirce's law, the excluded middle law, or the double negation law. We show that these laws are not equivalent and define *minimal classical logic*, which validates Peirce's law but not Ex Falso Quodlibet (EFQ), *i.e.* the law $\bot \to A$. This logic is interesting from a computational point of view since it corresponds to a calculus with a notion of control (such as *callcc* or μ) which however does not permit aborting of computations.

*Extended version of the conference article "Minimal Classical Logic and Control Operators" [1]

†Supported by National Science Foundation grant number CCR-0204389

‡Supported by National Science Foundation grant number CCR-0204389, by a Visiting Researcher position at Microsoft Research, Cambridge, U.K., and by a Visiting Professor position at the University of Genova, Italy.

Indeed our analysis reveals that closed typed terms of Parigot's $\lambda\mu$ [19, 20] correspond to tautologies of minimal classical logic and not of (full) classical logic. To prove tautologies of classical natural deduction, the $\lambda\mu$ calculus must be extended with a continuation `tp` which denotes the top-level.

On the programming calculi side, the presence of the continuation `tp` makes it possible to distinguish between aborting a computation and throwing to a continuation (as aborting corresponds to throwing to the special top-level continuation). This distinction can be used to develop more refined programming calculi for languages with control operators. In particular, in the seminal theory of control λ_c [10], there is a mismatch between the operational and proof-theoretical interpretation of the reduction theory. This mismatch is resolved by moving to a richer theory with a continuation constant $\lambda_{\mathcal{C}^-\mathtt{tp}}$.

We start in Section 2 with reviewing the definitions of minimal, intuitionistic and classical logic. We present both the axiomatic and structural rendering of these logics. We introduce an equivalent formulation of Prawitz's natural deduction which better corresponds to the axiomatic presentation. We conclude the section with the more recent formulation of Parigot's classical natural deduction which employs sequents with multiple conclusions. In Section 3, we introduce the new notion of *minimal classical logic*. We first analyze which axioms lead to this new logic. Next, we present the restrictions on Prawitz's and Parigot's logics. Section 4 reviews the basic control operators and calculi and their (Curry-Howard) isomorphism to classical logic discovered by Griffin. Sections 5 and 6 introduce our new calculus of control $\lambda_{\mathcal{C}^-\mathtt{tp}}$, establish its main properties, and use it to explain the computational counterparts of Prawitz's and Parigot's logics. Proposition 5.8 in Section 5 replaces Proposition 11 from the conference version [1] which claimed a stronger (but incorrect) result. We discuss related work in Section 7 and conclude in Section 8. Throughout the paper we restrict our attention to propositional logic.

2 Minimal, Intuitionistic and Classical Logic

We successively recall the definitions of minimal, intuitionistic and classical logic, and state simple facts about them. We use natural deduction to formalize the various logics.

2.1 Preliminaries

We assume a set of *formulas*, denoted by Roman uppercase letters A, B, *etc.*, which are built from an infinite set of *propositional atoms* (ranged over by X, Y, *etc.*), a distinguished formula $\bot$ denoting *false*, and *implication* written $\to$. A *named formula* is a pair of a formula and a name taken from an infinite set of *names*. We write A^x, B^α, *etc.* for named formulas. A *context* is a set of named formulas. We use Greek uppercase letters Γ, Δ, *etc.* for contexts.

$$
\begin{array}{rcl}
A, B & ::= & X \mid A \to B \\
\Gamma & ::= & \cdot \mid \Gamma, A
\end{array}
$$

$$
\frac{}{\Gamma, A \vdash_M A}\,Ax \qquad \frac{\Gamma, A \vdash_M B}{\Gamma \vdash_M A \to B}\,\to_i
$$

$$
\frac{\Gamma \vdash_M A \to B \quad \Gamma \vdash_M A}{\Gamma \vdash_M B}\,\to_e
$$

Figure 1: Minimal logic

If we are only interested in provability, contexts could have been defined just as sets of formulas (not as sets of named formulas). But in order to be able to assign λ-terms to proofs, we need to distinguish between different occurrences of the same formula. This is the role of names. Otherwise, the two distinct normal proofs of $A, A \vdash A$ (representable by the λ-terms $\lambda x.\lambda y.x$ and $\lambda x.\lambda y.y$) are identified.

We first consider *sequents* of the form $\Gamma \vdash A$, where the formulas in Γ are the *hypotheses* and the formula on the right-hand side of the symbol $\vdash$ is the *conclusion*. If S is a schematic axiom or rule, we denote by $S, \Gamma \vdash A$ the fact that $\Gamma \vdash A$ is derivable using an arbitrary number of instances of S. In the following we sometimes omit irrelevant hypothesis to make proofs more readable.

2.2 Minimal Logic

Minimal natural deduction is an implementation of minimal logic [15]. It is defined by the set of (schematic) inference rules given in Figure 1 (the names of formulas are left implicit and omitted). It contains a distinguished atomic formula $\bot$ to which no particular rule applies. This is the main difference with intuitionistic logic for which proving $\bot$ entails the inconsistency of the logic.

Minimal logic originally included negation but no formula $\bot$. Negation in minimal logic is quite formal in the sense that $A \to \neg A \to B$ does not hold in general. It can actually be shown that $\neg A$ in the original minimal logic behaves exactly the same as $A \to A_0$ where A_0 is some globally distinguished formula. We here transfer to minimal logic the standard decomposition of $\neg A$ as $A \to \bot$ from intuitionistic or classical logic, hence taking $A_0 \triangleq \bot$, but throwing away any specific rules about $\bot$. Thus, the formula $\bot$, despite having a name which suggests it denotes the absurd formula, could essentially be interpreted by any formula, even a true one.

This is in turn shows that minimal logic can alternatively be seen as the positive fragment (i.e. the fragment without negation) of intuitionistic logic. Since negation $\neg A$ is definable as $A \to \bot$, removing negation from intuitionistic logic can only be understood as removing the inference rules dealing with $\bot$. This is what minimal logic does.

Valuations and *normal proofs* are important tools for reasoning about provability in propositional logic. Reasoning with valuations first requires proving completeness results for classical and intuitionistic logic, while reasoning with normal proofs first requires proving normalization results. We adopt the second solution.

We say that an occurrence of $\rightarrow_e$ (also called Modus Ponens) is *normal* if its left premise is an axiom or another normal instance of Modus Ponens. We say that a proof in minimal logic is *normal* if any occurrence of Modus Ponens in the proof is normal. It is known that a provable statement can be proved with a normal proof.

Lemma 2.1 *From a normal proof of $\Gamma \vdash_M A$ and a normal proof π of $\Gamma, A \vdash_M B$ we can build a normal proof π' of $\Gamma \vdash_M B$ which satisfies the following property: if the last rule of π' is $\rightarrow_i$ and the last rule of π is not, then B is structurally smaller than or equal to A.*

Proof. By induction on the structure of A, then on the structure of proofs. If π is an axiom, take for π' the proof of $\Gamma \vdash_M A$: the property is satisfied. If π starts with a $\rightarrow_i$ rule, then reason by induction on the proof of the premise of the $\rightarrow_i$ rule: the side property is trivially satisfied since its condition is false. Finally, if π starts with the rule $\rightarrow_e$ with premises $\Gamma, A \vdash_M C \rightarrow B$ and $\Gamma, A \vdash_M C$, then we get proofs of $\Gamma \vdash_M C \rightarrow B$ and $\Gamma \vdash_M C$ by the induction hypothesis. Since π is normal, the proof of $\Gamma, A \vdash_M C \rightarrow B$ does not start with rule $\rightarrow_i$. If the proof of $\Gamma \vdash_M C \rightarrow B$ starts with rule $\rightarrow_i$ and premise $\Gamma, C \vdash_M B$, then the side property tells that C is structurally smaller than A. Hence we get a normal proof of $\Gamma \vdash_M B$ by the induction hypothesis on A.

Theorem 2.2 (Prawitz) *If $\Gamma \vdash_M A$ is provable then there is a normal proof of $\Gamma \vdash_M A$.*

Proof. We reason by induction on the structure of the proof. Assume that the proof starts with a non normal occurrence of Modus Ponens:

$$\dfrac{\dfrac{\Gamma, A \vdash_M B}{\Gamma \vdash_M A \rightarrow B}\rightarrow_i \qquad \Gamma \vdash_M A}{\Gamma \vdash_M B}\rightarrow_e$$

By induction hypothesis, there are normal proofs of $\Gamma, A \vdash_M B$ and $\Gamma \vdash_M A$, hence, by the previous lemma, there is normal proof of $\Gamma \vdash_M B$.

For all the systems we present the following weakening lemma will hold.

Lemma 2.3 *Let $\Gamma \vdash A$ a derivable sequent. Then for every Γ' such that $\Gamma \subseteq \Gamma'$, $\Gamma' \vdash A$ is derivable.*

Proof. By structural induction on the proof of $\Gamma \vdash A$.

$$
\begin{array}{rcl}
A, B & ::= & X \mid \bot \mid A \to B \\
\Gamma & ::= & \cdot \mid \Gamma, A
\end{array}
$$

$$
\frac{}{\Gamma, A \vdash_{I} A}\ Ax \qquad \frac{\Gamma, A \vdash_{I} B}{\Gamma \vdash_{I} A \to B} \to_i
$$

$$
\frac{\Gamma \vdash_{I} A \to B \quad \Gamma \vdash_{I} A}{\Gamma \vdash_{I} B} \to_e \qquad \frac{\Gamma \vdash_{I} \bot}{\Gamma \vdash_{I} A} \bot_e
$$

Figure 2: Intuitionistic Logic

2.3 Intuitionistic Logic

Any formula is implied by $\bot$ in intuitionistic logic. A way to express this in natural deduction is to consider $\bot$ as a connective with no introduction rule and a single elimination rule as shown in Figure 2. Obviously, this presentation of intuitionistic logic is equivalent to minimal logic extended with the following schematic axiom:

$$\bot \to A \qquad \text{Ex Falso Quodlibet sequitur (EFQ)}$$

Proposition 2.4 $\Gamma \vdash_{I} A$ *iff* $EFQ, \Gamma \vdash_{M} A$.

In propositional or first-order predicate logic, there is no formula $\bot$ with the desired property, as stated by the following lemma which expresses that (propositional) intuitionistic logic is strictly stronger than minimal logic.

Proposition 2.5 $\nvdash_{M}$ *EFQ.*

Proof. By analysis of the possible forms of a normal proof of the judgment $\vdash_{M} \bot \to A$. Since the context is empty, it can only starts with $\to_i$. Now, take for A a propositional variable, then a normal proof of $\bot \vdash A$ starts with a (possibly empty) sequence of $\to_e$ then an axiom rule $\bot \vdash_{M} \bot$, which implies that $\bot$ has the form $A_1 \to \ldots \to A_n \to A$ which is not independent of A.

In contrast, in minimal second-order logic, a formula having the property of $\bot$ is $\forall X.X$.

Remark 2.6 Negation *can be defined as follows:*

$$\neg A \triangleq A \to \bot \qquad \text{(Abbrev. 1)}$$

or directly with the following inference rules with no reference to $\bot$*:*

$$
\frac{\Gamma, A \vdash B \quad \Gamma, A \vdash \neg B}{\Gamma \vdash \neg A} \neg_i \qquad \frac{\Gamma \vdash A \quad \Gamma \vdash \neg A}{\Gamma \vdash B} \neg_e
$$

The rule $\neg_i$ is derivable in minimal logic whereas $\neg_e$ uses $\bot_e$. In the following we also use the abbreviation:

$$\neg_B A \;\triangleq\; A \to B \tag{Abbrev. 2}$$

Interesting tautologies containing negation can be derived in minimal logic. An example is the contrapositive axiom $(A \to B) \to (\neg B \to \neg A)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \to B, A, \neg B \vdash A \to B}\,Ax \quad \dfrac{}{A \to B, A, \neg B \vdash A}\,Ax}{A \to B, A, \neg B \vdash B}\,\to_e \quad \dfrac{}{A \to B, A, \neg B \vdash \neg B}\,Ax}{A \to B, \neg B \vdash \neg A}\,\neg_i}{A \to B \vdash \neg B \to \neg A}\,\to_i}{\vdash (A \to B) \to (\neg B \to \neg A)}\,\to_i$$

Assuming the set of formulas is enriched with disjunction and the following inference rules:

$$\dfrac{\Gamma \vdash A_1}{\Gamma \vdash A_1 \vee A_2}\,\vee_i^1 \qquad \dfrac{\Gamma \vdash A_2}{\Gamma \vdash A_1 \vee A_2}\,\vee_i^2$$

$$\dfrac{\Gamma \vdash A_1 \vee A_2 \qquad \Gamma, A_1 \vdash C \qquad \Gamma, A_2 \vdash C}{\Gamma \vdash C}\,\vee_e$$

the formula $(\neg A \vee B) \to (A \to B)$ is provable in intuitionistic logic but not in minimal logic. The converse is only provable in weak classical logic. Normalization still holds with the addition of disjunction.

2.4 Axiomatic Presentation of Classical Logic

Traditionally *classical logic* is obtained by adding any of the schematic axioms given in Figure 3 to intuitionistic logic [17]. To acquire a better understanding of the strength of these classical axioms we analyze them in minimal logic, and further classify them in three categories: we call $\mathrm{PL}_\bot$ and EM *weak classical axioms*, PL and GEM *minimal classical axioms*, and DN a *(full) classical axiom.* We remark that none of the classical axioms are derivable in minimal logic and that the weak classical axioms are weaker than the minimal classical axioms which themselves are weaker than DN. Together with EFQ, weak and minimal classical axioms are however equivalent to DN.

Proposition 2.7 *In minimal logic, we have:*

1. *None of $\mathrm{PL}_\bot$, PL, EM, GEM, nor DN is derivable.*
2. *$\mathrm{PL}_\bot$ and EM are equivalent (as schemes).*
3. *GEM and PL are equivalent (as schemes).*

Weak classical:	
$(\neg A \to A) \to A$	Weak Peirce's law ($\mathrm{PL}_\bot$)
$\neg A \vee A$	Excluded middle (EM)
Minimal Classical :	
$((A \to B) \to A) \to A$	Peirce's law (PL)
$(A \to B) \vee A$	Generalized excluded-middle (GEM)
Classical :	
$\neg\neg A \to A$	Double negation law (DN)

Figure 3: Classical Schematic Axioms

4. *GEM and PL imply EM and $\mathrm{PL}_\bot$ but not conversely.*

5. *DN implies GEM and PL but not conversely.*

6. *DN, EM+EFQ, GEM+EFQ, $\mathrm{PL}_\bot$+EFQ, and PL+EFQ are all equivalent.*

Proof. In the following proofs we will sometimes omit the irrelevant assumptions.

