Adding Negation to Lambda Mu

Steffen van Bakel

[1]Department of Computing, Imperial College London, 180 Queen's Gate, London SW7 2BZ, UK
s.vanbakel@imperial.ac.uk

Abstract

We present $\mathcal{L}$, an extension of Parigot's $\lambda\mu$-calculus by adding negation as a type constructor, together with syntactic constructs that represent negation introduction and elimination.
We will define a notion of reduction that extends $\lambda\mu$'s reduction system with two new reduction rules, and show that the system satisfies subject reduction. Using Aczel's generalisation of Tait and Martin-Löf's notion of parallel reduction, we show that this extended reduction is confluent.
Although the notion of type assignment has its limitations with respect to representation of proofs in natural deduction with implication and negation, $\vdash_{\mathrm{NI}}$, we will show that all propositions that can be shown in $\vdash_{\mathrm{NI}}$ have a witness in $\mathcal{L}$.
Using Girard's approach of reducibility candidates, we show that all typeable terms are strongly normalisable, and conclude the paper by showing that type assignment for $\mathcal{L}$ enjoys the principal typing property.

keywords: classical logic, lambda calculus, negation, confluence, termination

Introduction

Intuitionistic Logic (IL) [10, 11, 12] plays an important role in Computer Science, given its strong relation with types in functional programming and the λ-calculus [14, 8] through the Curry-Howard isomorphism [24], *i.e.* through the fact that typeable functions in a functional programming language correspond to proofs in IL, and provable properties to inhabitable types. Its importance is most prominent in the context of proof assistants, that all seem to be rooted in IL. Proof assistants or theorem provers can also be seen as programming languages for which the type system corresponds to a formal logic and ensure proof correctness by capitalising on the Curry-Howard correspondence through their type system. Under this correspondence, checking that a term has a type is operationally equivalent to checking a proof of a proposition [43].

There are currently many different proof assistants in use, that come in different shapes and forms, each with their own characteristic: Coq [13] has a particular focus on the theorem proving aspect where proofs can be written with intuitive tactics, whereas Agda [28] and Idris [9] are more deeply connected to functional programming languages like Haskell.

The more widely used proof assistants are all founded on *intuitionistic type theory* [25]. However, the use of IL inescapably limits these languages to the fact that they are unable to prove a simple notion, which use is widespread in normal, everyday mathematics: a proposition is either true or false. This is known as the law of the excluded middle (LEM), and is the distinguishing feature of Classical Logic (CL) [18, 41].

It can be argued that IL very rightly rejects this notion and there are many that do exactly that: they stress the value of IL, where a proof of 'A or B' must be *constructive*, *i.e.* constructed from a proof of either A or from a proof of B, so stating 'A or not A', without justifying either first, is unacceptable. Likewise, a proof for the statement $\exists\, x \in C\ (Q(x))$ is only acceptable if first $Q(c)$ is shown, for some object $c \in C$ (*i.e.* a *witness* for Q has been produced). Therefore $\neg\forall\, x \in C\ (P(x)) \Rightarrow \exists\, x \in C\ (\neg P(x))$ cannot be shown in IL, since knowing that there has to be an element in C for which P does not hold is not the same as knowing which element that is. Accepting IL as the basis for mathematical reasoning, which for many is the only right thing to do for philosophical reasons, severely limits the collection of provable results, and is therefore not a popular choice amongst mathematicians. Some theorem provers, perhaps begrudgingly, allow for the addition of the axiom 'A or not A', witnessed through a term constant; although it allows for provability of mathematical statements, this approach does not lend computational context to proofs, as theorem provers for IL do, and is more of a hack than a solution.

In fact, the popularity of IL, constructive logic and constructive mathematics in computer science can be explained through its strong ties with computability through the Curry-Howard correspondence and the relation between the IL, the λ-calculus and functional programming. This link was so strong that until fairly recently it was believed that a proof had a correspondence with a function only if the proof were constructive, and that classical logic did not have a computational counterpart.

