Denotations for classical proofs
– Preliminary results –

Philippe de Groote
Université Catholique de Louvain, Unité d'Informatique
BE-1348 Louvain-la-Neuve, BELGIUM

Abstract

This paper addresses the problem of extending the formulae-as-types principle to classical logic. More precisely, we introduce a typed lambda-calculus (λ-LK$^{\rightarrow}$) whose inhabited types are exactly the implicative tautologies of classical logic and whose type assignment system is a classical sequent calculus. Intuitively, the terms of λ-LK$^{\rightarrow}$ correspond to constructs that are highly non-deterministic. This intuition is made much more precise by providing a simple model where the terms of λ-LK$^{\rightarrow}$ are interpreted as non-empty sets of (interpretations of) untyped lambda-terms. We also consider the system (λ-LK$^{\rightarrow}$ + *cut*) and investigate the relation existing between cut elimination and reduction. Finally, we show how to extend our system in order to take conjunction, disjunction and negation into account.

1 Introduction

In this paper, we investigate the possibility of designing a calculus for denoting proofs of classical logic based on the formulae-as-types principle [15]. In other words, we try to define a typed λ-calculus whose types are classical propositions and whose terms denote classical proofs. This may seem hopeless, because it is known that the technical content of the formulae-as-types principle, namely the Curry-Howard isomorphism [6, 10, 15, 23], is strongly related to the constructive aspect of intuitionistic logic. Indeed, any straightforward adaptation of the isomorphism to the case of classical logic yields a degenerate model where all the proofs of a given proposition are identified [17]. This can be stated by saying that the only denotational semantics of classical logic is trivial [10].

We do not think that this last technical statement gives a final answer to the problem. To tell if a classical proposition (i.e. a proposition classically provable) is or is not intuitionistic, one has to consider its possible proofs. Given a proposition and its proof, it is not the proposition itself that is intuitionistic or non-intuitionistic, but rather its proof that is constructive or not. Of course, there are classically valid propositions for which there does not exist any constructive proof, but one must not forget, on the other hand, that there exist non-constructive proofs of intuitionistically valid propositions. Therefore, it makes sense, at least from a syntactic point of view, to consider that the set of proofs of classical logic is a proper extension of the set of proofs of intuitionistic logic.

At this point, it is worth mentioning that our primary motivation, when starting this work, was chiefly pragmatic. Typed λ-calculus, as a mere notation for intuitionistic proofs, is useful in practice. In the case of interactive theorem proving, it allows proofs to be turned into objects that can be manipulated in different ways. When we implemented one of the earliest versions

In: *Logical Foundations of Computer Science – Tver'92,*
A. Nerode, M. Taitslin (Eds.),
Lecture Notes in Computer Science, Vol. 620, Springer-Verlag (1992), pp. 105-116.

of classical predicate calculus in Isabelle [21], we found that some notation to handle classical proofs would also be useful. Hence it was natural to try to extend the syntax of λ-terms in order to capture the non-constructive proofs of classical logic.

In fact, such an extension may be achieved very cheaply by considering a λ-calculus with given constants of type $\neg\alpha \vee \alpha$. In some sense, this is how Gentzen extends NJ (the intuitionistic system of natural deduction) into NK (the classical one) [8]. This possible solution is not so fruitful, however, for it does not enlighten us about the very nature of classical proofs. Natural deduction systems do not really fit classical logic.

The deep difference between classical and intuitionistic logic, that is the non-constructive aspect of classical proofs, is much more apparent when one considers sequent calculus. In both Gentzen's systems LJ and LK [8], the logical rules controlling the left- or right-introduction of connectives are instances of the same schemes. The only difference between intuitionistic and classical proofs is structural. The succedent of any intuitionistic sequent must consist of at most one formula. On the other hand, the succedent of a classical sequent may consist of several formulas (interpreted disjunctively). This yields, in the latter case, a kind of non-determinism that corresponds to the non-constructive aspect of classical logic.

The above discussion leads us to the conclusions that follow. The real challenge in designing a *classical λ-calculus* is to design a typed language whose type assignment system consists of a classical sequent calculus, i.e. a system whose sequents may be manifold concluded. Moreover, such a language should come together with a decent interpretation that would help us to understand the non-determinism related to classical logic. This is exactly the problem we want to tackle.

The remainder of this paper is organized as follows.

In Section 2, we study the implicative fragment of Gentzen's LK. In particular, we try to give a computational meaning to its rules. This results in the concrete proposal of a λ-calculus (λ-LK$^{\rightarrow}$) whose inhabited types are exactly the implicative tautologies of classical logic.