1. By analysis of the possible forms of a normal proof of the axioms.

2. From EM to $\mathrm{PL}_\bot$:

$$\dfrac{\dfrac{\mathrm{EM} \qquad \dfrac{}{\neg A \to A, A \vdash A}\,Ax \qquad \dfrac{\dfrac{}{\neg A, \neg A \vdash \neg A}\,Ax \qquad \dfrac{}{\neg A, \neg A \to A \vdash \neg A \to A}\,Ax}{\neg A \to A, \neg A \vdash A}\,\to_e}{\neg A \to A \vdash A}\,\vee_e}{\vdash (\neg A \to A) \to A}\,\to i$$

From $\mathrm{PL}_\bot$ to EM:

$$\dfrac{\mathrm{PL}_\bot \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg(A \vee \neg A) \vdash \neg(A \vee \neg A)}\,Ax \qquad \dfrac{\dfrac{}{A \vdash A}\,Ax}{A \vdash A \vee \neg A}\,\vee_i}{\neg(A \vee \neg A), A \vdash \bot}\,\to_e}{\neg(A \vee \neg A) \vdash \neg A}\,\to_i}{\neg(A \vee \neg A) \vdash A \vee \neg A}\,\vee_i}{\vdash \neg(A \vee \neg A) \to (A \vee \neg A)}\,\to_i}{\vdash A \vee \neg A}\,\to_e$$

The instance of $\mathrm{PL}_\bot$ used in the above proof is:

$$(\neg(A \vee \neg A) \to (A \vee \neg A)) \to A \vee \neg A$$

3. The proofs are similar to the given before.
   From GEM to PL:

$$\dfrac{\dfrac{GEM \quad \dfrac{}{A \vdash A}\,Ax \quad \dfrac{\dfrac{}{\neg_B A \vdash \neg_B A}\,Ax \quad \dfrac{}{\neg_B A \to A \vdash \neg_B A \to A}\,Ax}{\neg_B A \to A, \neg_B A \vdash A}\,\to_e}{\neg_B A \to A \vdash A}\,\vee_e}{\vdash (\neg_B A \to A) \to A}\,\to_i$$

   From PL to GEM:

$$\dfrac{\mathrm{PL} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg_B(A \vee \neg_B A) \vdash \neg_B(A \vee \neg_B A)}\,Ax \quad \dfrac{\dfrac{}{A \vdash A}\,Ax}{A \vdash A \vee \neg_B A}\,\vee_i}{\neg_B(A \vee \neg_B A), A \vdash B}\,\to_e}{\neg_B(A \vee \neg_B A) \vdash \neg_B A}\,\to_i}{\neg_B(A \vee \neg_B A) \vdash A \vee \neg_B A}\,\vee_i}{\vdash \neg_B(A \vee \neg_B A) \to (A \vee \neg_B A)}\,\to_i}{\vdash A \vee \neg_B A}\,\to_e$$

   The instance of PL used above is:

$$(\neg_B(A \vee \neg_B A) \to (A \vee \neg_B A)) \to A \vee \neg_B A$$

4. By analysis of possible forms of normal proofs of GEM and PL.

5. DN implies PL:

$$\dfrac{\dfrac{\mathrm{DN} \quad \dfrac{\dfrac{\dfrac{}{\neg A \vdash \neg A}\,Ax \quad \dfrac{\dfrac{}{\neg A \vdash \neg A}\,Ax \quad \dfrac{}{\neg A \to A \vdash \neg A \to A}\,Ax}{\neg A, \neg A \to A \vdash A}\,\to_e}{\neg A, \neg A \to A \vdash \bot}\,\to_e}{\neg A \to A \vdash \neg\neg A}\,\to_i}{\neg A \to A \vdash A}\,\to_e}{\vdash (\neg A \to A) \to A}\,\to_i$$

   PL does not imply DN by analysis of possible forms of normal proofs of PL.

6. $\mathrm{PL}_\bot$ and EFQ imply DN:

$$\dfrac{\dfrac{\mathrm{PL}_\bot \quad \dfrac{\dfrac{\mathrm{EFQ} \quad \dfrac{\dfrac{}{\neg\neg A \vdash \neg\neg A}\,Ax \quad \dfrac{}{\neg A \vdash \neg A}\,Ax}{\neg\neg A, \neg A \vdash \bot}\,\to_e}{\neg\neg A, \neg A \vdash A}\,\to_e}{\neg\neg A \vdash \neg A \to A}\,\to_i}{\neg\neg A \vdash A}\,\to_e}{\vdash \neg\neg A \to A}\,\to_i$$

$$
\begin{array}{rcl}
A, B & ::= & X \mid \bot \mid A \to B \\
\Gamma & ::= & \cdot \mid \Gamma, A \mid \Gamma, A \to \perp\!\!\!\perp
\end{array}
$$

$$
\frac{}{\Gamma, A \vdash_{RAA} A}\ Ax
$$

$$
\frac{\Gamma, A \vdash_{RAA} B}{\Gamma \vdash_{RAA} A \to B}\ \to_i
\qquad
\frac{\Gamma \vdash_{RAA} A \to B \quad \Gamma \vdash_{RAA} A}{\Gamma \vdash_{RAA} B}\ \to_e
$$

$$
\frac{\Gamma, A \to \perp\!\!\!\perp \vdash_{RAA} A}{\Gamma, A \to \perp\!\!\!\perp \vdash_{RAA} \perp\!\!\!\perp}\ \perp\!\!\!\perp_i
\qquad
\frac{\Gamma, A \to \perp\!\!\!\perp \vdash_{RAA} \perp\!\!\!\perp}{\Gamma \vdash_{RAA} A}\ RAA_{\perp\!\!\!\perp}
$$

$$
\frac{\Gamma \vdash_{RAA} \bot}{\Gamma \vdash_{RAA} \perp\!\!\!\perp}\ \bot_e
$$

Figure 4: Prawitz's Classical Logic

DN implies EFQ:

$$
\dfrac{\dfrac{DN \qquad \dfrac{\dfrac{}{\neg A, \bot \vdash \bot}\ Ax}{\bot \vdash \neg\neg A}\ \to_i}{\bot \vdash A}\ \to_e}{\vdash \bot \to A}\ \to_i
$$

2.5 Structural Presentation of Classical Logic

We present two implementations of classical logic: we start with an alternative of Prawitz's classical logic [23] and then present Parigot's Classical Natural Deduction [19].

2.5.1 Prawitz's Classical Logic

Prawitz [23] defines classical logic as minimal logic plus the Reductio Ad Absurdum rule (RAA):

$$
\frac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}
$$

This rule implies EFQ as DN implies EFQ. The $\mathrm{PL}_\bot$ axiom is not equivalent to the RAA rule. However, $\mathrm{PL}_\bot$ plus EFQ are equivalent to RAA. In here we develop a revision of Prawitz's classical logic which comes from analyzing the following proof of PL, where Γ, Γ_1 and Γ_2 stand for $\{A, \neg A, \neg_B A \to A\}$, $\{\neg A, \neg_B A \to A\}$, and $\{\neg_B A \to A\}$ respectively:

$$
\begin{prooftree}
\AxiomC{$\mathrm{PL}_\bot$}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$\Gamma_1 \vdash \neg_B A \to A$}
\AxiomC{EFQ}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$\Gamma \vdash A$}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$\Gamma \vdash \neg A$}
\RightLabel{$\to_e$}
\BinaryInfC{$\Gamma \vdash \bot$}
\RightLabel{$\to_e$}
\BinaryInfC{$\Gamma \vdash B$}
\RightLabel{$\to_i$}
\UnaryInfC{$\Gamma_1 \vdash A \to B$}
\RightLabel{$\to_e$}
\BinaryInfC{$\Gamma_1 \vdash A$}
\RightLabel{$\to_i$}
\UnaryInfC{$\Gamma_2 \vdash \neg A \to A$}
\RightLabel{$\to_e$}
\BinaryInfC{$\Gamma_2 \vdash A$}
\RightLabel{$\to_i$}
\UnaryInfC{$\vdash (\neg_B A \to A) \to A$}
\end{prooftree}
$$

The proof uses EFQ. However, this seems counter-intuitive since PL is weaker than DN, and therefore the proof should not need the full power of EFQ. We address this discrepancy by providing a restricted version of EFQ that can be used to prove PL from $\mathrm{PL}_\bot$ but not DN. This is done by defining a new $\bot_e$ rule as follows:

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash \perp\!\!\!\perp} \bot_e$$

and by introducing the *Weakening* rule defined as follows:

$$\frac{\Gamma \vdash \perp\!\!\!\perp}{\Gamma \vdash A} \; Weakening$$

The new symbol $\perp\!\!\!\perp$ stands for a sequent with no conclusions, that is, instead of writing $\Gamma \vdash$ we write $\Gamma \vdash \perp\!\!\!\perp$. As we explain in Section 5, the symbol $\perp\!\!\!\perp$ has an interesting computational justification. In Figure 4 we present Prawitz's revision. Formulae do not contain $\perp\!\!\!\perp$, thus for example the sequent:

$$\Gamma \vdash A \to \perp\!\!\!\perp$$

is not a syntactically correct judgment. However, $\perp\!\!\!\perp$ can occur in the context Γ in the form of $A \to \perp\!\!\!\perp$. Other than the $\bot_e$ rule, $\perp\!\!\!\perp$ is introduced by the $\perp\!\!\!\perp_i$ rule. This rule is not subsumed by the implication elimination:

$$\frac{\Gamma, A \to \perp\!\!\!\perp \vdash A \to \perp\!\!\!\perp \quad \Gamma, A \to \perp\!\!\!\perp \vdash A}{\Gamma, A \to \perp\!\!\!\perp \vdash \perp\!\!\!\perp} \to_e$$

since $A \to \perp\!\!\!\perp$ is not a formula and hence the left-hand side sequent in the premise is not a correct instance of the axiom.

The addition of $\bot_e$ and *Weakening* to minimal logic gives intuitionistic logic. To obtain classical logic, a variant of RAA, called $RAA_{\perp\!\!\!\perp}$, must be added. The addition eliminates the need for *Weakening* which becomes an admissible rule as stated in the next remark. We may later on use the admissible *Weakening* rule to simplify the presentation.

Remark 2.8 *The following rule is admissible in Prawitz's Classical Logic*

$$\frac{\Gamma \vdash \Perp}{\Gamma \vdash A}\ \mathit{Weakening} \quad .$$

Indeed, $\Gamma, A \to \Perp \vdash \Perp$ *is derivable by Lemma 2.3 and by* $RAA_{\Perp}$ *one obtains* $\Gamma \vdash A$.

Example 2.9 *We formulate the proofs of PL and DN in the logic of Figure 4. Only the proof of DN uses the* $\bot_e$ *rule. The use of EFQ in the proof of PL is replaced by an application of the Weakening rule.*

1.

$$\begin{prooftree}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$\neg_B A \to A \vdash \neg_B A \to A$}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$\neg_{\Perp} A, A \vdash A$}
\RightLabel{$\Perp_i$}
\UnaryInfC{$\neg_{\Perp} A, A \vdash \Perp$}
\RightLabel{$Weakening$}
\UnaryInfC{$\neg_{\Perp} A, A \vdash B$}
\RightLabel{$\to_i$}
\UnaryInfC{$\neg_{\Perp} A \vdash \neg_B A$}
\RightLabel{$\to_e$}
\BinaryInfC{$\neg_{\Perp} A, \neg_B A \to A \vdash A$}
\RightLabel{$\Perp_i$}
\UnaryInfC{$\neg_B A \to A, \neg_{\Perp} A \vdash \Perp$}
\RightLabel{$RAA_{\Perp}$}
\UnaryInfC{$\neg_B A \to A \vdash A$}
\RightLabel{$\to_i$}
\UnaryInfC{$\vdash (\neg_B A \to A) \to A$}
\end{prooftree}$$

2.

$$\begin{prooftree}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$\neg\neg A \vdash \neg\neg A$}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$A \to \Perp, A \vdash A$}
\RightLabel{$\Perp_i$}
\UnaryInfC{$A \to \Perp, A \vdash \Perp$}
\RightLabel{$Weakening$}
\UnaryInfC{$A \to \Perp, A \vdash \bot$}
\RightLabel{$\to_i$}
\UnaryInfC{$A \to \Perp \vdash \neg A$}
\RightLabel{$\to_e$}
\BinaryInfC{$A \to \Perp, \neg\neg A \vdash \bot$}
\RightLabel{$\bot_e$}
\UnaryInfC{$A \to \Perp, \neg\neg A \vdash \Perp$}
\RightLabel{$RAA_{\Perp}$}
\UnaryInfC{$\neg\neg A \vdash A$}
\RightLabel{$\to_i$}
\UnaryInfC{$\vdash \neg\neg A \to A$}
\end{prooftree}$$

Proposition 2.10 $\Gamma \vdash_{RAA} A$ *iff* $\Gamma', PL \vdash_I A$, *where* Γ' *is obtained from* Γ *by replacing each* $B \to \Perp$ *by a subcontext* Δ *of the form*

$$B \to C_1, \cdots B \to C_n \quad .$$

Proof.

$\Leftarrow$ follows from the fact that PL is provable in Prawitz's logic and from the fact that each axiom of the form

$$\overline{\Gamma', A \to B_1, \cdots A \to B_n, PL \vdash A \to B_i}$$

can be replaced by

$$\begin{prooftree}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$\Gamma, A \to \Perp, A \vdash A$}
\RightLabel{$\Perp_i$}
\UnaryInfC{$\Gamma, A \to \Perp, A \vdash \Perp$}
\RightLabel{$Weakening$}
\UnaryInfC{$\Gamma, A \to \Perp, A \vdash B_i$}
\RightLabel{$\to_i$}
\UnaryInfC{$\Gamma, A \to \Perp \vdash A \to B_i$}
\end{prooftree}$$

$\Rightarrow$ by rule induction. The only interesting case is rule RAA. We distinguish two cases depending on which rule introduced $\perp\!\!\!\perp$.