That situation changed when Griffin [20] observed that the C-operator of Felleisen's λC-calculus [17], similar to the call/cc function in Scheme, can be typed with $\neg\neg A \to A$ (or rather $((A \to \bot) \to \bot) \to A$), *double negation elimination*, another property that only holds in CL, thus highlighting the first link between CL and sequential control in computer science. This soon led to the definition of $\lambda\mu$ by Parigot [30, 31], a calculus that represents minimal classical logic [3], followed by an impressive body of work in the area of CL and computer science, with many contributions from various authors.

Looking to investigate the possibility and suitability of developing theorem provers for CL base on $\lambda\mu$, in [16] the case was made that in terms of implementability, expressiveness, and elegance, proof assistants based on CL have much to contribute. It presented Candid, a theorem prover based on $\lambda\mu$, but enriched with dependent types, as an extension of $\mathrm{ECC_K}$ [27] adding co-products and dependent algebraic data types. It treated a system of classical natural deduction that uses the logical connectors *implication*, *negation*, *conjunction*, and *disjunction*.

As seen in that paper, the link between first order classical logic and computation is tricky. Theorem provers are based in a dependently typed systems, but [22] showed that by naively combining dependent types and $\lambda\mu$'s control operators, all types have only one inhabitant. Fortunately, [23] shows a way to restrict how dependent types and control operators can interact, which regains a logically consistent type theory. An important notion to address this problem is the use of *negative elimination free* (NEF) terms that cannot contain a negation elimination. Since in $\lambda\mu$ negation elimination gets represented through application, as well as through *naming* (see Example 1.13), this restriction is quite drastic. Although it is unavoidable for a NEF term to not contain sub-terms of the form $\mu\alpha.[\beta]N$ as the subterm $[\beta]N$ corresponds with an application of $(\neg E)$, it also cannot contain an application MN, as this could correspond with $(\neg E)$ when M has type $A \to \bot$ and N has type A. Introducing separate syntax for negation, as we do here, strongly expands the set of NEF-terms to those really not dealing with negation, and will strengthen the implementation of Candid.

Another reason to deal with negation explicitly is the fact that $\lambda\mu$ does not really represent CL, in that tautologies are not necessarily represented by closed terms. This is in part due to the fact the system only has implicit negation and (PBC) to express dealing with conflict, so negation $\neg A$ is expressed through $A \to \bot$ (where $\bot$ is not a type in the original presentation), negation introduction through abstraction and negation elimination through application. For example, in Example 1.13 we will show Parigot's proof for double negation elimination in $\lambda\mu$; the witness $\lambda y.\mu\alpha.[\gamma]y(\lambda x.\mu\delta.[\alpha]x)$ contains a free name γ of type $\bot$. It is needed because the subterm $y(\lambda x.\mu\delta.[\alpha]x)$ has type $\perp$, and the only way to deal with that in $\lambda\mu$ is applying the rule for (PBC), which forces the prefix $\mu\alpha.[\gamma]$ to the term. We will see that, dealing explicitly with negation, this problem disappears.

This paper presents $\mathcal{L}$ and shows all the necessary properties for it, like soundness, confluence, expressiveness, termination, and principal typing.

Overview

This paper introduces the calculus $\mathcal{L}$, which expands on $\lambda\mu$ by adding negation. We will start in Section 1 with an overview of two of the common representations of CL, where we will focus on natural deduction and proof contraction, and why double negation elimination poses a particular problem for the latter. We will define $\vdash_{\mathrm{NI}}$, a restriction to natural deduction of CL that uses negation and implication, and plays a central role in this paper. We will revisit Parigot's $\lambda\mu$ also through its underlying logic, and explain how it deals with negation, implicitly through assumptions stored in the co-context, and explicitly through $\cdot\rightarrow\perp$. We in particular highlight that $\lambda\mu$ is not fully equipped to deal with the latter kind of negation, as witnesses to tautologies not necessarily are closed terms. We also revisit Summer's $\nu\lambda\mu$-calculus that fully represents $\vdash_{\mathrm{NI}}$, together with its non-confluent notion of reduction.