Intuitively, the terms of λ-LK$^{\rightarrow}$ correspond to constructs that are highly non-deterministic. In Section 3, this intuition is made much more precise by interpreting the terms of λ-LK$^{\rightarrow}$ as non-empty sets of (interpretations of) untyped λ-terms. The type assignment system of Section 2 is showed to be sound for this simple semantics. However, it is far from being complete. This yields a discussion about completeness.

The system defined in Section 2 is cut-free and its typable terms are all in normal form. In Section 4, we consider the system (λ-LK$^{\rightarrow}$ + *cut*) and we investigate the relation existing between cut elimination and reduction.

So far, only the implicative fragment of classical logic has been considered. In Section 5, we show how to extend our system in order to take conjunction, disjunction, and negation into account.

We have subtitled this paper *preliminary results*. We do not consider λ-LK as a definitive system but as an experimental one. This paper proposes some solutions but it also raises many problems. We summarize these problems in Section 6 and conclude by presenting our plans for future work.

2 The positive implicative fragment

The problem that we want to tackle is to give a computational interpretation to the sequents of Gentzen's LK. This problem is already non-trivial in the case of the implicative fragment of

LK. Indeed, as it is well-known, the positive implicative fragment of classical logic is a proper extension of the corresponding fragment of intuitionistic logic. This can be demonstrated by giving a classical implicative tautology that is not intuitionistically provable. The paradigmatic example of such a tautology is Peirce's law:

$$((\alpha \to \beta) \to \alpha) \to \alpha \qquad \text{(Peirce's law)}$$

Let us, for the time being, restrict ourself to the implicative fragment of LK.

Classical sequents have the form $\Gamma \vdash \Theta$ where the succedent Θ may consist of more than one formula. This is where the difficulty is. We must, therefore, focus on the rules acting directly on the succedent: the right weakening rule, the right contraction rule and the right introduction rule for implication ($\to$). Let us consider small examples of derivations involving these rules.

A first typical example is the following:

$$\dfrac{\dfrac{\alpha \vdash \alpha}{\alpha \vdash \alpha,\, \beta}}{\vdash \alpha,\, \alpha \to \beta}$$

How can we decorate the formulas involved in this derivation with terms standing for proofs? The first sequent, which is an axiom, corresponds simply to the declaration (and the use) of a variable ($x : \alpha \vdash x : \alpha$). Now, what can we say of the second sequent? What proof can we assign to the formula β? The second sequent is clearly valid because if we are given a proof of α we certainly have a proof of α or a proof of β. However, we do not know anything about β in terms of a possible proof. This can be stressed out by assigning some dummy term to β ($x : \alpha \vdash x : \alpha, \perp : \beta$). The third sequent is obtained by ($\to$)-introduction. This rule, in the intuitionistic case, corresponds to the formation of a λ-abstraction. Therefore we assign the term $\lambda x{:}\alpha.\perp$ to the formula $\alpha \to \beta$. Then the problem becomes what term can we assign to α after having discarded the declaration $x : \alpha$? If α (seen as a type) is empty, $\lambda x{:}\alpha.\perp$ may be interpreted as the empty function, which in this case is a proof of $\alpha \to \beta$. Hence, when α is empty, the sequent is valid and we may assign a dummy proof to α. If α is not empty, then any of its elements may be used as a proof. Therefore we introduce a choice operator $\epsilon(-)$ that, given some non-empty type, picks out one of its elements. Thus the all derivation becomes the following:

$$\dfrac{\dfrac{x : \alpha \vdash x : \alpha}{x : \alpha \vdash x : \alpha,\, \perp : \beta}}{\vdash \epsilon(\alpha) : \alpha,\, \lambda x{:}\alpha.\perp : \alpha \to \beta}$$

Now consider the derivation which follows (it starts as the previous one where α and β have been identified):

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \vdash x : \alpha}{x : \alpha \vdash x : \alpha,\, \perp : \alpha}}{\vdash \epsilon(\alpha) : \alpha,\, \lambda x{:}\alpha.\perp : \alpha \to \alpha}}{y : \alpha \vdash \epsilon(\alpha) : \alpha,\, \lambda x{:}\alpha.\perp : \alpha \to \alpha}}{\vdash \lambda y{:}\alpha.\,\epsilon(\alpha) : \alpha \to \alpha,\, \lambda x{:}\alpha.\perp : \alpha \to \alpha}$$

The informal interpretation that we can give to the last sequent is typical of classical reasoning. We have to show that the functional type $\alpha \to \alpha$ is not empty. We proceed as follows: if