$\perp_e$) We have:

$$\cfrac{\cfrac{\Gamma A \to \perp\!\!\!\perp \vdash \perp}{\Gamma A \to \perp\!\!\!\perp \vdash \perp\!\!\!\perp}\ \perp_e}{\Gamma \vdash A}\ RAA_{\perp\!\!\!\perp}$$

By the induction hypothesis:

$$\Gamma', A \to B_1, \cdots, A \to B_n, PL \vdash \perp \ .$$

The result then follows from $\perp_e$ and n applications of PL_r:

$$\cfrac{\cfrac{\Gamma', A \to B_1, \cdots, A \to B_n, PL \vdash \perp}{\Gamma', A \to B_1, \cdots A \to B_n, PL \vdash A}\ \perp_e}{\Gamma', PL \vdash A}\ \mathrm{PL}^n$$

where PL_r stands for the derived inference rule

$$\cfrac{\cfrac{}{\Gamma, PL \vdash ((A \to B) \to A) \to A} \qquad \cfrac{\Gamma, PL, A \to B \vdash A}{\Gamma, PL \vdash (A \to B) \to A}}{\Gamma, PL \vdash A}$$

$\perp\!\!\!\perp_i$) We further distinguish two cases. Suppose:

$$\cfrac{\cfrac{\Gamma, A \to \perp\!\!\!\perp \vdash A}{\Gamma, A \to \perp\!\!\!\perp \vdash \perp\!\!\!\perp}\ \perp\!\!\!\perp_i}{\Gamma \vdash A}\ RAA_{\perp\!\!\!\perp}$$

By the induction hypothesis:

$$\Gamma', A \to B_1, \cdots, A \to B_n, PL \vdash A$$

The result then follows by n applications of PL_r.
For the other case, we have:

$$\cfrac{\cfrac{\Gamma, A \to \perp\!\!\!\perp, B \to \perp\!\!\!\perp \vdash B}{\Gamma, A \to \perp\!\!\!\perp, B \to \perp\!\!\!\perp \vdash \perp\!\!\!\perp}\ \perp\!\!\!\perp_i}{\Gamma, B \to \perp\!\!\!\perp \vdash A}\ RAA_{\perp\!\!\!\perp}$$

By the induction hypothesis:

$$\Gamma', \Delta_1, \Delta_2, PL \vdash B$$

where Δ_1 and Δ_2 are as follows:

$$\Delta_1 = \{A \to C_1, \cdots, A \to C_n\}$$
$$\Delta_2 = \{B \to D_1, \cdots, B \to D_m\}$$

$$
\begin{array}{rcl}
A, B & ::= & X \mid A \to B \mid \neg A \\
\Gamma, \Delta & ::= & \cdot \mid \Gamma, A
\end{array}
$$

$$
\frac{}{\Gamma, A \vdash_C A; \Delta}\ Ax
$$

$$
\frac{\Gamma, A \vdash_C B; \Delta}{\Gamma \vdash_C A \to B; \Delta}\ \to_i
\qquad
\frac{\Gamma \vdash_C A \to B; \Delta \qquad \Gamma \vdash_C A; \Delta}{\Gamma \vdash_C B; \Delta}\ \to_e
$$

$$
\frac{\Gamma, A \vdash_C ; \Delta}{\Gamma \vdash_C \neg A; \Delta}\ \neg_i
\qquad
\frac{\Gamma \vdash_C \neg A; \Delta \qquad \Gamma' \vdash_C A; \Delta'}{\Gamma, \Gamma' \vdash_C ; \Delta, \Delta'}\ \neg_e
$$

$$
\frac{\Gamma \vdash_C A; A, \Delta}{\Gamma \vdash_C ; A, \Delta}\ \mathit{Passivate}
\qquad
\frac{\Gamma \vdash_C ; A, \Delta}{\Gamma \vdash_C A; \Delta}\ \mathit{Activate}
$$

Figure 5: Parigot's Classical Natural Deduction for the $\{\neg, \to\}$-fragment

We proceed as follows:

$$
\dfrac{\dfrac{\dfrac{}{B \to A \vdash B \to A}\ Ax \qquad \Gamma', \Delta_1, \Delta_2, PL \vdash B}{\Gamma', \Delta_1, \Delta_2, B \to A, PL \vdash A}\ \to_e}{\Gamma', \Delta_2, B \to A, PL \vdash A}\ \mathrm{PL}_r^n
$$

2.5.2 Parigot's Natural Deduction with Multiple Conclusions

As initially shown by Gentzen [11] in his sequent calculus LK, classical logic can be obtained by considering sequents with several conclusions instead of adding new inference rules. Parigot [19] extended this approach to natural deduction. We show that using sequents with several conclusions allows for a uniform presentation of different logics.

Parigot's convention is to have two kinds of sequents, one with only named formulas on the right:

$$\Gamma \vdash \Delta \ ,$$

and one with exactly one unnamed formula on the right:

$$\Gamma \vdash A, \Delta \ .$$

To clearly mark the distinction between an unnamed formula and the set Δ of named formulas, we write the above sequents as:

$$\Gamma \vdash ; \Delta \text{ and } \Gamma \vdash A; \Delta \ .$$

$$A, B \quad ::= \quad X \mid \bot \mid A \to B$$
$$\Gamma, \Delta \quad ::= \quad \cdot \mid \Gamma, A$$

$$\frac{}{\Gamma, A \vdash_C A; \Delta}\ Ax$$

$$\frac{\Gamma, A \vdash_C B; \Delta}{\Gamma \vdash_C A \to B; \Delta} \to_i \qquad \frac{\Gamma \vdash_C A \to B; \Delta \qquad \Gamma \vdash_C A; \Delta}{\Gamma \vdash_C B; \Delta} \to_e$$

$$\frac{\Gamma \vdash_C A; A, \Delta}{\Gamma \vdash_C; A, \Delta}\ \textit{Passivate} \qquad \frac{\Gamma \vdash_C; A, \Delta}{\Gamma \vdash_C A; \Delta}\ \textit{Activate}$$

$$\frac{\Gamma \vdash_C \bot; \Delta}{\Gamma \vdash_C; \Delta} \bot_e$$

Figure 6: Classical Natural Deduction for the $\{\bot, \to\}$-fragment

Girard [12] calls the optional formula a *stoup*. The formulas in Γ are the *hypotheses* and the formulas on the right-hand side of the symbol $\vdash$ are the *conclusions*. In each case, the intuitive meaning is that the conjunction of the hypotheses implies the disjunction of the conclusions. A sequent with no conclusion means the negation of the conjunction of the hypotheses.

The inference rules are shown in Figure 5. The formulas explicitly mentioned in the inference rules are called *active*. The one bearing the connective (introduced or eliminated) is called the *main* formula. The notation A, Δ stands for the union of the singleton context A and Δ and assumes that both components are disjoint. All rules, except the *Passivate* rule and the $\neg_e$ rule, have an active formula in the conclusion. The axiom and implication rules are as in minimal logic. The goal of the *Activate* and *Passivate* rules is to allow changing focus from the main formula to any other formula in the context. For example, the sequent:

$$\frac{}{A \vdash A; B, A}\ Ax$$

indicates that our focus is on formula A. To focus on formula B, appearing in the right-hand side context, we first need to unfocus A (using the *Passivate* rule):

$$\frac{A \vdash A; B, A}{A \vdash; B, A}\ \textit{Passivate}$$

thus obtaining a sequent with no active formula. To focus on B we use the *Activate* rule:

$$\frac{A \vdash; B, A}{A \vdash B; A,}\ \textit{Activate}$$

These steps are necessary to show $\vdash A \to B; A$, since in order to apply the implication introduction rule the formula B has to be active:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A; B, A}\,\mathit{Ax}}{A \vdash ; B, A}\,\mathit{Passivate}}{A \vdash B; A}\,\mathit{Activate}}{\vdash A \to B; A}\,\to_i$$

Example 2.11 *Proving PL and DN in Parigot's logic.*

1.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(A \to B) \to A \vdash (A \to B) \to A;}\,\mathit{Ax} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A; A, B}\,\mathit{Ax}}{A \vdash ; A, B}\,\mathit{Passivate}}{A \vdash B; A}\,\mathit{Activate}}{\vdash A \to B; A}\,\to_i}{(A \to B) \to A \vdash A; A}\,\to_e}{(A \to B) \to A \vdash ; A}\,\mathit{Passivate}}{(A \to B) \to A \vdash A;}\,\mathit{Activate}}{\vdash ((A \to B) \to A) \to A;}\,\to_i$$

2.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\neg\neg A \vdash \neg\neg A}\,\mathit{Ax} \qquad \dfrac{\dfrac{\dfrac{}{\neg\neg A, A \vdash A; A}\,\mathit{Ax}}{\neg\neg A, A \vdash ; A}\,\mathit{Passivate}}{\neg\neg A \vdash \neg A; A}\,\neg_i}{\neg\neg A \vdash ; A}\,\neg_e}{\neg\neg A \vdash A;}\,\mathit{Activate}}{\vdash \neg\neg A \to A;}\,\to_i$$

Since we are only focusing on implication, it is desirable to give an equivalent formulation of classical natural deduction using $\bot$ and $\to$. The $\neg_i$ corresponds to implication introduction:

$$\dfrac{\dfrac{\Gamma, A \vdash ; \bot, \Delta}{\Gamma, A \vdash \bot; \Delta}\,\mathit{Activate}}{\Gamma \vdash \neg A; \Delta}\,\to_i$$

The above contains a silent weakening of the right-hand side context. With respect to $\neg_e$, we have:

$$\dfrac{\Gamma \vdash \neg A; \Delta \quad \Gamma \vdash A; \Delta}{\Gamma \vdash \bot; \Delta}\,\to_e$$

We still have a focused formula instead of just having named formulas on the right-hand side of the sequent. Applying the *Passivate* rule gives:

$$\Gamma \vdash ; \Delta, \bot$$

instead of:

$$\Gamma \vdash ; \Delta$$

We conclude that the simulation of $\neg_e$ requires a special passivate rule:

$$\frac{\Gamma \vdash \bot ; \Delta}{\Gamma \vdash ; \Delta}$$

The new system is given in Figure 6. DN is now provable as follows:

$$\dfrac{\dfrac{\dfrac{}{\neg\neg A \vdash \neg\neg A}\ Ax \quad \dfrac{\dfrac{\dfrac{}{\neg\neg A, A \vdash A ; \bot}\ Ax}{\neg\neg A, A \vdash \bot ; A}\ Pass/Act}{\neg\neg A \vdash \neg A ; A}\ \to_i}{\dfrac{\neg\neg A \vdash \bot ; A}{\neg\neg A \vdash A ;}\ \bot_e/Act}\ \to_e}{\vdash \neg\neg A \to A ;}\ \to_i$$

The above proof uses the rule $\bot_e$ whereas the proof of PL does not.

Parigot's classical logic is a correct characterization of classical logic, as expressed in the following proposition. If the context Δ is the set of formulas $A_1, \cdots, A_n$, then we write $\neg_{\perp\!\!\!\perp}\Delta$ for the set $\neg_{\perp\!\!\!\perp}A_1, \cdots, \neg_{\perp\!\!\!\perp}A_n$, where $\neg_{\perp\!\!\!\perp}A$ stands for $A \to \perp\!\!\!\perp$.

Proposition 2.12
- $\Gamma \vdash_C A; \Delta$ *iff* $\Gamma, \neg_{\perp\!\!\!\perp}\Delta \vdash_{RAA} A$.
- $\Gamma \vdash_C A$ *iff* $PL, \Gamma \vdash_I A$

3 Minimal Classical Logic

The previous section suggests that there is space for a classical logic which does validate Peirce's law (or GEM) but not EFQ. By analogy with minimal intuitionistic logic, we call this logic *minimal classical logic*. As discussed in the case of minimal intuitionistic logic, minimal classical logic can be seen as the positive fragment of classical logic where the absurdity formula and negation are still there but in a degenerated form that does not allow to prove any formula from an absurdity.

Since $\bot$, in minimal classical logic, is a distinguished formula without any special properties, EM and $\text{PL}_\bot$, without EFQ, are weaker than PL and their addition to minimal logic seems uninteresting.

3.1 Prawitz's Minimal Classical Logic

The minimal subset of Prawitz's classical logic is obtained by disallowing the $\bot_e$ rule and is denoted by $\vdash_{MRAA}$. See Figure 7.

Proposition 3.1 $\Gamma \vdash_{MRAA} A$ *iff* $\Gamma, PL \vdash_M A$.

$$\begin{array}{rcl} A, B & ::= & X \mid \perp \mid A \to B \\ \Gamma & ::= & \cdot \mid \Gamma, A \mid \Gamma, A \to \perp\!\!\!\perp \end{array}$$

$$\frac{}{\Gamma, A \vdash_{MRAA} A}\ Ax$$

$$\frac{\Gamma, A \vdash_{MRAA} B}{\Gamma \vdash_{MRAA} A \to B} \to_i \qquad \frac{\Gamma \vdash_{MRAA} A \to B \quad \Gamma \vdash_{MRAA} A}{\Gamma \vdash_{MRAA} B} \to_e$$

$$\frac{\Gamma, A \to \perp\!\!\!\perp \vdash_{MRAA} A}{\Gamma, A \to \perp\!\!\!\perp \vdash_{MRAA} \perp\!\!\!\perp}\ \perp\!\!\!\perp_i \qquad \frac{\Gamma, A \to \perp\!\!\!\perp \vdash_{MRAA} \perp\!\!\!\perp}{\Gamma \vdash_{MRAA} A}\ RAA_{\perp\!\!\!\perp}$$

Figure 7: Prawitz's Minimal Natural Deduction

$$\begin{array}{rcl} A, B & ::= & X \mid \perp \mid A \to B \\ \Gamma, \Delta & ::= & \cdot \mid \Gamma, A \mid \end{array}$$

$$\frac{}{\Gamma, A \vdash_{MC} A; \Delta}\ Ax$$

$$\frac{\Gamma, A \vdash_{MC} B; \Delta}{\Gamma \vdash_{MC} A \to B; \Delta} \to_i \qquad \frac{\Gamma \vdash_{MC} A \to B; \Delta \qquad \Gamma \vdash_{MC} A; \Delta}{\Gamma \vdash_{MC} B; \Delta} \to_e$$

$$\frac{\Gamma \vdash_{MC} A; A, \Delta}{\Gamma \vdash_{MC}; A, \Delta}\ \mathit{Passivate} \qquad \frac{\Gamma \vdash_{MC}; A, \Delta}{\Gamma \vdash_{MC} A; \Delta}\ \mathit{Activate}$$

Figure 8: Minimal Classical Natural Deduction for the $\{\perp, \to\}$-fragment

3.2 Parigot's Minimal Classical Logic

An axiom-free implementation of minimal classical logic is actually Parigot's classical natural deduction with no special rule for $\perp$. That is, without hiding the occurrences of the $\perp$ formula on the right-hand-side of the sequents [19, p. 198]. We give Parigot's minimal classical natural deduction in Figure 8.

Proposition 3.2
- $\Gamma \vdash_{MC} A; \Delta$ *iff* $\Gamma, \neg_{\perp\!\!\!\perp}\Delta \vdash_{MRAA} A$.
- $\Gamma \vdash_{MC} A$ *iff* $PL, \Gamma \vdash_{M} A$

Using Proposition 2.7(5), we have the following corollary.

Corollary 3.3 *Minimal Parigot's classical natural deduction does not prove DN.*

However the sequent $\vdash_{MC} \neg\neg A \rightarrow A; \bot$ is provable as follows:

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$\neg\neg A \vdash \neg\neg A$}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$\neg\neg A, A \vdash A; \bot$}
\RightLabel{$Pass/Act$}
\UnaryInfC{$\neg\neg A, A \vdash \bot; A$}
\RightLabel{$\rightarrow_i$}
\UnaryInfC{$\neg\neg A \vdash \neg A; A$}
\RightLabel{$Pass/Act$}
\BinaryInfC{$\neg\neg A \vdash \bot; A$}
\RightLabel{$\rightarrow_e$}
\UnaryInfC{$\neg\neg A \vdash A; \bot$}
\RightLabel{$\rightarrow_i$}
\UnaryInfC{$\vdash \neg\neg A \rightarrow A; \bot$}
\end{prooftree}
$$

We now define the notion of *normal proof* for a minimal variant of Parigot's classical natural deduction. We say that an occurrence of the rule *Passivate* is *normal* if its premise is not an *Activate* rule. We say that a proof in minimal classical natural deduction is *normal* if any occurrence of Modus Ponens in the proof is normal (this is the same definition as for minimal non-classical natural deduction) and if any occurrence of *Passivate* is also normal.

Theorem 3.4 (Parigot [20]) *If $\Gamma \vdash_{MC} A; \Delta$ is provable then there is a normal proof of $\Gamma \vdash_{MC} A; \Delta$*

We define *normal proofs* for classical natural deduction as for minimal classical natural deduction where the rule $\bot_e$ is normal if its premise is not an *Activate* rule (*i.e.* $\bot_e$ is considered at the same level as *Passivate*). Parigot's normalization proof for minimal classical natural deduction [21, 22] applies also for full classical natural deduction.

Theorem 3.5 (Parigot) *If $\Gamma \vdash_C A; \Delta$ is provable then there is a normal proof of $\Gamma \vdash_C A; \Delta$.*

As expected, full classical logic is conservative over minimal classical logic for formulas not mentioning the $\bot$ formula, as stated by the following consequence of Theorem 3.5.