In Section 2 we define the calculus $\mathcal{L}$ as an extension of $\lambda\mu$ by adding syntax and inference rules that express negation; it can also be seen as a restriction of $\nu\lambda\mu$. This calculus comes with four elementary notions of reduction, and we will show soundness results for all of them. This is followed in Section 3 by the proof that reduction is confluent, and in Section 4 by the proof that, although a restriction of $\nu\lambda\mu$, $\mathcal{L}$ can still inhabit all provable judgements of $\vdash_{\mathrm{NI}}$. Then in Section 5, we will show that reduction is strongly normalisable, and conclude in Section 6 by showing that type assignment enjoys the principal typing property.

1 Natural Deduction for Classical Logic

Natural Deduction for CL, defined by Gentzen in [18] is a way of describing the structure of formal proofs in mathematics that follow the intuitive, human, lines of reasoning as much as possible. It is defined through *inference rules* that are generically of the shape

$$\frac{\textit{Premisses}}{\textit{Conclusion}}\ (\textit{Rule})$$

and describes a step allowed in this formal system, where, assuming that all the statements in the premisses hold, then after applying this step named (*Rule*) we can accept that the conclusion holds as well. Statements, also called judgements, are of the shape $\Gamma \vdash A$, where A is a formula and Γ is a *context*, a collection of formulas that form the assumptions needed for A to hold, and express that 'if all formulas in the collection Γ hold, then so does A'. A number of these can together form the premisses; there is only one judgement in the conclusion.[1]

Proofs are constructed by applying rules to each other, in the sense that the conclusion of one rule can be a premise of another. The premises on the initial rules (that are not the conclusion of other rules) are called the assumptions of the proof; these are usually not extended while the construction of the proof progresses, but only decrease as the proof evolves; the

[1] An different notation can be found in the literature, where inference rules express the relation between the inferred formulas, without stating the context, and the assumptions are the formulas occurring in the leaves of the derivation tree. Assumptions can be cancelled through steps like ($\rightarrow$I), and are then placed between square brackets or struck through. Since the latter is a non-local operation on the inference tree that is not easily defined or treated formally, here we prefer the 'sequent' notation, since it neatly collects in the derived statement the assumptions on which it depends in the context Γ.

(single) conclusion occurs at the bottom. Judgements that are considered to be proven are those that appear at the bottom of the derivation tree.

The inference rules of natural deduction systems almost all come in two varieties for each logical operator: introduction and elimination rules, each for any particular logical connective. For example, for the logical operator $\wedge$ (*conjuction*) and $\vee$ (*disjuction*), these rules look like:

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ (\wedge\mathbf{I}) \qquad\qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \quad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ (\wedge\mathbf{E})$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \quad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ (\vee I) \qquad\qquad \frac{\Gamma \vdash A \vee B \qquad \Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma \vdash C}\ (\vee E)$$

To deal with conclusions that need no premisses since they hold by themselves, an *axiom* rule is added; these form the 'leaves' of the proof tree.

$$\frac{}{\Gamma, A \vdash A}\ (\mathrm{AX})$$

(if we assume that A holds, then it does.)

In his paper, Gentzen also presents the Sequent Calculus, which differs from Natural Deduction in that it derives sequences of the shape

$$A_1, \ldots, A_n \vdash_{\mathrm{LK}} B_1, \ldots, B_m$$

with the intended meaning 'if all of the properties $A_1, \ldots, A_n$ hold, then at least one of the $B_1, \ldots, B_m$ does as well.' For each connector, there is a *left* and *right* introduction rule, as in

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta} \quad \frac{\Gamma, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\ (\wedge\mathbf{L}) \qquad\qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}\ (\wedge\mathbf{R})$$

There are no elimination rules for connectors, just a generic (cut)-rule:

$$\frac{\Gamma \vdash C, \Delta \quad \Gamma, C \vdash \Delta}{\Gamma \vdash \Delta}\ (cut)$$

where C of course can be $A \wedge B$.