α is non-empty, there exists some constant function $(\lambda y : \alpha.\, \epsilon(\alpha))$ assigning to each element of α one distinct element of α; if α is empty, then the empty function $(\lambda x : \alpha.\, \bot)$ belongs to $\alpha \to \alpha$. This interpretation illustrates the non determinism of classical disjunction: we have two possible proofs of $\alpha \to \alpha$; we know that one of these is an actual proof; but we do not (cannot) know which one is the actual proof. Therefore, when applying a contraction rule, we must internalize, at the level of terms, the non-determinism existing at the level of sequents. We introduce a binary choice operator $(- \mathbin{[\!]} -)$ to this end. Then, by contraction, we may end the above derivation as follows:

$$\frac{\vdots}{\vdash\ \lambda y\!:\!\alpha.\, \epsilon(\alpha) \mathbin{[\!]} \lambda x\!:\!\alpha.\, \bot : \alpha \to \alpha}$$

Before defining formally the system λ-LK$^{\to}$, let us summarize. We have introduced three new constructs: a special constant $(\bot)$ and two non-deterministic choice operators $(- \mathbin{[\!]} -$ and $\epsilon(-))$. The intuitive meaning of these constructs is the following:

- $\bot$ is a fictitious proof; in some sense, it stands for something that does not exist;
- $- \mathbin{[\!]} -$ is a binary choice operator whose non-determinism is angelic; the value of $M \mathbin{[\!]} N$ is the value of M or the value of N but cannot be fictitious unless the values of both M and N are fictitious;
- $\epsilon(-)$ is a choice operator akin to Hilbert's ϵ [18]; if α is a non-empty type, $\epsilon(\alpha)$ stands for some element of α; if α is empty, $\epsilon(\alpha)$ is a fictitious term.

Formally, we define the system λ-LK$^{\to}$ as follows.

Let $\mathcal{A}$ be a countably infinite set of type-variables. The set $\mathcal{T}$ of types of λ-LK$^{\to}$ is inductively defined as follows:

(i) if $\mathrm{a} \in \mathcal{A}$ then $\mathrm{a} \in \mathcal{T}$;

(ii) if $\alpha, \beta \in \mathcal{T}$ then $(\alpha \to \beta) \in \mathcal{T}$.

Let $\mathcal{X}$ be a countably infinite set of term-variables. The set Λ_{LK} of terms of λ-LK$^{\to}$ is inductively defined as follows:

(i) if $x \in \mathcal{X}$ then $x \in \Lambda_{\mathrm{LK}}$;

(ii) $\bot \in \Lambda_{\mathrm{LK}}$;

(iii) if $\alpha \in \mathcal{T}$ then $\epsilon(\alpha) \in \Lambda_{\mathrm{LK}}$;

(iv) if $M, N \in \Lambda_{\mathrm{LK}}$ then $(M \mathbin{[\!]} N) \in \Lambda_{\mathrm{LK}}$;

(v) if $M, N \in \Lambda_{\mathrm{LK}}$ then $(M\, N) \in \Lambda_{\mathrm{LK}}$;

(vi) if $x \in \mathcal{X}$, $\alpha \in \mathcal{T}$ and $M \in \Lambda_{\mathrm{LK}}$ then $(\lambda x\!:\!\alpha.\, M) \in \Lambda_{\mathrm{LK}}$.

We omit parentheses according to the usual conventions. We use $\alpha, \beta, \gamma, \ldots$ to denote types and $M, N, O, \ldots$ to denote terms. The expression $M[x{:=}N]$ denotes the result of substituting N for the free occurrences of x in M. Expressions of the form $M : \alpha$ are called statements and we use $\Gamma, \Delta, \Theta, \ldots$ to denote sequences of statements. If Θ is the sequence $(M_1 : \alpha_1, \ldots, M_n : \alpha_n)$ then $\Theta[x{:=}N]$ is the sequence $(M_1[x{:=}N] : \alpha_1, \ldots, M_n[x{:=}N] : \alpha_n)$.

The type assignment system of λ-LK$^{\rightarrow}$ consists of the following rules:

axiom:

$$x : \alpha \vdash x : \alpha \qquad \text{(identity)}$$

structural rules:

$$\frac{\Gamma,\, x : \alpha,\, y : \beta,\, \Delta \vdash \Theta}{\Gamma,\, y : \beta,\, x : \alpha,\, \Delta \vdash \Theta} \qquad \text{(exchange – left)}$$

$$\frac{\Gamma \vdash \Theta,\, M : \alpha,\, N : \beta,\, \Xi}{\Gamma \vdash \Theta,\, N : \beta,\, M : \alpha,\, \Xi} \qquad \text{(exchange – right)}$$

$$\frac{\Gamma \vdash \Theta}{x : \alpha,\, \Gamma \vdash \Theta} \qquad \text{(weakening – left)}$$