Proposition 3.6 *If $\bot$ does not occur in A then $\vdash_C A$ iff $\vdash_{MC} A$.*

Proof. The "if" part is trivial. For the "only if" part, let's apply Theorem 3.5 to get a normal form of $\vdash_C A$ and consider the more general case of a proof of $\Gamma \vdash_{MC} \Xi; \Delta$, where Ξ stands for an optional formula and $\bot$ does not occur in Γ.

We show by induction that the rule $\bot_e$ cannot occur in the proof. First, we have to observe that the only rule introducing a new hypothesis in Γ namely $\rightarrow_i$, preserves the property that $\bot$ does not occur in Γ. Next, we show that it is impossible that the rule $\bot_e$ occurs in the derivation. Assume that it were so, deriving $\Gamma \vdash_C ; \Delta$ from a proof π of $\Gamma \vdash_C \bot; \Delta$. Since π is normal, it does not start with an *Activate* rule so that it starts either by an axiom rule or by a sequence of $\rightarrow_e$ whose leftmost premise

$$
\begin{array}{rcl}
M & ::= & x \mid \lambda x.M \mid MM \\
\Gamma & ::= & \cdot \mid \Gamma, x : A
\end{array}
$$

$$
\frac{}{\Gamma, x : A \vdash x : A}\; Ax
$$

$$
\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B} \to_i \qquad \frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash M' : A}{\Gamma \vdash MM' : B} \to_e
$$

Figure 9: The λ-calculus and minimal logic

(by normality of π) is an axiom rule. This means that there should be an hypothesis of the form $A_1 \to ... \to A_n \to \bot$ in Γ which contradicts the fact that $\bot$ does not occur in the formulas of Γ.

Remark 3.7 *Without the rule Passivate minimal classical natural deduction yields minimal logic, since the context Δ is inert and can only remain empty in a derivation for which the end sequent has the form $\Gamma \vdash A$; (even the Activate rule cannot be applied). Similarly, classical natural deduction without the Passivate rule yields intuitionistic logic. As a consequence, minimal and intuitionistic natural deduction can both be seen as subsystems of classical natural deduction.*

4 Control Operators and Calculi

The logical systems introduced so far can be related to programming under the so called *Curry-Howard isomorphism.* The isomorphism establishes a correspondence between a proof of a formula A and a program of type A. Executing a program corresponds to *normalizing* a proof.

Minimal logic corresponds to the simply-typed λ-calculus [4]. As shown in Figure 9, the axiom of minimal logic corresponds to looking up an environment for a variable's typing, lambda abstraction corresponds to implication introduction, and function application corresponds to implication elimination. Execution is carried out by the β-rule:

$$
(\lambda x.N_1)N_2 \to N_1[N_2/x]
$$

where $N_1[N_2/x]$ stands for the capture-free substitution of N_2 for each free occurrences of x in N_1. A β-redex corresponds to a non normal occurrence of Modus Ponens and β-reduction eliminates these *detours*:

$$
\frac{\dfrac{\Gamma, x : A \vdash N_1 : B}{\Gamma \vdash \lambda x.N_1 : A \to B} \to_i \qquad \Gamma \vdash N_2 : A}{\Gamma \vdash (\lambda x.N_1)N_2 : B} \to_e
$$

$$
\begin{array}{|lcl|}
\hline
M & ::= & x \mid \lambda x.M \mid MN \mid \mathcal{A}\ M \mid \mathcal{K}\ M \mid \mathcal{C}\ M \\
V & ::= & x \mid \lambda x.M \\
E & ::= & \Box \mid E\ M \mid V\ E \\
\hline
\end{array}
$$

Figure 10: Syntax of λ_c

Griffin [13] was the first to extend the Curry-Howard isomorphism to classical logic. This entails extending the λ-calculus with control operators, which are reviewed next.

4.1 Control operators and their semantics

To reason about Scheme programs, Felleisen and Hieb [10] introduced the λ_c-calculus whose syntax is in Figure 10. The calculus extends the call-by-value λ-calculus with the operators *abort* ($\mathcal{A}$), *callcc* ($\mathcal{K}$), and $\mathcal{C}$. The operators $\mathcal{K}$ and $\mathcal{C}$ provide *abortive continuations*: the invocation of a continuation reinstates the captured context *in place* of the current one. Their semantics can be described most concisely using the following three operational rules, which rewrite complete programs:

$$
\begin{array}{rcl}
E[\mathcal{A}\ M] & \mapsto & M \\
E[\mathcal{K}\ M] & \mapsto & E[M\ (\lambda x.\ \mathcal{A}\ E[x])] \\
E[\mathcal{C}\ M] & \mapsto & M\ (\lambda x.\ \mathcal{A}\ E[x])
\end{array}
$$

In each of the rules, the entire program is split into an evaluation context E representing the continuation, and a current redex to rewrite. The operator $\mathcal{A}$ aborts the continuation returning its subexpression to the top-level; the other two operators capture the evaluation context E and reify it as a function $(\lambda x.\mathcal{A}\ E[x])$. When invoked, this function aborts the evaluation context at the point of invocation, and installs the captured context instead.

The rules show that $\mathcal{C}$ differs from $\mathcal{K}$ in that $\mathcal{C}$ does not duplicate the evaluation context:

$$
\begin{array}{lclcl}
\mathcal{C}\ (\lambda k.4) + 1 & \mapsto & (\lambda k.4)(\lambda z.z + 1) & \mapsto & 4 \\
\mathcal{K}\ (\lambda k.4) + 1 & \mapsto & ((\lambda k.4)(\lambda z.z + 1)) + 1 & \mapsto & 5
\end{array}
$$

This difference makes $\mathcal{C}$ at least as expressive as both $\mathcal{A}$ and $\mathcal{K}$; it can be used to define them as follows:

$$
\mathcal{A}\ M \triangleq \mathcal{C}\ (\lambda\_.\ M) \qquad \text{(Abbrev. 3)}
$$

$$
\mathcal{K}\ M \triangleq \mathcal{C}\ (\lambda c.\ c\ (M\ c)) \qquad \text{(Abbrev. 4)}
$$

where _ refers to an anonymous variable. The operator $\mathcal{K}$ is not as powerful as $\mathcal{C}$ [9]; expressing $\mathcal{C}$ using $\mathcal{K}$ is only possible if we also have the abort primitive $\mathcal{A}$:

$$
\mathcal{C}\ (M) \triangleq \mathcal{K}\ (\lambda k.\ \mathcal{A}\ (M\ k))
$$

$$
\begin{array}{rcl}
M & ::= & x \mid \lambda x.M \mid MM \mid \mathcal{C}\ M \\
V & ::= & x \mid \lambda x.M
\end{array}
$$

$$
\lambda_{n\mathcal{C}} \left\{
\begin{array}{llcl}
\beta : & (\lambda x.M)\ N & \to & M[N/x] \\
\mathcal{C}_L : & (\mathcal{C}\ M)\ N & \to & \mathcal{C}\ (\lambda k.M\ (\lambda f.\mathcal{A}\ (k\ (fN)))) \\
\mathcal{C}_{\mathsf{tp}} : & \mathcal{C}\ M & \to & \mathcal{C}\ (\lambda k.M\ (\lambda f.\mathcal{A}\ (k\ f))) \\
\mathcal{C}_{idem} : & \mathcal{C}\ (\lambda k.\mathcal{C}\ M) & \to & \mathcal{C}\ (\lambda k.M\ (\lambda x.\mathcal{A}\ (x))) \\
\mathcal{C}_{elim} : & \mathcal{C}\ (\lambda k.k\ M) & \to & M \qquad k \notin FV(M)
\end{array}
\right.
$$

$$
\lambda_{v\mathcal{C}} \left\{
\begin{array}{llcl}
\beta : & (\lambda x.M)\ V & \to & M[V/x] \\
\mathcal{C}_L : & (\mathcal{C}\ M)\ N & \to & \mathcal{C}\ (\lambda k.M\ (\lambda f.\mathcal{A}\ (k\ (fN)))) \\
\mathcal{C}_R : & V(\mathcal{C}\ M) & \to & \mathcal{C}\ (\lambda k.M\ (\lambda x.\mathcal{A}\ (k\ (Vx)))) \\
\mathcal{C}_{\mathsf{tp}} : & \mathcal{C}\ M & \to & \mathcal{C}\ (\lambda k.M\ (\lambda f.\mathcal{A}\ (k\ f))) \\
\mathcal{C}_{idem} : & \mathcal{C}\ (\lambda k.\mathcal{C}\ M) & \to & \mathcal{C}\ (\lambda k.M\ (\lambda x.\mathcal{A}\ (x)))
\end{array}
\right.
$$

Figure 11: Call-by-name and call-by-value λ_c reduction rules

In the sequel we focus on $\mathcal{C}$, but still occasionally treat $\mathcal{A}$ as a primitive control operator to provide more intuition.

4.2 Reduction Rules

Instead of presenting the semantics of control operators as a relation on complete programs, it is possible to give local reduction rules that are applicable anywhere in a term and in arbitrary order. The call-by-value and call-by-name reduction semantics of λ_c are presented in Figure 11.

It is possible to consider more rules, *e.g.* , η, but they are not needed for expressing evaluation. The rules simulate the capture of the evaluation context using several small steps: first the control operation is *lifted* across one context at a time until it reaches another control operator. At any point, it is possible to use $\mathcal{C}_{\mathsf{tp}}$ to start applying M to part of the captured context and then continue lifting the outer $\mathcal{C}$ to accumulate more of the context. The reduction rules are however not expressive enough to compute a value, which is obtainable by applying the computation rule (*i.e.* only applicable at the top-level):

$$\mathcal{C}\ M \mapsto M\ (\lambda x.\mathcal{A}\ (x)).$$

However, the reduction rules are powerful enough to delay the application of this rule until the end.

Example 4.1 *We illustrate call-by-value reduction:*

$$\begin{array}{lll}
3 + \mathcal{C}\ (\lambda k.2 + k1) & \to_{\mathcal{C}_R} & \mathcal{C}\ (\lambda q.\ (\lambda k.2 + k1)(\lambda x.\mathcal{A}\ (q\ (3 + x)))) \\
& \twoheadrightarrow & \mathcal{C}\ (\lambda q.\ 2 + \mathcal{A}\ (q\ 4)) \\
& \to_{\mathcal{C}_R} & \mathcal{C}\ (\lambda q.\ \mathcal{C}\ (\lambda k.(\lambda \delta.q\ 4)(\lambda x.\mathcal{A}\ (k\ (2 + x))))) \\
& \to_{\beta} & \mathcal{C}\ (\lambda q.\ \mathcal{C}\ (\lambda k.q\ 4)) \\
& \to_{\mathcal{C}_{idem}} & \mathcal{C}\ (\lambda q.\ (\lambda k.q\ 4)(\lambda x.\mathcal{A}\ x)) \\
& \to_{\beta} & \mathcal{C}\ (\lambda q.\ q\ 4)
\end{array}$$

Remark 4.2 *An important point to clarify is the presence of the abort operations in the right-hand sides of the reduction rules. As far as evaluation is concerned, they are not necessary. They are important in order to obtain a satisfying correspondence between the operational and reduction semantics. For example, by reducing the term in Example 4.1 with the reduction rules without the abort operations we have:*

$$\begin{array}{lll}
3 + \mathcal{C}\ (\lambda k.\ 2 + k\ 1) & \to_{\mathcal{C}_R} & \mathcal{C}\ (\lambda q.\ (\lambda k.\ 2 + k\ 1)(\lambda x.\ q\ (3 + x))) \\
& \twoheadrightarrow & \mathcal{C}\ (\lambda q.\ 2 + q\ 4)
\end{array}$$

*The absence of the abort makes it impossible to eliminate the control context $2+\Box$, at least without using some context-sensitive information about the binding of q. With the presence of the abort, the term includes specific information that q is not a normal function but is an abortive continuation which never returns to its caller. As we explain in Section 5, these abort steps are different from the abort used in defining $\mathcal{C}$ in terms of $\mathcal{K}$. The aborts in the reduction rules correspond to throwing to a user defined continuation (*i.e. *a Passivate step), whereas the abort in the definition of $\mathcal{C}$ corresponds to throwing to the predefined top-level continuation (*i.e. *a $\perp_e$ step).*

4.3 Griffin's Type System

Griffin noticed that the evaluation rule of $\mathcal{C}$ suggests the following type:

$$\frac{\Gamma \vdash M : (A \to B) \to B}{\Gamma \vdash \mathcal{C}\ (M) : A}$$

which would lead to the following typing of $\mathcal{A}$:

$$\frac{\Gamma \vdash M : B}{\Gamma \vdash \mathcal{A}\ (M) : A}$$

By reading types as propositions, the above rule would lead to an inconsistent system, since every formula would be provable. To solve the problem Griffin introduced a new type $\perp$ representing the proposition *false*, and set B to $\perp$:

$$\frac{\Gamma \vdash M : \perp}{\Gamma \vdash \mathcal{A}\ (M) : A}$$

which corresponds to the $\perp$-elimination rule.

In conclusion, with the addition of control operators we have a term assignment for intuitionistic and classical logic, as shown in Figures 12 and 13.

$$
\begin{array}{rcl}
M & ::= & x \mid \lambda x.M \mid MM \mid \mathcal{A}\ M \\
\Gamma & ::= & \cdot \mid \Gamma, x : A
\end{array}
$$

$$
\frac{}{\Gamma, x : A \vdash x : A}\ Ax \qquad \frac{}{\Gamma \vdash \mathcal{A} : \bot \to A}\ \mathrm{EFQ}
$$

$$
\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B} \to_i \qquad \frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash M' : A}{\Gamma \vdash MM' : B} \to_e
$$

Figure 12: The $\lambda_{\mathcal{A}}$ calculus and minimal logic + EFQ

$$
\begin{array}{rcl}
M & ::= & x \mid \lambda x.M \mid MM \mid \mathcal{C}\ M \\
\Gamma & ::= & \cdot \mid \Gamma, x : A
\end{array}
$$

$$
\frac{}{\Gamma, x : A \vdash x : A}\ Ax \qquad \frac{}{\Gamma \vdash \mathcal{C} : \neg\neg A \to A}\ \mathrm{DN}
$$

$$
\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B} \to_i \qquad \frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash M' : A}{\Gamma \vdash MM' : B} \to_e
$$

Figure 13: The $\lambda_{\mathcal{C}}$ calculus and minimal logic + DN

Proposition 4.3 *A formula A is provable in classical logic iff there exists a closed $\lambda_{\mathcal{C}}$ term M such that $\vdash M : A$ is provable.*

Defining $\mathcal{K}$ in terms of $\mathcal{C}$ (as in Abbrev. 4) produces the following typing for $\mathcal{K}$:

$$\mathcal{K} : (\neg A \to A) \to A$$

This type does not correspond to Peirce's law. We need an alternative operator which we call $\mathcal{K}_B$ defined as follows:

$$\mathcal{K}_B\ M \quad \triangleq \quad \mathcal{C}\ (\lambda c.\ c\ (M\ (\lambda x.\ \mathcal{A}\ (c\ x)))) \qquad \text{(Abbrev. 5)}$$

For example, instead of $\mathcal{K}\ (\lambda c.\ 1 + \mathcal{A}\ (c\ 1))$, we write $\mathcal{K}_B(\lambda c.\ 1 + c\ 1)$.