For this logical system, Gentzen defines a notion of (proof) contraction that removes occurrences of (cut), and shows that this is terminating: for every proof that shows $\Gamma \vdash_{\mathrm{LK}} \Delta$, there exists a (cut)-free proof that shows the same result. He does not show that result for Natural Deduction, which would eliminate all introduction-elimination pairs, and we can guess that that is because this notion of contraction is harder to define.[2] Prawitz [35] presented an extensive study of proof contraction for Natural Deduction.

The main difficulty is that in the Sequent Calculus, all logical connectors come with a left and a right introduction rule, whereas in Natural Deduction, not all proof-constructions follow the introduction-elimination pattern of the inference rules. For those that do, proof contraction consists of the removal from a proof of an introduction step followed immediately by an elimination step for the same logical connector; for '$\wedge$' that looks like:

$$\dfrac{\dfrac{\dfrac{\boxed{D_1}}{\Gamma \vdash A} \quad \dfrac{\boxed{D_2}}{\Gamma \vdash B}}{\Gamma \vdash A \wedge B}\ (\wedge\mathbf{I})}{\Gamma \vdash A}\ (\wedge\mathbf{E}) \ \Rightarrow\ \dfrac{\boxed{D_1}}{\Gamma \vdash A} \qquad\qquad \dfrac{\dfrac{\dfrac{\boxed{D_1}}{\Gamma \vdash A} \quad \dfrac{\boxed{D_2}}{\Gamma \vdash B}}{\Gamma \vdash A \wedge B}\ (\wedge\mathbf{I})}{\Gamma \vdash B}\ (\wedge\mathbf{E}) \ \Rightarrow\ \dfrac{\boxed{D_2}}{\Gamma \vdash B}$$

or, for implication:

[2] One particular difficulty with defining proof contractions on either the Sequent Calculus or Natural Deduction is that this notion is not *confluent*, in that proof contraction not always leads to the same result.

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\,(\to\textsc{i}) \qquad\qquad \dfrac{\dfrac{\begin{array}{c}\overline{\Gamma, A \vdash A}\,(\textsc{ax}) \\ \boxed{D_1} \\ \Gamma, A \vdash B\end{array}}{\Gamma \vdash A \to B}\,(\to\textsc{i}) \quad \begin{array}{c}\boxed{D_2} \\ \Gamma \vdash A\end{array}}{\Gamma \vdash B}\,(\to\textsc{e}) \quad \Rightarrow \quad \begin{array}{c}\boxed{D_2} \\ \Gamma \vdash A \\ \boxed{D_1} \\ \Gamma \vdash B\end{array}$$

$$\frac{\Gamma \vdash A \to B \quad \Gamma \vdash A}{\Gamma \vdash B}\,(\to\textsc{e})$$

Notice that, in the rule $(\to\textsc{i})$, the formula A ceases to be an assumption, and that, in the composed proof on the right, A is no longer an assumption needed to reach the conclusion, since it has been shown to hold by D_2.

This is not possible for all logical connectors: the way negation is dealt with is, for example, not straightforward. Negation comes of course with introduction and elimination rules:

$$\frac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}\,(\neg\textsc{i}) \qquad\qquad \dfrac{\dfrac{\begin{array}{c}\overline{\Gamma, A \vdash A}\,(\textsc{ax}) \\ \boxed{D_1} \\ \Gamma, A \vdash \bot\end{array}}{\Gamma \vdash \neg A}\,(\neg\textsc{i}) \quad \begin{array}{c}\boxed{D_2} \\ \Gamma \vdash A\end{array}}{\Gamma \vdash \bot}\,(\neg\textsc{e}) \quad \Rightarrow \quad \begin{array}{c}\boxed{D_2} \\ \Gamma \vdash A \\ \boxed{D_1} \\ \Gamma \vdash \bot\end{array}$$

$$\frac{\Gamma \vdash \neg A \quad \Gamma \vdash A}{\Gamma \vdash \bot}\,(\neg\textsc{e})$$

but, in Classical Logic, negation plays a more intricate role, in that the *law of excluded middle* '$A \vee \neg A$ is true for all A' (or something similar, like 'there is no distinction between the formulas $\neg\neg A$ and A') holds. This is a property that cannot be shown, but has to forced onto the system, and can cause havoc for proof contraction.