$$\frac{\Gamma \vdash \Theta}{\Gamma \vdash \Theta,\, \bot : \alpha} \qquad \text{(weakening – right)}$$

$$\frac{x : \alpha,\, y : \alpha,\, \Gamma \vdash \Theta}{x : \alpha,\, \Gamma \vdash \Theta[y{:=}x]} \qquad \text{(contraction – left)}$$

$$\frac{\Gamma \vdash \Theta,\, M : \alpha,\, N : \alpha}{\Gamma \vdash \Theta,\, M \,[\!]\, N : \alpha} \qquad \text{(contraction – right)}$$

logical rules:

$$\frac{\Gamma \vdash \Theta,\, M : \alpha \qquad y : \beta,\, \Delta \vdash \Xi}{x : \alpha \rightarrow \beta,\, \Gamma,\, \Delta \vdash \Theta,\, \Xi[y{:=}(x\, M)]} \qquad \text{(implication – left)}$$

$$\frac{x : \alpha,\, \Gamma \vdash \Theta,\, M : \beta}{\Gamma \vdash \Theta[x{:=}\epsilon(\alpha)],\, \lambda x{:}\alpha.\, M : \alpha \rightarrow \beta} \qquad \text{(implication – right)}$$

In Rule *weakening-left* and Rule *implication-left*, the variable x must be fresh.

If one forgets the terms and considers only the types, the above system corresponds exactly to the implicative fragment of Gentzen's LK such as defined in [8]. As an example, let us give a derivation of Peirce's law:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{x : \alpha \vdash x : \alpha}{x : \alpha \vdash x : \alpha,\, \bot : \beta}}{\vdash \epsilon(\alpha) : \alpha,\, \lambda x{:}\alpha.\bot : \alpha \rightarrow \beta} \qquad y : \alpha \vdash y : \alpha}{z : (\alpha \rightarrow \beta) \rightarrow \alpha \vdash \epsilon(\alpha) : \alpha,\, z\,(\lambda x{:}\alpha.\bot) : \alpha}}{z : (\alpha \rightarrow \beta) \rightarrow \alpha \vdash (\epsilon(\alpha) \,[\!]\, z\,(\lambda x{:}\alpha.\bot)) : \alpha}}{\vdash \lambda z{:}(\alpha \rightarrow \beta) \rightarrow \alpha.\, (\epsilon(\alpha) \,[\!]\, z\,(\lambda x{:}\alpha.\bot)) : ((\alpha \rightarrow \beta) \rightarrow \alpha) \rightarrow \alpha}$$

The term $\lambda z{:}(\alpha \rightarrow \beta) \rightarrow \alpha.\, (\epsilon(\alpha) \,[\!]\, z\,(\lambda x{:}\alpha.\bot))$, which corresponds to the proof of Peirce's law, may be interpreted as follows. Let z be a function from $(\alpha \rightarrow \beta)$ to α. If α is non-empty, one of its elements ($\epsilon(\alpha)$) is produced. Otherwise, when α is empty, there exists a function from α to any β, namely the empty function ($\lambda x : \alpha.\bot$). Then an element of α can be obtained by applying z to the empty function.

3 Simple semantics

The constructs that we have introduced in the previous section have been given an intuitive meaning. Is it possible to turn this intuition into a technical interpretation? To answer this question, we provide a model of λ-LK$^{\rightarrow}$.

To built this model, we adapt the simple semantics of type assignment [2, 5, 4, 13] to our non-deterministic constructs. Let $\langle D, \Phi, \Psi \rangle$ be an environment model of the untyped λ-calculus [19], where $\Phi \in D \longrightarrow [D \rightarrow D]$ is the projection and $\Psi \in [D \rightarrow D] \longrightarrow D$ is the embedding.

A type-environment is a map assigning to each type-variable a subset of D. We write $\mathcal{T}$-**env** for the set of of type-environments (i.e. the set $\mathcal{A} \longrightarrow \mathrm{P}(D)$) and we let ρ range over $\mathcal{T}$-**env**.

The interpretation of types $\mathcal{T}[\![-]\!]_{\_} \in (\mathcal{T} \times \mathcal{T}\text{-}\mathbf{env}) \longrightarrow \mathrm{P}(D)$ is defined as usual:

(i) $\mathcal{T}[\![\mathrm{a}]\!]_{\rho} = \rho(\mathrm{a})$

(ii) $\mathcal{T}[\![\alpha \rightarrow \beta]\!]_{\rho} = \{d \in D \mid \forall e \in \mathcal{T}[\![\alpha]\!]_{\rho}.\, \Phi(d)(e) \in \mathcal{T}[\![\beta]\!]_{\rho}\}$

To model the non-determinism of λ-LK$^{\rightarrow}$, we follow the usual idea that consists to interpret non-deterministic expressions as sets of values. More precisely, we interprets the terms of λ-LK$^{\rightarrow}$ as sets of functions mapping term-environments to elements of D. A term-environment is a map assigning to each type-variable an element of D. The set of term-environments (i.e. $\mathcal{X} \longrightarrow D$) is written $\mathcal{E}$-**env** and we use η as a metavariable to denote term-environments.