We arrive at the term assignment for minimal classical logic given in Figure 15, where $\neg_B A$ stands for $A \to B$. The reduction rules for $\mathcal{K}_B$ are given in Figure 14. It would seem natural to also include a rule similar to $\mathcal{C}_{idem}$ such as:

$$\mathcal{K}_B(\lambda k.\ \mathcal{K}_B M) \to \mathcal{K}_B(\lambda k.\ M\ k)$$

However, the above breaks subject reduction as the following example illustrates:

$$
\begin{array}{l}
\mathcal{K}_B\ (\lambda k.\ \mathcal{K}_B(\lambda q.\ \textbf{if}\ q\ 1\ \textbf{then}\ 7\ \textbf{else}\ k\ 99)) \quad \to \\
\mathcal{K}_B\ (\lambda k.\ \textbf{if}\ k\ 1\ \textbf{then}\ 7\ \textbf{else}\ k\ 99))
\end{array}
$$

From Proposition 3.2, we have the following result.

$$
\begin{array}{lcl}
M & ::= & x \mid \lambda x.M \mid MM \mid \mathcal{K}_B\ M \\
V & ::= & x \mid \lambda x.M \\
E & ::= & \Box \mid E\ M \mid V\ E \\
P & ::= & \Box \mid PM \mid MP \mid \lambda x.P \mid \mathcal{K}_B\ P
\end{array}
$$

$\lambda_{n\mathcal{K}_B}$:

$$
\begin{array}{lllll}
\beta : & (\lambda x.M)\ N & \rightarrow & M[N/x] & \\
\mathcal{K}_{B_L} : & (\mathcal{K}_B\ M)\ N & \rightarrow & \mathcal{K}_B\ (\lambda k.M\ (\lambda f.k\ (fN))\ N) & \\
\mathcal{K}_{B_{\mathsf{tp}}} : & \mathcal{K}_B\ (\lambda k.P[E[k\ M]]) & \rightarrow & \mathcal{K}_B\ (\lambda k.P[k\ M]) & \\
\mathcal{K}_{B_{elim}} : & \mathcal{K}_B\ (\lambda k.k\ M) & \rightarrow & M & k \notin FV(M)
\end{array}
$$

$\lambda_{v\mathcal{K}_B}$:

$$
\begin{array}{llll}
\beta : & (\lambda x.M)\ V & \rightarrow & M[V/x] \\
\mathcal{K}_{B_L} : & (\mathcal{K}_B\ M)\ N & \rightarrow & \mathcal{K}_B\ (\lambda k.M\ (\lambda f.k\ (fN))\ N) \\
\mathcal{K}_{B_R} : & V(\mathcal{K}_B\ M) & \rightarrow & \mathcal{K}_B\ (\lambda k.V\ (M\ (\lambda x.k\ (Vx)))) \\
\mathcal{K}_{B_{\mathsf{tp}}} : & \mathcal{K}_B\ (\lambda k.P[E[k\ M]]) & \rightarrow & \mathcal{K}_B\ (\lambda k.P[k\ M])
\end{array}
$$

Figure 14: Call-by-name and call-by-value $\lambda_{n\mathcal{K}_B}$ reduction rules

$$
\begin{array}{lcl}
M & ::= & x \mid \lambda x.M \mid MM \mid \mathcal{K}_B\ M \\
\Gamma & ::= & \cdot \mid \Gamma, x : A
\end{array}
$$

$$
\frac{}{\Gamma, x : A \vdash x : A}\ Ax \qquad \frac{}{\Gamma \vdash \mathcal{K}_B : (\neg_B A \rightarrow A) \rightarrow A}\ \mathrm{PL}
$$

$$
\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \rightarrow B} \rightarrow_i \qquad \frac{\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash M' : A}{\Gamma \vdash MM' : B} \rightarrow_e
$$

Figure 15: The $\lambda_{\mathcal{K}_B}$ calculus and minimal logic + PL

Proposition 4.4 *A formula A is provable in minimal classical logic iff there exists a closed $\lambda_{\mathcal{K}_B}$ term M such that $\vdash M : A$ is provable.*

Remark 4.5 *Parigot [19] criticized Griffin's work because the proposed $\mathcal{C}$-typing did not fit the operational semantics. Setting the type of a continuation to be $\neg A$ implies that the top-level has type $\bot$, but there is no closed term of type $\bot$, since $\bot$ corresponds to the empty type. Therefore, the typing is useless. Said otherwise, with the current typing for $\mathcal{C}$ we lose subject reduction. For example, in the following reduction:*

$$
\mathcal{C}\ (\lambda q.\ q\ 5) + 2 \mapsto (\lambda q.\ q\ 5)(\lambda x.\ \mathcal{A}\ (x + 2))\ .
$$

the left-hand side has type int, whereas the right-hand side does not type check: the abort is invoked with an int type instead of a $\bot$ type. To solve this conflict between $\bot$

$$
\begin{array}{rcl}
E & ::= & x \mid \lambda x.M \mid MM \mid \mathcal{C}\ M \\
V & ::= & x \mid \lambda x.M
\end{array}
$$

$$
\begin{array}{lcl}
\mathcal{C}\ (\lambda k.\ E[(\lambda x.M)\ V]) & \mapsto & \mathcal{C}\ (\lambda k.\ E[M[V/x])\\
\mathcal{C}\ (\lambda k.\ E[\mathcal{C}\ M]) & \mapsto & \mathcal{C}\ (\lambda k.\ M\ (\lambda x.\mathcal{A}\ (E[x])))\\
\mathcal{C}\ (\lambda k.\ k\ V) & \mapsto & V \qquad k \notin FV(V)
\end{array}
$$

Figure 16: Griffin's operational rules

and the top-level type, Griffin proposed to consider programs of the shape $\mathcal{C}(\lambda k.k\ M)$*. The top-level context* $\mathcal{C}(\lambda k.\Box)$ *contains a term of type* $\perp$ *which can be reduced using the reduction rules as shown in Figure 16.*

Example 4.6

$$
\begin{array}{ll}
\mathcal{C}\ (\lambda k.\ k\ (\mathcal{C}\ (\lambda q.\ q\ 5) + 2)) & \mapsto\\
\mathcal{C}\ (\lambda k.\ (\lambda q.\ q\ 5)(\lambda z.\ \mathcal{A}\ (k\ (z+2)))) & \mapsto\\
\mathcal{C}\ (\lambda k.\ (\lambda z.\ \mathcal{A}\ (k\ (z+2)))5) & \mapsto\\
\mathcal{C}\ (\lambda k.\ \mathcal{A}\ (k\ (5+2))) & \triangleq\\
\mathcal{C}\ (\lambda k.\ \mathcal{C}\ (\lambda d.\ (k\ (5+2)))) & \mapsto\\
\mathcal{C}\ (\lambda k.\ k\ (5+2)) & \mapsto\\
\mathcal{C}\ (\lambda k.\ k\ 7) & \mapsto\\
7 &
\end{array}
$$

The conversion between $\perp$ *and* int *is done in the last reduction step, which corresponds to binding* k *to the continuation* $\lambda z.z$ *instead of* $\lambda z.\mathcal{A}\ (z)$*, which would lead to a type conflict:*

$$\mathcal{C}\ (\lambda k.\ k\ 7) \mapsto (\lambda k.\ k\ 7)(\lambda z.\ \mathcal{A}\ (z))$$

As detailed in the next section, the classical version of Parigot's $\lambda\mu$ *requires a similar intervention; a free continuation constant is needed. It is also important to underline that Griffin's typing is preserved by the reduction semantics of Figure 11, as was also emphasized by de Groote [7]. The only rule that breaks subject reduction is the top-level computation rule (*i.e. $\mathcal{C}\ M \mapsto M\ (\lambda x.\mathcal{A}\ (x))$*), which forces a conversion from* $\perp$ *to the top-level type.*

4.4 Curry-Howard Isomorphism

Summarizing the previous results we have:

λ-calculus $+\ \mathcal{A}$	corresponds to minimal logic + EFQ
λ-calculus $+\ \mathcal{C}$	corresponds to minimal logic + DN
λ-calculus $+\ \mathcal{K}$	corresponds to minimal logic + $\mathrm{PL}_{\perp}$
λ-calculus $+\ \mathcal{K}_B$	corresponds to minimal logic + PL

INTERMEZZO 4.7 *The encodings presented above can be derived by assigning terms to proofs.*

1. *For example, the encoding of $\mathcal{A}$ in terms of $\mathcal{C}$ (see Abbrev. 3) is derived by assigning a term to the proof of $\bot \to A$ in terms of elimination of double negation:*

$$\dfrac{\dfrac{\dfrac{}{\vdash \mathcal{C} : DN}\,Ax \qquad \dfrac{\dfrac{}{k : \neg A, y : \bot \vdash y : \bot}\,Ax}{y : \bot \vdash \lambda k.y : \neg\neg A}\,\to_i}{y : \bot \vdash \mathcal{C}(\lambda k.y) : A}\,\to_e}{\vdash \lambda y.\mathcal{C}(\lambda k.y) : \bot \to A}\,\to_i$$

2. *We derive the encoding of $\mathcal{K}$ in terms of $\mathcal{C}$ by assigning a term to the proof of $PL_\bot$ in terms of DN:*

$$\dfrac{\dfrac{\dfrac{}{\vdash \mathcal{C} : DN}\,Ax \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{y : \neg A \to A \vdash y : \neg A \to A}\,Ax \quad \dfrac{}{k : \neg A \vdash k : \neg A}\,Ax}{y : \neg A \to A, k : \neg A \vdash yk : A}\,\to_e}{y : \neg A \to A, k : \neg A \vdash k(yk) : \bot}\,\to_e}{y : \neg A \to A \vdash \lambda k.k(yk) : \neg\neg A}\,\to_i}{y : \neg A \to A \vdash \mathcal{C}(\lambda k.k(yk) : A}\,\to_e}{\vdash \lambda y.\mathcal{C}(\lambda k.k(yk)) : \neg A \to A \to A}\,\to_i$$

3. *We derive the encoding of $\mathcal{C}$ in terms of $\mathcal{K}$ by assigning a term to the proof of DN in terms of $PL_\bot$ and EFQ:*

$$\dfrac{\dfrac{\dfrac{}{\vdash \mathcal{K} : PL_\bot}\,Ax \qquad \dfrac{\dfrac{\dfrac{}{\vdash \mathcal{A} : EFQ}\,Ax \quad \dfrac{\dfrac{}{z : \neg\neg A \vdash z : \neg\neg A}\,Ax \quad \dfrac{}{x : \neg A \vdash x : \neg A}\,Ax}{z : \neg\neg A, x : \neg A \vdash z\ x : \bot}\,\to_e}{z : \neg\neg A, x : \neg A \vdash \mathcal{A}\ (z\ x) : A}\,\to_e}{z : \neg\neg A \vdash \lambda x.\mathcal{A}\ (z\ x) : \neg A \to A}\,\to_i}{z : \neg\neg A \vdash \mathcal{K}(\lambda x.\mathcal{A}\ (z\ x)) : A}\,\to_e}{\vdash \lambda z.\mathcal{K}(\lambda x.\mathcal{A}\ (z\ x)) : \neg\neg A \to A}\,\to_i$$

4. *We derive the definition of $\mathcal{K}_B$ in terms of $\mathcal{K}_\bot$ by assigning a term to the proof of PL in terms of $PL_\bot$, where we let $\Gamma = \{y : \neg_B A \to A\}$ and $\Gamma_1 = \{x : A, z : \neg A\}$. To avoid clutter in the proof, instead of the axioms $\vdash \mathcal{A} : EFQ$ and $\vdash \mathcal{K} : PL_\bot$, we simply write $\mathcal{A}$ and $\mathcal{K}$.*

$$
\begin{prooftree}
\AxiomC{$\mathcal{K}$}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$\Gamma \vdash y : \neg_B A \to A$}
\AxiomC{$\mathcal{A}$}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$z : \neg A \vdash z : \neg A$}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$x : A \vdash x : A$}
\RightLabel{$\to_e$}
\BinaryInfC{$\Gamma_1 \vdash z\ x : \bot$}
\RightLabel{$\to_e$}
\BinaryInfC{$\Gamma_1 \vdash \mathcal{A}\ (z\ x) : B$}
\RightLabel{$\to_i$}
\UnaryInfC{$z : \neg A \vdash \lambda x.\mathcal{A}\ (z\ x) : A \to B$}
\RightLabel{$\to_e$}
\BinaryInfC{$\Gamma, z : \neg A \vdash y\ \lambda x.\mathcal{A}\ (z\ x) : A$}
\RightLabel{$\to_i$}
\UnaryInfC{$\Gamma \vdash \lambda z.(y\ \lambda x.\mathcal{A}\ (z\ x)) : \neg A \to A$}
\RightLabel{$\to_e$}
\BinaryInfC{$\Gamma \vdash \mathcal{K}(\lambda z.(y\ \lambda x.\mathcal{A}\ (z\ x))) : A$}
\RightLabel{$\to_i$}
\UnaryInfC{$\vdash \lambda y.\mathcal{K}(\lambda z.(y\ \lambda x.\mathcal{A}\ (z\ x))) : (\neg_B A \to A) \to A$}
\end{prooftree}
$$

We can write the above term in ML as follows:

```
- fun PL y = callcc (fn z => (y  (fn x => throw z x)))
val PL = fn : (('a -> 'b) -> 'a) -> 'a
```

If we compare the ML term and the PL proof assignment, we notice that one term contains an abort invocation whereas the other one contains a throw construct. This disparity was already pointed out before when we stressed that the proof of Peirce's law should not invoke EFQ or equivalently an abort statement. That disparity leads to the introduction of an alternative of Prawitz's logic which still needs a term assignment. We turn to its definition in the next section, which justifies the distinction between $\bot$ and $\bot\!\!\!\bot$.

5 Computational Content of Prawitz's Classical Deduction

We introduce a refinement of $\lambda_{\mathcal{C}}$ called the $\lambda_{\mathcal{C}^-\text{tp}}$-calculus which is better-behaved as a reduction system than $\lambda_{\mathcal{C}}$. The set of $\lambda_{\mathcal{C}}$ terms extends the λ-calculus terms with terms of the form $\mathcal{C}^-(\lambda k.\ J)$ where J, which stands for a *jump*, is defined as follows:

$$J ::= k\ M \mid \mathtt{tp}\ M$$

The meta-variable k ranges over continuation variables, which are distinct from regular variables. The continuation $\mathtt{tp}$ is a special constant which denotes the top-level. We call the control operator $\mathcal{C}^-$ instead of $\mathcal{C}$ since we are adopting some syntactic restrictions. The operator $\mathcal{C}^-$ is always applied to a lambda-abstraction whose body is a jump. Moreover, the operator $\mathcal{C}^-$ does not appear as constant but as a construct.