There are many different rules that express this to a different degree, like:

$$\frac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}\,(\textsc{pbc}) \quad \frac{\Gamma \vdash \neg\neg A}{\Gamma \vdash A}\,(\textsc{dne}) \quad \frac{}{\vdash A \vee \neg A}\,(\textsc{lem}) \quad \frac{}{\vdash ((A \to B) \to A) \to A}\,(\textsc{pl}) \quad \frac{}{\vdash (\neg A \to A) \to A}\,(\textsc{raa})$$

(called '*proof by contradiction*', '*double negation elimination*', '*law of excluded middle*', '*Peirce's law*', and *reductio ad absurdum*, respectively.) These rules have different expressive power, and adding one rather than another can change the set of derivable properties (see [2]).

The variant of Classical Natural Deduction we will consider in this paper uses the logical connectors $\neg$ (*negation*) and $\to$ (*implication*).

Definition 1.1 (Natural deduction with negation and implication) The formulas we use for our system of natural deduction with negation and implication are:

$$A, B \ ::= \ \varphi \mid A \to B \mid \neg A$$

where '$\to$' associates to the right and '$\neg$' binds stronger than '$\to$'. A context Γ is a set of formulas, where $\Gamma, A = \Gamma \cup \{A\}$ and the inference rules are:

$$(\textsc{ax}): \frac{}{\Gamma, A \vdash A} \qquad (\to\textsc{i}): \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B} \qquad (\to\textsc{e}): \frac{\Gamma \vdash A \to B \quad \Gamma \vdash A}{\Gamma \vdash B}$$

$$(\textsc{pbc}): \frac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A} \qquad (\neg\textsc{i}): \frac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A} \qquad (\neg\textsc{e}): \frac{\Gamma \vdash \neg A \quad \Gamma \vdash A}{\Gamma \vdash \bot}$$

We write $\Gamma \vdash_{\mathrm{NI}} A$ for judgements derivable in this system, and $\vdash_{\mathrm{NI}}$ as name for the system.

Notice that $\bot$ is not a formula in $\vdash_{\mathrm{NI}}$, but only represents conflict; it could be omitted from the system, by deriving also $\Gamma \vdash$.

As suggested above, in the presence of $(\textsc{pbc})$ defining proof contraction is not straightforward.

Example 1.2 Take the following proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{D_1}{\Gamma,\neg(A\to B)\vdash\neg\neg(A\to B)} \qquad \dfrac{}{\neg(A\to B)\vdash\neg(A\to B)}\,(\mathrm{AX})}{\Gamma,\neg(A\to B)\vdash\bot}\,(\neg\mathrm{E})}{\Gamma\vdash A\to B}\,(\mathrm{PBC}) \qquad \dfrac{D_2}{\Gamma\vdash A}}{\Gamma\vdash B}\,(\to\mathrm{E})$$

It is *a priori* not clear how to contract this proof. We would like to use the sub-derivations to be the building stones for the proof for $\Gamma \vdash B$ without the (PBC)-$(\to\mathrm{E})$ pair, but it is not immediately clear how to do that: there is no sub-derivation above the step (PBC) that has A as an assumption (so does not contain $\Gamma, A \vdash A$ as the result of rule (AX)), or that derives $\Gamma \vdash A\to B$.

There are many ways around this problem presented in the literature, but at this point we just want to highlight the problem. We will see in Section 1.3 a term calculus that directly represents proofs in $\vdash_{\mathrm{NI}}$, and presents a solution by presenting a different kind of term substitution.

To better be able to reason about the structure of proofs and the technicalities of proof contraction, we need to represent the structure of proofs via term information from an appropriate calculus, and inhabit the inference rules with terms, such that proof contractions will come to correspond to term reduction. This employs the Curry-Howard principle, which expresses a correspondence between terms and their types on one side, and proofs for propositions on the other. We will see below that associating a term calculus to an inference system unlocks the subtle differences between the variants of Classical Logic we consider here.