The interpretation of terms $\mathcal{E}[\![-]\!]_{\_} \in (\Lambda_{\mathrm{LK}} \times \mathcal{T}\text{-}\mathbf{env}) \longrightarrow \mathrm{P}(\mathcal{E}\text{-}\mathbf{env} \longrightarrow D)$ is defined according to the following rules:

(i) $\mathcal{E}[\![x]\!]_{\rho} = \{\eta \mapsto \eta(x)\}$

(ii) $\mathcal{E}[\![\bot]\!]_{\rho} = \emptyset$

(iii) $\mathcal{E}[\![\epsilon(\alpha)]\!]_{\rho} = \{\eta \mapsto d \mid d \in \mathcal{T}[\![\alpha]\!]_{\rho}\}$

(iv) $\mathcal{E}[\![M \,[\!]\, N]\!]_{\rho} = \mathcal{E}[\![M]\!]_{\rho} \cup \mathcal{E}[\![N]\!]_{\rho}$

(v) $\mathcal{E}[\![M\,N]\!]_{\rho} = \{\eta \mapsto \Phi(d(\eta))(e(\eta)) \mid d \in \mathcal{E}[\![M]\!]_{\rho} \wedge e \in \mathcal{E}[\![N]\!]_{\rho}\}$

(vi) $\mathcal{E}[\![\lambda x\!:\!\alpha.\, M]\!]_{\rho} = \{\eta \mapsto \Psi(d \mapsto e(\eta[d/x])) \mid \mathcal{T}[\![\alpha]\!]_{\rho} \neq \emptyset \supset e \in \mathcal{E}[\![M]\!]_{\rho}\}$

Given $\rho \in \mathcal{T}$-**env**, $\eta \in \mathcal{E}$-**env**, and $M \in \Lambda_{\mathrm{LK}}$, we also define:

- $\mathcal{E}[\![M]\!]_{\rho,\eta} = \{d(\eta) \mid d \in \mathcal{E}[\![M]\!]_{\rho}\}$

The above simple semantics corresponds fairly to the intuitive interpretation given in Section 2. The fact that this intuitive interpretation makes sense is expressed by a soundness property.

Let ρ and η be respectively a type- and a term-environment. We say that the pair ρ, η satisfies the statement $M : \alpha$, and we write $\rho, \eta \models M : \alpha$, if and only if

(a) $\mathcal{E}[\![M]\!]_{\rho,\eta} \neq \emptyset$,

(b) $\mathcal{E}[\![M]\!]_{\rho,\eta} \subset \mathcal{T}[\![\alpha]\!]_{\rho}$.

Let $\Gamma \equiv (x_1 : \alpha_1, \ldots, x_n : \alpha_n)$ be a sequence of declarations and $\Theta \equiv (M_1 : \beta_1, \ldots, M_m : \beta_m)$ be a sequence of statements. We say that the succedent Θ is a semantic consequence of the antecedent Γ, and we write $\Gamma \models \Theta$, if and only if, for all pairs ρ, η such that

- $\rho, \eta \models x_i : \alpha_i$ for all $i \in \{1, \ldots, n\}$,

there exists $j \in \{1, \ldots, m\}$ such that

- $\rho, \eta \models M_j : \beta_j$.

Proposition 3.1 (*Soundness) If* $\Gamma \vdash \Theta$ *then* $\Gamma \models \Theta$.

The proof proceeds by induction on the derivation of $\Gamma \vdash \Theta$.

It is natural to ask if the converse property holds, i.e. if the type assignment system of Section 2 is complete for the above semantics. The answer is no.

In the intuitionistic case, to get a complete type assignment system it is sufficient to add the following conversion rule [2, 5, 13]:

$$\frac{\Gamma \vdash M : \alpha \qquad M = N}{\Gamma \vdash N : \alpha} \qquad \text{(conversion)}$$

where $=$ stands for β-conversion when the semantic model is not extensional, and for $\beta\eta$-conversion when it is.