The translation from $\lambda_{\mathcal{C}}$ to $\lambda_{\mathcal{C}^-\text{tp}}$ is given in Figure 19. In the new calculus, it is possible to distinguish between capturing a continuation and expressing where to go next. The embedding of $\mathcal{C}$ explicitly expresses the jump to the top-level, and the

$$
\begin{array}{rcl}
M & ::= & x \mid MM \mid \lambda x.M \mid \mathcal{C}^{\text{-}}(\lambda k.\ J) \\
J & ::= & k'M \mid \mathsf{tp}\ M \\
\Gamma & ::= & \cdot \mid \Gamma, x : A \mid \Gamma, k : \neg_{\perp\!\!\!\perp} A
\end{array}
$$

$$
\frac{}{\Gamma, x : A \vdash x : A}\ Ax
$$

$$
\frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash M' : A}{\Gamma \vdash MM' : B} \to_e \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B} \to_i
$$

$$
\frac{\Gamma, k : \neg_{\perp\!\!\!\perp} A \vdash M : A}{\Gamma, k : \neg_{\perp\!\!\!\perp} A \vdash k\ M : \perp\!\!\!\perp} \perp\!\!\!\perp_i \qquad \frac{\Gamma, k : \neg_{\perp\!\!\!\perp} A \vdash J : \perp\!\!\!\perp}{\Gamma \vdash \mathcal{C}^{\text{-}}(\lambda k.\ J) : A}\ RAA_{\perp\!\!\!\perp}
$$

$$
\frac{\Gamma \vdash M : \perp}{\Gamma \vdash \mathsf{tp}\ M : \perp\!\!\!\perp} \perp_e
$$

Figure 17: $\lambda_{\mathcal{C}^{\text{-}}\mathsf{tp}}$ and Prawitz's Classical Logic

$$
\begin{array}{rcl}
M & ::= & x \mid MM \mid \lambda x.M \mid \mathcal{C}^{\text{-}}(\lambda k.\ J) \\
J & ::= & k'M \\
\Gamma & ::= & \cdot \mid \Gamma, x : A \mid \Gamma, k : \neg_{\perp\!\!\!\perp} A
\end{array}
$$

$$
\frac{}{\Gamma, x : A \vdash x : A}\ Ax
$$

$$
\frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash M' : A}{\Gamma \vdash MM' : B} \to_e \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B} \to_i
$$

$$
\frac{\Gamma, k : \neg_{\perp\!\!\!\perp} A \vdash M : A}{\Gamma, k : \neg_{\perp\!\!\!\perp} A \vdash k\ M : \perp\!\!\!\perp} \perp\!\!\!\perp_i \qquad \frac{\Gamma, k : \neg_{\perp\!\!\!\perp} A \vdash J : \perp\!\!\!\perp}{\Gamma \vdash \mathcal{C}^{\text{-}}(\lambda k.\ J) : A}\ RAA_{\perp\!\!\!\perp}
$$

Figure 18: $\lambda_{\mathcal{C}^{\text{-}}}$ and Prawitz's Minimal Classical Logic

$$
\begin{array}{l}
x^\circ = x \\
(\lambda x.M)^\circ = \lambda x.M^\circ \\
(MN)^\circ = M^\circ\ N^\circ \\
(\mathcal{C}\ M)^\circ = \mathcal{C}^{\text{-}}(\lambda k.\mathsf{tp}\ (M^\circ(\lambda x.\mathit{throw}\ k\ x))) \\
(\mathcal{K}\ M)^\circ = \mathcal{C}^{\text{-}}(\lambda k.k\ (M^\circ(\lambda x.\mathit{throw}\ k\ x)))
\end{array}
$$

Figure 19: Embedding of λ_c into $\lambda_{\mathcal{C}^{\text{-}}\mathsf{tp}}$

embedding of $\mathcal{K}$ explicitly expresses that control goes back to the current context. These things are left implicit in $\lambda_{\mathcal{C}}$.

To summarize, aborting a computation (*i.e.* , throwing to the top-level continuation) is written as:

$$\mathcal{A}^{-}\ M \quad \triangleq \quad \mathcal{C}^{-}(\lambda\_.\ \mathsf{tp}\ M) \qquad \text{(Abbrev. 6)}$$

and throwing to a user-defined continuation is written as:

$$\textit{throw}\ k\ M \quad \triangleq \quad \mathcal{C}^{-}(\lambda\_.\ k\ M) \qquad \text{(Abbrev. 7)}$$

5.1 Type System

We adopt two types of judgments:

$$\Gamma \vdash M : A \text{ and } \Gamma \vdash J : \perp\!\!\!\perp$$

Since the invocation of a continuation leaves the current context, it intuitively corresponds to a sequent with no conclusion. Instead of typing a continuation as $A \rightarrow \bot$ we use the type $A \rightarrow \perp\!\!\!\perp$, thus distinguishing the top-level type from the result type of a continuation. However, we still need a conversion from $\perp\!\!\!\perp$ to any other type in order to write terms such as $1 + k\ 1$ where k is a continuation. This conversion is done by the *Weakening* rule. The invocation of a continuation simply corresponds to an implication elimination. However, if the argument has type $\bot$, which is the top-level type, the special continuation denoting the top-level is called.

Example 5.1 *We can now provide a term assignment for the proof of DN given in Example 2.9.*

$$\begin{prooftree}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$y : \neg\neg A \vdash y : \neg\neg A$}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$k : a \rightarrow \perp\!\!\!\perp, x : A \vdash x : A$}
\RightLabel{$\perp\!\!\!\perp_i$}
\UnaryInfC{$k : A \rightarrow \perp\!\!\!\perp, x : A \vdash k\ x : \perp\!\!\!\perp$}
\RightLabel{$Weak.$}
\UnaryInfC{$k : A \rightarrow \perp\!\!\!\perp, x : A \vdash \textit{throw}\ k\ x : \bot$}
\RightLabel{$\rightarrow_i$}
\UnaryInfC{$k : A \rightarrow \perp\!\!\!\perp \vdash \lambda x.\textit{throw}\ k\ x : \neg A$}
\RightLabel{$\rightarrow_e$}
\BinaryInfC{$k : A \rightarrow \perp\!\!\!\perp, y : \neg\neg A \vdash y\ (\lambda x.\textit{throw}\ k\ x) : \bot$}
\RightLabel{$\bot_e$}
\UnaryInfC{$k : A \rightarrow \perp\!\!\!\perp, y : \neg\neg A \vdash \mathsf{tp}\ (y\ (\lambda x.\textit{throw}\ k\ x)) : \perp\!\!\!\perp$}
\RightLabel{$RAA_{\perp\!\!\!\perp}$}
\UnaryInfC{$y : \neg\neg A \vdash \mathcal{C}^{-}(\lambda k.\ \mathsf{tp}\ (y\ (\lambda x.\textit{throw}\ k\ x))) : A$}
\RightLabel{$\rightarrow_i$}
\UnaryInfC{$\vdash \lambda y.\mathcal{C}^{-}(\lambda k.\ \mathsf{tp}\ (y\ (\lambda x.\textit{throw}\ k\ x))) : \neg\neg A \rightarrow A$}
\end{prooftree}$$

The term refers to the top-level continuation as expected.

Proposition 5.2 *A formula A is provable in Prawitz's classical logic iff there exists a closed $\lambda_{\mathcal{C}^{-}\mathsf{tp}}$ term M such that $\vdash M : A$ is provable.*

We define a subset of $\lambda_{\mathcal{C}^{-}\mathsf{tp}}$, which does not allow terms of the form $\mathcal{C}^{-}(\lambda k.\mathsf{tp}\ M)$. This subset called $\lambda_{\mathcal{C}^{-}}$ is defined in Figure 18.

Example 5.3 *The term corresponding to PL is given below, where we let* $\Gamma = \{k : \neg_{\perp\!\!\!\perp} A, y : (\neg_B A) \to A, x : A\}$*,* $\Gamma_1 = \{k : \neg_{\perp\!\!\!\perp} A, y : (\neg_B A) \to A\}$*, and* $\Gamma_2 = \{y : (\neg_B A) \to A\}$*:*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma_1 \vdash y : \neg_B A \to A}\,Ax \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash x : A}\,Ax}{\Gamma \vdash k\ x : \perp\!\!\!\perp}\,\perp\!\!\!\perp_i}{\Gamma \vdash \mathit{throw}\ k\ x : B}\,Weak.}{\Gamma_1 \vdash \lambda x.\mathit{throw}\ k\ x : \neg_B A}\,\to_i}{\Gamma_1 \vdash (y\ (\lambda x.\mathit{throw}\ k\ x)) : A}\,\to_e}{\Gamma_1 \vdash k\ (y\ (\lambda x.\mathit{throw}\ k\ x)) : \perp\!\!\!\perp}\,\perp\!\!\!\perp_i}{\Gamma_2 \vdash \mathcal{C}^{-}(\lambda k.k\ (y\ (\lambda x.\mathit{throw}\ k\ x))) : A}\,RAA_{\perp\!\!\!\perp}}{\vdash \lambda y.\ \mathcal{C}^{-}(\lambda k.k\ (y\ (\lambda x.\mathit{throw}\ k\ x))) : (\neg_B A \to A) \to A}\,\to_i$$

which in ML's syntax is:

```
- fun PL y = callcc (fn k => (y  (fn x => throw k x)))
```

Proposition 5.4 *A formula A is provable in minimal Prawitz's classical logic iff there exists a closed* $\lambda_{\mathcal{C}^-}$ *term* M *such that* $\vdash M : A$ *is provable.*

By Propositions 3.1, 4.4 and 5.4, $\lambda_{\mathcal{C}^-}$ is equivalent to $\lambda_{\mathcal{K}_B}$. However, it might not be at all obvious how in $\lambda_{\mathcal{K}_B}$ to use a continuation in different contexts, since we do not have weakening available. Consider for example the following $\lambda_{\mathcal{C}^-}$ term:

$$\mathcal{C}^{-}(\lambda k.\ k\ (\textbf{if}\ \mathit{throw}\ k\ 1\ \textbf{then}\ 7\ \textbf{else}\ \mathit{throw}\ k\ 99))$$

We use the continuation in both boolean and integer contexts. How can we write the above expression without making use of weakening or throw? The proof of Proposition 3.2 gives the answer:

$$\mathcal{K}_B\ (\lambda k.\ \mathcal{K}_B\ (\lambda q.\ \textbf{if}\ q\ 1\ \textbf{then}\ 7\ \textbf{else}\ k\ 99))$$

We define a further subset of $\lambda_{\mathcal{C}^-}$ which allows only jumps to the top-level, see Figure 20.

Proposition 5.5 *A formula A is provable in intuitionistic logic iff there exists a closed* $\lambda_{\mathcal{A}^-}$ *term* M *such that* $\vdash M : A$ *is provable.*

5.2 Reduction Semantics

The call-by-name and call-by-value $\lambda_{\mathcal{C}^-\mathtt{tp}}$ reduction rules are given in Figure 21. The rule $\mathcal{C}_{top}$, whose action is to wrap an application of a continuation with a throw operation, is not needed. The rule $\mathcal{C}^{-}_{idem'}$ is a special case of $\mathcal{C}_{idem}$ where the continuation k' is tp. The rule $\mathcal{C}^{-}_{idem}$ is similar to the rule proposed by Barbanera and Berardi [3]:

$$M\ (\mathcal{C}\ N) \to N\ (\lambda a.\ (M\ a))\ ,$$

$$
\begin{array}{rcl}
M & ::= & x \mid MM \mid \lambda x.M \mid \mathcal{C}^{-}(\lambda\_.\ J) \\
J & ::= & \mathsf{tp}\ M \\
\Gamma & ::= & \cdot \mid \Gamma, x : A
\end{array}
$$

$$
\frac{}{\Gamma, x : A \vdash x : A}\ Ax
$$

$$
\frac{\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash M' : A}{\Gamma \vdash MM' : B} \rightarrow_e \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \rightarrow B} \rightarrow_i
$$

$$
\frac{\Gamma \vdash J : \perp\!\!\!\perp}{\Gamma \vdash \mathcal{C}^{-}(\lambda\_.\ J) : A}\ Weakening \qquad \frac{\Gamma \vdash M : \perp}{\Gamma \vdash \mathsf{tp}\ M : \perp\!\!\!\perp} \perp_e
$$

Figure 20: $\lambda_{\mathcal{A}^{-}}$ and Intuitionistic Logic

$$
V ::= x \mid \lambda x.M
$$

$\lambda_{n\mathcal{C}^{-}}$ and $\lambda_{n\mathcal{C}^{-}\mathsf{tp}}$

$$
\begin{array}{lll}
\beta : & (\lambda x.\ M)\ N & \rightarrow\ M[N/x] \\
\mathcal{C}^{-}_{L} : & (\mathcal{C}^{-}\ \lambda k.\ J)\ N & \rightarrow\ \mathcal{C}^{-}(\lambda k.\ J[k\ (PN)/k\ P]) \\
\mathcal{C}^{-}_{idem} : & \mathcal{C}^{-}(\lambda k.\ k'\ (\mathcal{C}^{-}\ (\lambda q.J)) & \rightarrow\ \mathcal{C}^{-}(\lambda k.\ J[k'/q]) \\
\mathcal{C}^{-}_{idem'} : & \mathcal{C}^{-}(\lambda k.\ \mathsf{tp}\ (\mathcal{C}^{-}\ (\lambda q.J)) & \rightarrow\ \mathcal{C}^{-}(\lambda k.\ J[\mathsf{tp}/q]) \\
\mathcal{C}^{-}_{elim} : & \mathcal{C}^{-}(\lambda k.\ k\ M) & \rightarrow\ M \quad k \notin FV(M)
\end{array}
$$

$\lambda_{v\mathcal{C}^{-}}$ and $\lambda_{v\mathcal{C}^{-}\mathsf{tp}}$

$$
\begin{array}{lll}
\beta : & (\lambda x.M)V & \rightarrow\ M[V/x] \\
\mathcal{C}^{-}_{elim} : & \mathcal{C}^{-}(\lambda k.\ k\ M) & \rightarrow\ M \quad k \notin FV(M) \\
\mathcal{C}^{-}_{L} : & (\mathcal{C}^{-}(\lambda k.\ J))\ N & \rightarrow\ \mathcal{C}^{-}(\lambda k.\ J[k\ (PN)/k\ P]) \\
\mathcal{C}^{-}_{R} : & V\ (\mathcal{C}^{-}(\lambda k.\ J)) & \rightarrow\ \mathcal{C}^{-}(\lambda k.\ J[k\ (VP)/k\ P]) \\
\mathcal{C}^{-}_{idem} : & \mathcal{C}^{-}(\lambda k.\ k'\ (\mathcal{C}^{-}\ (\lambda q.\ J))) & \rightarrow\ \mathcal{C}^{-}(\lambda k.\ J[k'/q]) \\
\mathcal{C}^{-}_{idem'} : & \mathcal{C}^{-}(\lambda k.\ \mathsf{tp}\ (\mathcal{C}^{-}\ (\lambda q.\ J))) & \rightarrow\ \mathcal{C}^{-}(\lambda k.\ J[\mathsf{tp}/q])
\end{array}
$$

Figure 21: Call-by-name and call-by-value $\lambda_{\mathcal{C}^{-}}$ and $\lambda_{\mathcal{C}^{-}\mathsf{tp}}$ reduction rules

where M has type $\neg A$. Felleisen and Hieb [10] proposed the following additional rules for $\lambda_{v\mathcal{C}}$:

$$\mathcal{C}_E : E[\mathcal{C}\ M] \to \mathcal{C}\ (\lambda k.\ M\ (\lambda x.\ \mathcal{A}\ (k\ E[x])))$$

(where E stands for a call-by-value evaluation context) and

$$\mathcal{C}_{elim} : \mathcal{C}\ (\lambda k.\ k\ M) \to M\ \ ,$$

where k is not free in M. The first rule is a generalization of $\mathcal{C}_L$, $\mathcal{C}_R$, and $\mathcal{C}_{\mathsf{tp}}$ which adds expressive power to the calculus. The second rule, which is also used by Hofmann [14], leads to better simulation of evaluation. However, both rules destroy confluence of $\lambda_{v\mathcal{C}}$ as the following example illustrates.