The natural way to inhabit $\vdash_{\mathrm{NI}}$ is using Summer's $\nu\lambda\mu$ [40]; we will first present Parigot's calculus $\lambda\mu$ [32], as this historically came first, and gives a very elegant solution to the proof-contraction problem of Example 1.2.

1.1 A classical logic with focus

Parigot's $\lambda\mu$-calculus is a proof-term syntax for classical logic, expressed in Natural Deduction, defined as an extension of the Curry type assignment system for the λ-calculus. With $\lambda\mu$ Parigot created a multi-conclusion typing system which corresponds to a classical logic with *focus*; there derivable statements have the shape $\Gamma \vdash A \mid \Delta$, where A is the main conclusion of the statement, expressed as the *active* conclusion, Γ is the set of assumptions and Δ is the set of alternative conclusions, or have the shape $\Gamma \vdash \bot \mid \Delta$ if there is no formula under focus.

Before discussing $\lambda\mu$, in order to better compare it with the other calculi we discuss in this paper, we first revise its underlying logic, which corresponds to the following:

Definition 1.3 (A CLASSICAL LOGIC WITH FOCUS) The formulas for this system are:

$$A, B \ ::= \ \varphi \mid A \to B$$

and a context Γ is a set of formulas, where $\Gamma, A = \Gamma \cup \{A\}$ and the inference rules are defined through:

$$(\mathrm{AX}): \dfrac{}{\Gamma, A \vdash A \mid \Delta} \qquad (\to\mathrm{I}): \dfrac{\Gamma, A \vdash B \mid \Delta}{\Gamma \vdash A \to B \mid \Delta} \qquad (\to\mathrm{E}): \dfrac{\Gamma \vdash A \to B \mid \Delta \quad \Gamma \vdash A \mid \Delta}{\Gamma \vdash B \mid \Delta}$$

$$(\mathbf{ACT}): \dfrac{\Gamma \vdash \bot \mid A, \Delta}{\Gamma \vdash A \mid \Delta} \qquad (\mathbf{PASS}): \dfrac{\Gamma \vdash A \mid A, \Delta}{\Gamma \vdash \bot \mid A, \Delta}$$

We write $\Gamma \vdash_{\mathrm{F}} M : A \mid \Delta$ for judgements derivable in this system.

Example 1.4 Notice that negation is not part of the type language, so does not occur in Γ nor in Δ. It is therefore not possible to show $(\mathbf{DNE})$ in this logic (not without first extending the syntax of formulas, see Example 1.13); however, it is possible to show Peirce's law:

$$\dfrac{\dfrac{}{(A\to B)\to A \vdash_{\mathrm{F}} (A\to B)\to A \mid A}\;(\textsc{ax}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{(A\to B)\to A, A \vdash_{\mathrm{F}} A \mid A, B}\;(\textsc{ax})}{(A\to B)\to A, A \vdash_{\mathrm{F}} \bot \mid A, B}\;(\textbf{PASS})}{(A\to B)\to A, A \vdash_{\mathrm{F}} B \mid A}\;(\textbf{ACT})}{(A\to B)\to A \vdash_{\mathrm{F}} A\to B \mid A}\;(\to\textsc{i})}{\dfrac{\dfrac{\dfrac{(A\to B)\to A \vdash_{\mathrm{F}} A \mid A}{(A\to B)\to A \vdash_{\mathrm{F}} \bot \mid A}\;(\textbf{PASS})}{(A\to B)\to A \vdash_{\mathrm{F}} A \mid}\;(\textbf{ACT})}{\vdash_{\mathrm{F}} ((A\to B)\to A)\to A \mid}\;(\to\textsc{i})}\;(\to\textsc{e})$$