In our case, β- or $\beta\eta$-conversion is not enough. We also need some conversion theory for the new constructs. For instance, our simple semantics suggests the following laws:

$$(M \,[\!]\, N) \,[\!]\, O = M \,[\!]\, (N \,[\!]\, O) \qquad M \,[\!]\, N = N \,[\!]\, M$$

$$M \,[\!]\, M = M \qquad M \,[\!]\, \bot = M \qquad M \,[\!]\, \epsilon(\alpha) = \epsilon(\alpha)$$

To add these laws, however, is not sufficient. In fact, the problem is deeper. Before designing any appropriate conversion theory for λ-LK$^{\rightarrow}$, two questions must be answered: do we really want completeness? in the affirmative, with respect to what class of models do we want to be complete?

To answer the first question is not obvious. For any possible model and any provable proposition α, we may expect to have $\models \epsilon(\alpha) : \alpha$. Hence, for an undecidable logic, a complete typing-system would yield an undecidable typing-relation.

For the second question, our present feeling is that the simple semantics defined in this section is too simple.

4 Reduction and cut elimination

All the terms typable according to the system of Section 2 are in β-normal form. Indeed, the substitution in Rule *implication-left* cannot create any β-redex, for the substituted term is of the form $(x\, M)$.

To allow non-normal terms to be typable, we must provide λ-LK$^{\rightarrow}$ with a cut rule:

$$\frac{\Gamma \vdash \Theta, M : \alpha \qquad x : \alpha, \Delta \vdash \Xi}{\Gamma, \Delta \vdash \Theta, \Xi[x{:=}M]} \qquad \text{(cut)}$$

Then a proof-theoretic question arises: to what corresponds cut elimination in λ-LK$^{\to}$ + *cut*?

In the context of natural deduction, Prawitz has introduced the notion of normal proof [22]. In the intuitionistic setting, there is a homomorphism between cut elimination in LJ and proof normalization in NJ, which is isomorphic to β-normalization.

In our case, there is a mismatch between cut elimination, which acts at the global level of the sequents, and β-reduction, which acts at the local level of the terms. This can be shown by considering the main step of cut elimination:

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\vdots\\ \Pi_1\\ \vdots\end{matrix}}{x:\alpha,\,\Gamma\vdash\Theta,\,M:\beta}}{\Gamma\vdash\Theta[x{:=}\epsilon(\alpha)],\,\lambda x{:}\alpha.\,M:\alpha\to\beta}\qquad\dfrac{\dfrac{\begin{matrix}\vdots\\ \Pi_2\\ \vdots\end{matrix}}{\Delta_1\vdash\Xi_1,\,N:\alpha}\quad\dfrac{\begin{matrix}\vdots\\ \Pi_3\\ \vdots\end{matrix}}{y:\beta,\,\Delta_2\vdash\Xi_2}}{z:\alpha\to\beta,\,\Delta_1,\,\Delta_2\vdash\Xi_1,\,\Xi_2[y{:=}(z\,N)]}}{\Gamma,\,\Delta_1,\,\Delta_2\vdash\Theta[x{:=}\epsilon(\alpha)],\,\Xi_1,\,\Xi_2[y{:=}(\lambda x{:}\alpha.\,M)\,N]}\;CUT$$

reduces to:

$$\dfrac{\dfrac{\dfrac{\dfrac{\begin{matrix}\vdots\\ \Pi_2\\ \vdots\end{matrix}}{\Delta_1\vdash\Xi_1,\,N:\alpha}\quad\dfrac{\begin{matrix}\vdots\\ \Pi_1\\ \vdots\end{matrix}}{x:\alpha,\,\Gamma\vdash\Theta,\,M:\beta}}{\Delta_1,\,\Gamma\vdash\Xi_1,\,\Theta[x{:=}N],\,M[x{:=}N]:\beta}\;CUT\qquad\dfrac{\begin{matrix}\vdots\\ \Pi_3\\ \vdots\end{matrix}}{y:\beta,\,\Delta_2\vdash\Xi_2}}{\Delta_1,\,\Gamma,\,\Delta_2\vdash\Xi_1,\,\Theta[x{:=}N],\,\Xi_2[y{:=}M[x{:=}N]]}\;CUT}{\Gamma,\,\Delta_1,\,\Delta_2\vdash\Theta[x{:=}N],\,\Xi_1,\,\Xi_2[y{:=}M[x{:=}N]]}$$

At the semantic level, we get an interesting interpretation: cut elimination corresponds to a gain of determinism. At the syntactic level, however, cut elimination does not correspond to β-reduction. On the one hand, the sequent obtain by cut elimination is the following one:

$$\Gamma,\,\Delta_1,\,\Delta_2\vdash\Theta[x{:=}N],\,\Xi_1,\,\Xi_2[y{:=}M[x{:=}N]]$$

On the other hand, the process of β-reduction yields the following result:

$$\Gamma,\,\Delta_1,\,\Delta_2\vdash\Theta[x{:=}\epsilon(\alpha)],\,\Xi_1,\,\Xi_2[y{:=}M[x{:=}N]]$$

This observation raises a new question: which reduction theory must we consider for the terms of our calculus?