Example 5.6 *In the following diagram if M does not reduce to a value then the two reduction sequences in the extended $\lambda_{v\mathcal{C}}$ cannot be brought together.*

$$\begin{array}{lcl}
\mathcal{C}(\lambda k.\ k\ M) & \to & M \\
\downarrow & & \\
\mathcal{C}(\lambda q.\ (\lambda k.\ k\ M)(\lambda x.\ \mathcal{A}\ (q\ x))) & & \\
\downarrow & & \\
\mathcal{C}(\lambda q.\ (\lambda x.\ \mathcal{A}\ (q\ x))\ M) & &
\end{array}$$

Felleisen *et al.* left unresolved the problem of finding an extended theory that would include $\mathcal{C}_E$ or $\mathcal{C}_{elim}$ and still satisfy the classical properties of reduction theories. Since $\mathcal{C}_{elim}$ is already present in our calculi and $\mathcal{C}_E$ is derivable, one may consider our calculi as a solution.

Proposition 5.7 *1. $\lambda_{v\mathcal{C}^-\mathsf{tp}}$ and $\lambda_{n\mathcal{C}^-\mathsf{tp}}$ are confluent and strongly normalizing.*

2. Subject reduction: Given $\lambda_{v\mathcal{C}^-\mathsf{tp}}$ ($\lambda_{n\mathcal{C}^-\mathsf{tp}}$) terms M, N, if $\Gamma \vdash M : A$ and $M \twoheadrightarrow N$ then $\Gamma \vdash N : A$.

Proof. As shown in the next section, the $\lambda_{n\mathcal{C}^-\mathsf{tp}}$ calculus corresponds to Parigot's call-by-name $\lambda\mu$ calculus, which is confluent and strongly normalizable [24, 21, 22]. The $\lambda_{v\mathcal{C}^-\mathsf{tp}}$ calculus corresponds to a subset of the $\lambda\mu_v$ calculus of Ong and Stewart [18] which is strongly normalizing. Confluence follows from the fact that all critical pairs converge.

Soundness and completeness properties for $\lambda_{v\mathcal{C}^-\mathsf{tp}}$ with respect to $\lambda_{v\mathcal{C}}$ are stated in terms of *operational equivalence*. Given a reduction relation X, and two terms M and N, possibly containing free variables, we say $M \simeq_X N$ if for every context P which binds all the free variables of M and N, $P[M] \twoheadrightarrow_X V_1$ iff $P[N] \twoheadrightarrow_X V_2$ for some values V_1 and V_2. For example, any two terms that are convertible using the reduction relation are operationally equivalent. Two non-convertible terms may still be equivalent if no sequence of reductions in any context can invalidate their equivalence. An example of this kind is the equivalence $(\lambda x.x)\ (y\ z) \simeq_{\lambda_{\mathcal{C}}} (y\ z)$. The embedding of $\lambda_{v\mathcal{C}^{-}\mathsf{tp}}$ into $\lambda_{v\mathcal{C}}$ essentially removes occurrences of the top-level continuation, which is implicit in $\lambda_{v\mathcal{C}}$:

$$\begin{array}{rcl} (\mathcal{C}^{-}(\lambda k.J))^{\bullet} & = & \mathcal{C}\ (\lambda k.J^{\bullet}) \\ (\mathsf{tp}\ M)^{\bullet} & = & M^{\bullet} \end{array}$$

Proposition 5.8 *Let M be a* closed *$\lambda_{v\mathcal{C}}$ term:*

- *If $M \twoheadrightarrow_{\lambda_{v\mathcal{C}}} V$ then $M^{\circ} \simeq_{\lambda_{v\mathcal{C}^{-}tp}} V^{\circ}$.*
- *If $M^{\circ} \twoheadrightarrow_{\lambda_{v\mathcal{C}^{-}tp}} V$ then $M \simeq_{\lambda_{v\mathcal{C}}} V^{\bullet}$.*

The proof of the first clause reduces to checking that embedding both sides of every $\lambda_{v\mathcal{C}}$-reduction produces semantically-equivalent terms in $\lambda_{v\mathcal{C}^{-}\mathsf{tp}}$. For some of the cases embedded terms are related by a corresponding reduction in $\lambda_{v\mathcal{C}^{-}\mathsf{tp}}$ and hence are obviously semantically-equivalent. For the $\mathcal{C}_{idem}$ case, the embedded left-hand side does not reduce to the embedded right-hand side, but both can reduce to a common term, and hence are again semantically-equivalent. The left-hand side and right-hand side of the $\mathcal{C}_{\mathsf{tp}}$ rule map into convertible terms. The lifting rules $\mathcal{C}_L$ are $\mathcal{C}_R$ introduce a complication: proving the equivalence of the embedded terms requires using the following equivalence:

$$(\lambda x.\mathit{throw}\ k\ x)\ M \quad \simeq_{\lambda_{\mathcal{C}}} \quad \mathit{throw}\ k\ M$$

even when M is not a value. This happens because in contrast to the regular substitution operation, structural substitutions can replace arbitrary jumps $(k\ M)$ by $(k\ (V\ M))$ even when M is not a value.

The proof of the second clause reduces to proving:

1. For all $\lambda_{v\mathcal{C}}$-terms M, we have $M \simeq_{\lambda_{v\mathcal{C}}} M^{\circ\bullet}$
2. For every $\lambda_{v\mathcal{C}^{-}\mathsf{tp}}$-reduction $M \to N$, we have $M^{\bullet} \simeq_{\lambda_{\mathcal{C}}} N^{\bullet}$.

The proof of the first statement is almost straightforward: proving that $\mathcal{C}\ M$ is equivalent to $(\mathcal{C}\ M)^{\circ\bullet}$ requires using $\mathcal{C}_{\mathsf{tp}}$ which is not a reduction rule but otherwise a valid operational equivalence.

When attempting to prove the second statement, we encounter a problem related to free continuation variables. Even though programs are closed terms, reductions can happen anywhere including under binders and hence it is possible for a $\lambda_{v\mathcal{C}^{-}\mathsf{tp}}$-reduction to manipulate an open term. In particular, consider $\mathcal{C}^{-}_{idem}$ where the continuation variable k' is free. The right-hand side maps to the $\lambda_{v\mathcal{C}}$-term $\mathcal{C}(\lambda k.\ J[k'/q]^{\bullet})$ but for the left-hand side we have:

$$\begin{array}{ll} \mathcal{C}(\lambda k.\ k'\ \mathcal{C}(\lambda q.\ J^{\bullet})) & \to \\ \mathcal{C}(\lambda k.\ \mathcal{C}(\lambda r.\ (\lambda q.\ J^{\bullet})\ (\lambda x.\ \mathcal{A}\ (r\ (k'\ x))))) & \to \\ \mathcal{C}(\lambda k.\ \mathcal{C}(\lambda r.\ J^{\bullet}[\lambda x.\ \mathcal{A}\ (r\ (k'\ x))/q])) & \to \\ \mathcal{C}(\lambda k.J^{\bullet}[\lambda x.\ \mathcal{A}\ ((\lambda x.\mathcal{A}\ x)\ (k'\ x))/q]) & \end{array}$$

which is equivalent to the $\lambda_{v\mathcal{C}}$-term $\mathcal{C}(\lambda k.J^\bullet[\lambda x.\ \mathcal{A}\ (k'\ x)/q])$. Since the variable k' is not special in $\lambda_{v\mathcal{C}}$ it could, as far as the $\lambda_{v\mathcal{C}}$-theory is concerned, be substituted with an arbitrary procedure and hence it is definitely not the case that one can assume that k' and $(\lambda x.\mathcal{A}\ (k'\ x))$ are operationally equivalent. This assumption would be correct if we could somehow guarantee that k' is substituted by a continuation. In a complete program, this is clearly the case as the left-hand side must occur in a context $\ldots\mathcal{C}^-(\lambda k'.\ldots\Box\ldots)\ldots$ which binds k' to a continuation variable. We just need to make this information explicit in the statement of the Proposition [25, Lemma 19]:

2'. Let $M \to N$ be a $\lambda_{v\mathcal{C}^-\mathsf{tp}}$-reduction, and let $k_1, \ldots, k_n$ be the free continuation variables in M, then we have the equivalence

$$\mathcal{C}\ (\lambda k_1 \ldots \mathcal{C}\ (\lambda k_n.M^\bullet)) \simeq_{\lambda_\mathcal{C}} \mathcal{C}\ (\lambda k_1 \ldots \mathcal{C}\ (\lambda k_n.N^\bullet))$$

The proof of the modified clause proceeds by cases. For the reduction $\mathcal{C}^-_{elim}$ we use the fact that even though $\mathcal{C}_{elim}$ is not a reduction of $\lambda_{v\mathcal{C}}$, it is a valid equivalence. The reductions $\mathcal{C}^-_L$ and $\mathcal{C}^-_R$ require the following equivalences in $\lambda_{v\mathcal{C}}$ where k is a continuation variable and M may not be a value:

$$\begin{array}{lcl}(\lambda x.k\ (V\ x))\ M & \simeq_{\lambda_\mathcal{C}} & k\ (V\ M) \\ (\lambda x.k\ (x\ N))\ M & \simeq_{\lambda_\mathcal{C}} & k\ (M\ N)\end{array}$$

These again allow one to jump with a non-value. All the required $\lambda_{v\mathcal{C}}$ equivalences are known to be valid [16, 25].

Remark 5.9 *Reducing the term corresponding to $\mathcal{C}(\lambda k.\ k\ I\ x)\ 1$ we have:*

$$\begin{array}{ll}(\mathcal{C}^-(\lambda k.\ \mathsf{tp}\ ((\lambda q.q\ I\ x)(\lambda f.\ \mathit{throw}\ k\ f))))\ 1 & \to \\ (\mathcal{C}^-(\lambda k.\ \mathsf{tp}\ (((\lambda f.\ \mathit{throw}\ k\ f)\ I)\ x)))\ 1 & \to \\ (\mathcal{C}^-(\lambda k.\ \mathsf{tp}\ ((\mathit{throw}\ k\ I)\ x)))\ 1 & \to \\ (\mathcal{C}^-(\lambda k.\ \mathsf{tp}\ (\mathit{throw}\ k\ I)))\ 1 & \to \\ \mathcal{C}^-(\lambda k.\ \mathsf{tp}\ (\mathit{throw}\ k\ (I\ 1))) & \to \\ \mathcal{C}^-(\lambda k.\ k\ (I\ 1)) & \to \\ \mathcal{C}^-(\lambda k.\ k\ 1) & \to \\ 1 & \end{array}$$

This reduction sequence is better than the corresponding sequence in $\lambda_{v\mathcal{C}}$. However, the rules are still not complete with respect to evaluation. In particular, it is not possible to simulate the computation rules:

$$\begin{array}{lcl}\mathcal{C}^-(\lambda k.\ k.\ M) & \to & M[\mathsf{tp}/k] \\ \mathcal{C}^-(\lambda\_.\ \mathsf{tp}\ M) & \to & M\end{array}$$

For example, the reduction rules cannot reduce the following program:

$$\mathcal{C}^-(\lambda c.\ c\ (\lambda x.\ \mathit{throw}\ c\ (\lambda y.\ x)))$$

to $\lambda x.\mathcal{A}^-(\lambda y.\ x)$. To do that the calculus must be extended with a notion of prompt [2].

$$
\begin{array}{lcl}
t, x & ::= & x \mid \lambda x.t \mid t\ s \mid \mu\alpha.c \\
c & ::= & [\beta]t \mid [\mathsf{tp}]t \\
\Gamma & ::= & \cdot \mid \Gamma^x \\
\Delta & ::= & \cdot \mid \Delta^\alpha
\end{array}
$$

$$
\frac{}{\Gamma, A^x \vdash x : A; \Delta}\ Ax
$$

$$
\frac{\Gamma, A^x \vdash t : B; \Delta}{\Gamma \vdash \lambda x.t : A \to B; \Delta} \to_i \qquad \frac{\Gamma \vdash t : A \to B; \Delta \quad \Gamma \vdash s : A; \Delta}{\Gamma \vdash t\ s : B; \Delta} \to_e
$$

$$
\frac{\Gamma \vdash t : A; A^\alpha, \Delta}{[\alpha]t : \Gamma \vdash; A^\alpha, \Delta}\ \mathit{Passivate} \qquad \frac{c : \Gamma \vdash; A^\alpha, \Delta}{\Gamma \vdash \mu\alpha.c : A; \Delta}\ \mathit{Activate}
$$

$$
\frac{\Gamma \vdash t : \bot; \Delta}{[\mathsf{tp}]t : \Gamma \vdash; \Delta} \bot_e
$$

Figure 22: The $\lambda_{\mu\mathsf{tp}}$ calculus and Classical Natural Deduction

6 Computational Content of Classical Natural Deduction

Figure 22 describes the $\lambda\mu$ calculus of Parigot [19] which is a term assignment for classical natural deduction. The *Passivate* rule reads as follows: given a term producing a value of type A, if α is a continuation variable waiting for something of type A (*i.e.* A *cont*), then by invoking the continuation variable we leave the current context. Terms of the form $[\alpha]t$ are called *commands*. The *Activate* rule reads as follows: given a command (*i.e.* no formula is focused) we can select which result to get by capturing the associated continuation. If A^α is not present in the precondition then the rule corresponds to weakening. The rule $\bot_e$ differs from Parigot's version as follows. In the original formulation, the elimination rule for $\bot$ is interpreted by a named term $[\gamma]t$, where γ is any continuation variable (not always the same for every instance of the rule). In contrast, the rule is here systematically associated to the same primitive continuation variable, called tp, considered as a constant. This was also observed by Streicher and Reus [27]. In Parigot's style, DN is represented with the term:

$$\lambda y.\mu\alpha.[\gamma](y\ (\lambda x.\mu\delta.[\alpha]x))$$

whereas our representation is:

$$\lambda y.\mu\alpha.[\mathsf{tp}](y\ (\lambda x.\mu\delta.[\alpha]x))\ .$$

We use $\lambda_{\mu\mathsf{tp}}$ to denote the whole calculus with $\bot_e$ and $\lambda\mu$ to denote the calculus without $\bot_e$ (see Figure 23). The need for an extra continuation constant to interpret

$$
\begin{array}{rcl}
t, s & ::= & x \mid \lambda x.t \mid t\ s \mid \mu\alpha.c \\
c & ::= & [\beta]t
\end{array}
$$

$$
\frac{}{\Gamma, A^x \vdash x : A; \Delta}\ Ax
$$

$$
\frac{\Gamma, A^x \vdash t : B; \Delta}{\Gamma \vdash \lambda x.t : A \to B; \Delta} \to_i \qquad \frac{\Gamma \vdash t : A \to B; \Delta \quad \Gamma \vdash s : A; \Delta}{\Gamma \vdash t\ s : B; \Delta} \to_e
$$

$$
\frac{\Gamma \vdash t : A; A^\alpha, \Delta}{[\alpha]t : \Gamma \vdash; A^\alpha, \Delta}\ \textit{Passivate} \qquad \frac{c : \Gamma \vdash; A^\alpha, \Delta}{\Gamma \vdash \mu\alpha.c : A; \Delta}\ \textit{Activate}
$$

Figure 23: The $\lambda\mu$ calculus and Minimal Classical Natural Deduction

the elimination of $\perp$ can be emphasized by the following statement.