5 Adding other connectives

λ-LK$^{\to}$ can be easily extended in order to deal with negation, conjunction, and disjunction. This is achieved by considering one given empty type and by introducing products and sums.

For negation, we consider one type constant $\mathbf{0}$, we define $\neg\alpha$ as $\alpha\to\mathbf{0}$, and we add the following axiom:

$$x:\mathbf{0}\vdash x:\alpha \qquad \text{(e falso sequitur quod libet)}$$

For conjunction, we extend the formation rules by allowing a term to be a pair $\langle M, N\rangle$, to be a left projection $\mathbf{fst}(M)$, or to be a right projection $\mathbf{snd}(M)$. Then we introduce the following rules:

$$\frac{x : \alpha,\, \Gamma \vdash \Theta}{y : \alpha \wedge \beta,\, \Gamma \vdash \Theta[x{:=}\mathbf{fst}(y)]} \qquad \text{(conjunction – left – 1)}$$

$$\frac{x : \beta,\, \Gamma \vdash \Theta}{y : \alpha \wedge \beta,\, \Gamma \vdash \Theta[x{:=}\mathbf{snd}(y)]} \qquad \text{(conjunction – left – 2)}$$

$$\frac{\Gamma \vdash \Theta,\, M : \alpha \qquad \Delta \vdash \Xi,\, N : \beta}{\Gamma,\, \Delta \vdash \Theta,\, \Xi,\, \langle M, N\rangle : \alpha \wedge \beta} \qquad \text{(conjunction – right)}$$

For disjunction, we use sums. Let $\mathbf{in}_l$ and $\mathbf{in}_r$ be the two injection operators. The right introduction rules are then the following:

$$\frac{\Gamma \vdash \Theta,\, M : \alpha}{\Gamma \vdash \Theta,\, \mathbf{inj}_l(M) : \alpha \vee \beta} \qquad \text{(disjunction – right – 1)}$$

$$\frac{\Gamma \vdash \Theta,\, M : \beta}{\Gamma \vdash \Theta,\, \mathbf{inj}_r(M) : \alpha \vee \beta} \qquad \text{(disjunction – right – 2)}$$

For the left-introduction rule, we propose two alternative forms. Gentzen's original rule is the following one:

$$\frac{\alpha,\, \Gamma \vdash \Theta \qquad \beta,\, \Gamma \vdash \Theta}{\alpha \vee \beta,\, \Gamma \vdash \Theta} \qquad \text{(OEA)}$$

This rule may be decorated with terms as follows:

$$\frac{x : \alpha,\, \Gamma \vdash (M_i : \theta_i)_{i \in n} \qquad y : \beta,\, \Gamma \vdash (N_i : \theta_i)_{i \in n}}{z : \alpha \vee \beta,\, \Gamma \vdash (\mathbf{D}xy.(z, M_i, N_i) : \theta_i)_{i \in n}}$$

where the binding operator $\mathbf{D}$ is such that the free occurrences of x in M and the free occurrences of y in N are bound in $\mathbf{D}xy.(L, M, N)$. Moreover, this operator obey the following laws:

$$\mathbf{D}xy.(\mathbf{in}_l(L), M, N) = M[x{:=}L] \qquad \mathbf{D}xy.(\mathbf{in}_r(L), M, N) = N[y{:=}L]$$

The above left-introduction rule, which is a straightforward adaptation of the intuitionistic rule, appears to be peculiar when compared to the other rules. The contexts in the antecedents (Γ) and the types in the succedents (θ_i) are required to be the same in both the premises. This was not the case in all the others rules. Therefore, we propose the following alternative:

$$\frac{x : \alpha,\, \Gamma \vdash \Theta \qquad y : \beta,\, \Delta \vdash \Xi}{z : \alpha \vee \beta,\, \Gamma,\, \Delta \vdash \Theta[x{:=}\mathbf{out}_l(z)],\, \Xi[y{:=}\mathbf{out}_r(z)]} \qquad \text{(disjunction – left)}$$

where the operators $\mathbf{out}_l$ and $\mathbf{out}_r$ are akin to the sum destructors used in Edimburgh LCF [11] and obey the following laws:

$$\mathbf{out}_l(\mathbf{in}_l(M)) = M \qquad \mathbf{out}_l(\mathbf{in}_r(M)) = \bot \qquad \mathbf{out}_r(\mathbf{in}_r(M)) = M \qquad \mathbf{out}_r(\mathbf{in}_l(M)) = \bot$$

6 Conclusions and future work

The results that have been reported in this paper show that the design of *classical* λ*-calculi* is feasible and worthy. As we pointed out in the introduction, a system such as λ-LK may be useful in practice. It can be used for interactive theorem proving and also provides a possible formalism to study the computational content of classical proofs, which is a problem addressed in [3]. At a more fundamental level, to design systems such as λ-LK and to provide semantics for them may give us a better understanding of the nature of classical proofs. Related works on this topic include [3, 9, 12, 20].