Proposition 6.1 *A formula A is provable in minimal classical logic (resp. classical logic) iff there exists a closed $\lambda\mu$ term (resp. $\lambda_{\mu tp}$ term) t such that $\vdash t : A$ is provable.*

We write $\lambda\mu_n$ and $\lambda\mu_v$ (resp. $\lambda_{n\mu\mathrm{tp}}$ and $\lambda_{v\mu\mathrm{tp}}$) for the $\lambda\mu$ calculus (resp. $\lambda_{\mu\mathrm{tp}}$ calculus) equipped with call-by-name and call-by-value reduction rules, respectively. The reduction rules are given in Figure 24 (substitutions $[[\alpha](ws)/[\alpha]w]$ and $[[\alpha](sw)/[\alpha]w]$ are defined as in the original formulation of the $\lambda\mu$ calculus [19]). The rules are the same for the $\lambda\mu$ and $\lambda_{\mu\mathrm{tp}}$ calculi. The calculus $\lambda\mu_n$ is Parigot's original calculus, while our presentation of $\lambda\mu_v$ is similar to the presentation of Ong and Stewart [18]. Both sets of reduction rules are well-typed and satisfy subject reduction.

6.1 Relation between the $\lambda_{\mu\mathrm{tp}}$ and the $\lambda_{\mathcal{C}^-\mathrm{tp}}$ calculi

The $\lambda_{\mu\mathrm{tp}}$ calculi and the $\lambda_{\mathcal{C}^-\mathrm{tp}}$ calculi are in one-to-one correspondence:

$$
\begin{array}{rcl}
\overline{\lambda x.t} & = & \lambda x.\overline{t} \\
\overline{ts} & = & \overline{t}\overline{s} \\
\overline{\mathcal{C}^-(\lambda\alpha.\gamma t)} & = & \mu\alpha.[\gamma]\overline{t}
\end{array}
$$

This correspondence extends to the reduction rules (Figure 21 matches Figure 24), as expressed by the following statement.

Lemma 6.2 *Let t, s be $\lambda_{\mu tp}$-terms, then:*

- $t \to_{\lambda_{n\mu tp}} s$ *iff* $\overline{t} \to_{\lambda_{n\mathcal{C}^- tp}} \overline{s}$
- $t \to_{\lambda_{v\mu tp}} s$ *iff* $\overline{t} \to_{\lambda_{v\mathcal{C}^- tp}} \overline{s}$.

$$v ::= x \mid \lambda x.t$$

$\lambda\mu_n$ and $\lambda_{n\mu\mathsf{tp}}$

Logical rule:	$(\lambda x.t)s$	$\to$	$t[s/x]$
Structural rule:	$(\mu\alpha.t)s$	$\to$	$(\mu\alpha.t[[\alpha](ws)/[\alpha]w])$
Renaming rule:	$\mu\alpha.[\beta]\mu\gamma.u$	$\to$	$\mu\alpha.u[\beta/\gamma]$
Renaming rule':	$\mu\alpha.[\mathsf{tp}]\mu\gamma.u$	$\to$	$\mu\alpha.u[\mathsf{tp}/\gamma]$
Simplification rule:	$\mu\alpha.[\alpha]u$	$\to$	$u \qquad \alpha \notin FV(u)$

$\lambda\mu_v$ and $\lambda_{v\mu\mathsf{tp}}$

Logical rule:	$(\lambda x.t)v$	$\to$	$t[v/x]$
Left structural rule:	$(\mu\alpha.t)s$	$\to$	$(\mu\alpha.t[[\alpha](ws)/[\alpha]w])$
Right structural rule:	$v(\mu\alpha.t)$	$\to$	$(\mu\alpha.t[[\alpha](vw)/[\alpha]w])$
Renaming rule:	$\mu\alpha.[\beta]\mu\gamma.u$	$\to$	$\mu\alpha.u[\beta/\gamma]$
Renaming rule':	$\mu\alpha.[\mathsf{tp}]\mu\gamma.u$	$\to$	$\mu\alpha.u[\mathsf{tp}/\gamma]$
Simplification rule:	$\mu\alpha.[\alpha]u$	$\to$	$u \qquad \alpha \notin FV(u)$

Figure 24: Call-by-name and call-by-value $\lambda\mu$ and $\lambda_{\mu\mathsf{tp}}$ reduction rules

The above proposition implies that Parigot's $\lambda\mu$ is a correct implementation of $\lambda_{\mathcal{C}}$. Correctness of Parigot's $\lambda\mu$ with respect to a modified reduction theory for $\mathcal{C}$ was already shown by de Groote [7]. However, the reduction rules did not contain the abort steps. The relation between $\lambda\mu$ and this modified theory was also studied by Ong and Stewart [18].

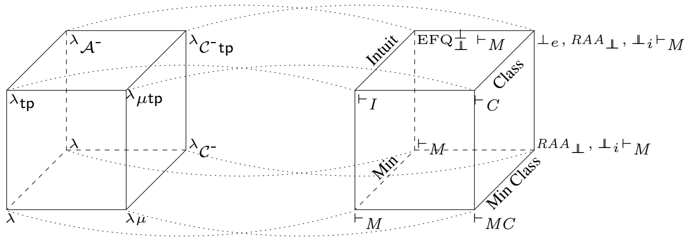
Corollary 6.3 *Parigot's $\lambda\mu$ calculus is sound and complete with respect to $\lambda_{\mathcal{C}}$ in the sense that the reductions of one calculus can be simulated by the other.*

7 Related Work

The relation between Parigot's $\lambda\mu$ and $\lambda_{\mathcal{C}}$ has been investigated by de Groote [7], who only considers the $\lambda\mu$ structural rule but not renaming and simplification. As for $\lambda_{\mathcal{C}}$, he only considers $\mathcal{C}_L$ and $\mathcal{C}_{\mathsf{tp}}$. However, these rules are not the original rules of Felleisen, since they do not contain abort. For example, $\mathcal{C}_{\mathsf{tp}}$ is $\mathcal{C}M \to \mathcal{C}(\lambda k.M(\lambda f.kf))$ which is in fact a reduction rule for $\lambda_{\mathcal{F}}$ [8]. This work fails in relating $\lambda\mu$ to $\lambda_{\mathcal{C}}$ in an untyped framework, since it does not express continuations as abortive functions. It says in fact that $\mathcal{F}$ behaves as $\mathcal{C}$ in the simply-typed case. Ong and Stewart [18] also do not consider the abort step in Felleisen's rules. This could be justified because in a simply-typed setting these steps are of type $\perp \to \perp$. Therefore, it seems we have a mismatch. While the aborts are essential in the reduction semantics, they are irrelevant in the corresponding proof. We are the first to provide a proof theoretic justification for those abort steps, they correspond to the step $\bot \to \bot\!\!\!\bot$. In addition to Ong and Stewart, Py [24] and Bierman [5] have pointed out the peculiarity of having an open $\lambda\mu$ term corresponding to a tautology. Their solution is to abolish the distinction between commands and terms. A command is a term returning $\bot$. The body of a μ-abstraction is not restricted to a command, but can be of the form $\mu\alpha.t$, where t is of type $\bot$. Thus, we have $\lambda y.\mu\alpha.(y\ \ \lambda x.[\alpha]x) : \neg\neg A \to A$. We would then represent the term $\mathcal{C}(\lambda k.(kI)x)$ (where I is $\lambda x.x$) as $\mu\alpha.([\alpha]I)x$. Whereas $\mathcal{C}(\lambda k.kIx)$ would reduce to $\mathcal{C}(\lambda k.kI)$ according to $\lambda_{n\mathcal{C}}$ and to I in $\lambda_{n\mu\mathsf{tp}}$, it would be in normal form in their calculus. Thus, their work in relating $\lambda\mu$ to λ_c only applies to typed λ_c, whereas our work also applies to the untyped case. Crolard [6] studied the relation between Parigot's $\lambda\mu$ and a calculus with a *catch* and *throw* mechanism. He showed that contraction corresponds to the *catch* operator ($\mu\alpha.[\alpha]t = catch\ \alpha\ t$) and weakening corresponds to the *throw* operator ($\mu\delta.[\alpha]t = throw\ \alpha\ t$ for δ not free in t). He only considers terms of the form $\mu\alpha.[\alpha]t$ and $\mu\beta.[\alpha]t$, where β does not occur free in t. This property is not preserved by the renaming rule, therefore reduction is restricted. We do not require such restrictions on reduction. We can simulate Ong and Stewart's $\lambda\mu$ and Crolard's calculus via this simple translation: $\mu\alpha.t$ becomes $\mu\alpha.[\mathsf{tp}]t$ and $[\beta]t$ becomes $\mu\delta.[\beta]t$, where δ is not free in t.

A categorical semantics of the call-by-name and call-by-value Parigot's $\lambda\mu$-calculus extended with disjunction has been provided by Selinger in [26].

8 Conclusions



Our analysis of the logical strengths of EFQ, PL (or EM), and DN has led naturally to a restricted form of classical logic called *minimal classical logic*. Depending on whether EFQ, PL, or both are assumed in minimal logic, we get intuitionistic, minimal classical, or classical logic. Depending on whether or not we admit $RAA_{\bot\!\!\!\bot}$ and $\bot_e$ in full classical natural deduction (on top of minimal natural deduction), we get the correspondences with the λ-calculi considered in this paper, as summarized above. The calculus λ_{tp} is the subset of $\lambda_{\mu\mathsf{tp}}$ in which expressions of the form $\mu\delta.[\alpha]t$ are only allowed when δ is not free in t and α is tp. The symbol $\mathrm{EFQ}^{\bot}_{\bot\!\!\!\bot}$ stands for $\bot_e$ and *Weakening*. Among these systems, $\lambda_{\mathcal{C}^-\mathsf{tp}}$ is a confluent extension of Felleisen's theory of control.

Acknowledgements

We thank Matthias Felleisen for numerous discussions about his theory of control.

References

[1] Ariola, Z. M., and Herbelin, H. Minimal classical logic and control operators. In *Thirtieth International Colloquium on Automata, Languages and Programming , ICALP'03, Eindhoven, The Netherlands, June 30 - July 4, 2003* (2003), vol. 2719, Springer-Verlag, LNCS, pp. 871–885.

[2] Ariola, Z. M., Herbelin, H., and Sabry, A. A type-theoretic foundation of continuations and prompts. In *ACM SIGPLAN International Conference on Functional Programming* (2004), ACM Press, New York, pp. 40–53.

[3] Barbanera, F., and Berardi, S. Extracting constructive content from classical logic via control-like reductions. In *Proceedings 1st Intl. Conf. on Typed Lambda Calculi and Applications, TLCA'93, Utrecht, The Netherlands, 16-18 March 1993*, M. Bezem and J. F. Groote, Eds., vol. 664. Springer-Verlag, Berlin, 1993, pp. 45–59.

[4] Barendregt, H. P. Lambda calculi with types. In *Handbook of Logic in Computer Science*, A. . G. . Maibaum, Ed., vol. 2. Oxford University Press, Inc., 1992, pp. 117–309.

[5] Bierman, G. A computational interpretation of the lambda-mu calculus. In *Mathematical foundations of computer science semantics, LNCS 1450* (1998), L. Brim, J. Gruska, and J. Zlatuska, Eds., Springer-Verlag, pp. 336–345.

[6] Crolard, T. A confluent lambda-calculus with a catch/throw mechanism. *Journal of Functional Programming 9(6)* (1999), 625–647.

[7] de Groote, P. On the relation between the lambda-mu calculus and the syntactic theory of sequential control. In *Logic Programming and Automated Reasoning, Proc. of the 5th International Conference, LPAR'94*, F. Pfennig, Ed. Springer, Berlin, Heidelberg, 1994, pp. 31–43.

[8] Felleisen, M. The theory and practice of first-class prompts. In *Proceedings of the 15th ACM Symposium on Principles of Programming Languages (POPL '88)* (Jan 1988), ACM Press, New York, pp. 180–190.

[9] Felleisen, M. On the expressive power of programming languages. In *ESOP '90 3rd European Symposium on Programming, Copenhagen, Denmark*, N. Jones, Ed., vol. 432. Springer-Verlag, New York, N.Y., 1990, pp. 134–151.

[10] Felleisen, M., and Hieb, R. The revised report on the syntactic theories of sequential control and state. *Theoretical Computer Science 103*, 2 (1992), 235–271.

[11] Gentzen, G. Investigations into logical deduction. In *Collected papers of Gerhard Gentzen*, M. Szabo, Ed. North-Holland, 1969, pp. 68–131.

[12] Girard, J.-Y. A new constructive logic: Classical logic. *Mathematical Structures in Computer Science 1*, 3 (1991), 255–296.

[13] Griffin, T. G. The formulae-as-types notion of control. In *Conf. Record 17th Annual ACM Symp. on Principles of Programming Languages, POPL'90, S an Francisco, CA, USA, 17-19 Jan 1990* (1990), ACM Press, New York, pp. 47–57.

[14] Hofmann, M. Sound and complete axiomatisations of call-by-balue control operators. *Mathematical Structures in Computer Science 5*, 4 (Dec. 1995), 461–482.

[15] Johansson, I. Der minimalkalkül, ein reduzierter intuitionistischer formalismus. *Compositio Math. 4* (1937), 119–136.

[16] Kameyama, Y., and Hasegawa, M. A sound and complete axiomatization of delimited continuations. In *Proc. of 8th ACM SIGPLAN Int. Conf. on Functional Programming, ICFP'03, Uppsala, Sweden, 25-29 Aug. 2003*, vol. 38(9) of *SIGPLAN Notices*. ACM Press, New York, 2003, pp. 177–188.

[17] Lalement, R. *Computation as Logic*. Prentice Hall International Series in Computer Science, Amsterdam, 1993.

[18] Ong, C.-H. L., and Stewart, C. A. A Curry-Howard foundation for functional computation with control. In *Conf. Record 24th ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages, POPL'97, Paris, France, 15-17 Jan. 1997*. ACM Press, New York, 1997, pp. 215–227.

[19] Parigot, M. Lambda-mu-calculus: An algorithmic interpretation of classical natural deduction. In *Logic Programming and Automated Reasoning: International Conference LPAR '92 Proceedings, St. Petersburg, Russia* (1992), Springer-Verlag, pp. 190–201.

[20] Parigot, M. Classical proofs as programs. *Computational logic and theory 713* (1993), 263–276.

[21] Parigot, M. Strong normalization for second order classical natural deduction. In *Proceedings 8th Annual IEEE Symp. on Logic in Computer Science, LICS'93*. IEEE Computer Society Press, 1993, pp. 39–47.

[22] Parigot, M. Proofs of strong normalisation for second order classical natural deduction. *Journal of Symbolic Logic 62*, 4 (1997), 1461–1479.

[23] Prawitz, D. *Natural Deduction, a Proof-Theoretical Study.* Almquist and Wiksell, Stockholm, 1965.

[24] Py, W. *Confluence en $\lambda\mu$-calcul.* PhD thesis, Université de Savoie, 1998.

[25] Sabry, A., and Felleisen, M. Reasoning about programs in continuation-passing style. *Lisp Symb. Comput. 6*, 3-4 (1993), 289–360.

[26] Selinger, P. Control categories and duality: on the categorical semantics of the lambda-mu calculus. *Mathematical. Structures in Comp. Sci. 11*, 2 (2001), 207–260.

[27] Streicher, T., and Reus, B. Classical logic: Continuation semantics and abstract machines. *Journal of Functional Programming 8(6)* (1998), 543–572.