Nevertheless, λ-LK is still an experimental system and problems remain. The two main ones concern the definition of suitable notions of conversion and reduction.

The problem of defining an appropriate conversion theory is related to the issue of completeness with respect to some class of models. For this purpose, further semantic investigations are needed. Models used in relational programming are good candidates for such investigations. It is not clear, however, that completeness is a desirable property. This illustrates a general principle. To be classical, we have to pay a price; we must drop some of the properties that hold in the intuitionistic context.

The notion of reduction is related to the notion of cut elimination and to the computational content of classical proofs. Here also properties that hold in the intuitionistic case are no more compatible in the classical one. For instance, conversion cannot be defined as the symmetric, transitive closure of reduction if the latter corresponds to cut elimination. Indeed, it would amount to equate all the proofs because of the non-confluence of cut elimination. Therefore, to define an appropriate reduction theory requires further proof-theoretic investigations. Interesting results may be obtained when considering interpretations of classical logic into intuitionistic systems [3, 12, 20]. The intuitive meaning given to the operator $\epsilon(-)$ suggests also possible connections with the notions of oracle and relative recursiveness.

References

[1] H.P. Barendregt. *The lambda calculus, its syntax and semantics.* North-Holland, revised edition, 1984.

[2] F. Cardone and M. Coppo. Two extensions of Curry's type inference system. In P. Odifreddi, editor, *Logic and Computer Science*, pages 19–75. Academic Press, 1990.

[3] R. Constable and C. Murthy. Finding computational content in classical proofs. In G. Huet and G. Plotkin, editors, *Logical Frameworks*, pages 341–362. Cambridge University Press, 1991.

[4] M. Coppo. On the semantics of polymorphism. *Acta Informatica*, 20:159–170, 1983.

[5] M. Coppo. Completeness of type assignment in continuous λ models. *Theoretical Computer Science*, 29:309–324, 1984.

[6] H.B. Curry and R. Feys. *Combinatory Logic, Vol. I.* North-Holland, 1958.

[7] J.H. Gallier. *Logic for Computer Science.* John Wiley & Sons, 1988.

[8] G. Gentzen. *Recherches sur la déduction logique (Untersuchungen über das logische schliessen).* Presses Universitaires de France, 1955. Traduction et commentaire par R. Feys et J. Ladrière.

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

[10] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, 1989.

[11] M. Gordon, R. Milner, and C. Wadsworth. *Edinburgh LCF: A mechanized Logic of Computation*, volume 78 of *Lecture Notes in Computer Science.* Springer-Verlag, 1979.

[12] T. G. Griffin. A formulae-as-types notion of control. In *Conference record of the seventeenth annual ACM symposium on Principles of Programming Languages*, pages 47–58, 1990.

[13] J.R. Hindley. The completeness theorem for typing λ-terms. *Theoretical Computer Science*, 22:1–17, 1983.

[14] J.R. Hindley and J.P. Seldin. *Introduction to combinators and λ-calculus.* London Mathematical Society Student Texts. Cambridge University Press, 1986.

[15] W.A. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *to H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, 1980.

[16] J.-L. Krivine. *Lambda-calcul, types et modèles.* Masson, 1990.

[17] J. Lambek and P.J. Scott. *An introduction to higher order categorical logic.* Cambridge University Press, 1986.

[18] A.C. Leisenring. *Mathematical Logic and Hilbert's ϵ-Symbol.* Gordon and Breach Science Publishers, New-York, 1969.

[19] A. R. Meyer. What is a model of the lambda calculus. *Information and Control*, 52:87–122, 1982.

[20] C. R. Murthy. An evaluation semantics for classical proofs. In *Proceedings of the sixth annual IEEE symposium on Logic In Computer Science*, pages 96–107, 1991.

[21] L.C. Paulson. Isabelle: The next 700 theorem provers. In P. Odifreddi, editor, *Logic and Computer Science*, pages 361–386. Academic Press, 1990.

[22] D. Prawitz. *Natural deduction, A Proof-Theretical Study.* Almqvist & Wiksell, Stockholm, 1965.

[23] S. Stenlund. *Combinators λ-terms and proof theory.* D. Reidel Publishing Company, 1972.