The intention of this system is to express classical logic, and for this it encapsulates the rule $(\textsc{pbc})$. To see this, we need first to emphasise that the formulas in Δ are seen as *negated*. In fact, any statement $\Gamma \vdash_{\mathrm{F}} A \mid \Delta$ can be seen as $\Gamma, \neg\Delta \vdash_{\mathrm{NI}} A$ (where $\neg\Delta$ lists the negated versions of all types in Δ). With that view, the rules $(\textbf{ACT})$ and $(\textbf{PASS})$ corresponds to allowing the following variants of rule $(\textsc{pbc})$ and $(\neg\textsc{e})$

$$\dfrac{\Gamma, \neg\Delta, \neg A \vdash \bot}{\Gamma \vdash A}\;(\textsc{pbc}) \qquad\qquad \dfrac{\dfrac{}{\Gamma, \neg\Delta, \neg A \vdash \neg A}\;(\textsc{ax}) \qquad \Gamma, \neg\Delta, \neg A \vdash A}{\Gamma, \neg\Delta, \neg A \vdash \bot}\;(\neg\textsc{e})$$

but in a version of Natural Deduction where formulas have at most a negation at the front. Note that it therefore solves the problem of Example 1.2 by not allowing the rule $(\textsc{pbc})$ to be applied to assumptions on the right in $(\neg\textsc{e})$: A cannot be a negated type, so the premises in the right-hand proof cannot occur swapped.

Example 1.5 We can construct in $\vdash_{\mathrm{F}}$ and $\vdash_{\mathrm{NI}}$, respectively:

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash A\to B \mid A\to B, \Delta}{\Gamma \vdash \bot \mid A\to B, \Delta}\;(\textbf{PASS})}{\Gamma \vdash A\to B \mid \Delta}\;(\textbf{ACT}) \qquad \Gamma \vdash A \mid \Delta}{\Gamma \vdash B \mid \Delta}\;(\to\textsc{e}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, \neg\Delta, \neg A\to B \vdash \neg A\to B}\;(\textsc{ax}) \qquad \Gamma, \neg\Delta, \neg A\to B \vdash A\to B}{\Gamma, \neg\Delta, \neg A\to B \vdash \bot}\;(\neg\textsc{e})}{\Gamma, \neg\Delta \vdash A\to B}\;(\textsc{pbc}) \qquad \Gamma, \neg\Delta \vdash A}{\Gamma, \neg\Delta \vdash B}\;(\to\textsc{e})$$

(notice that there now is a subderivation for $A\to B$) which can be contracted to, respectively:

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash A\to B \mid B, \Delta \qquad \Gamma \vdash A \mid B, \Delta}{\Gamma \vdash B \mid B, \Delta}\;(\to\textsc{e})}{\Gamma \vdash \bot \mid B, \Delta}\;(\textbf{PASS})}{\Gamma \vdash B \mid \Delta}\;(\textbf{ACT}) \qquad \dfrac{\dfrac{\dfrac{}{\Gamma, \neg\Delta, \neg B \vdash \neg B}\;(\textsc{ax}) \qquad \dfrac{\Gamma, \neg\Delta, \neg B \vdash A\to B \qquad \Gamma, \neg\Delta, \neg B \vdash A}{\Gamma, \neg\Delta, \neg B \vdash B}\;(\to\textsc{e})}{\Gamma, \neg\Delta \vdash \bot}\;(\neg\textsc{e})}{\Gamma, \neg\Delta \vdash B}\;(\textsc{pbc})$$

(Notice that weakening is permitted in that we can always add formulas to the context without affecting the result.) This forms the basis of structural reduction in $\lambda\mu$.

1.2 The $\lambda\mu$-calculus

We now present the variant of $\lambda\mu$ we consider in this paper, as defined by Parigot in [31] and that gives a Curry-Howard interpretation to the above inference rules:

Definition 1.6 (Syntax of $\lambda\mu$) The $\lambda\mu$*-terms* we consider are defined by the grammar:

$$\begin{array}{rcll} M, N & ::= & V \mid MN \mid \mu\alpha.[\beta]M & \\ V & ::= & x \mid \lambda x.M & \qquad (\textit{values}) \end{array}$$

We will use C for the pseudo terms $[\beta]M$.

Recognising both λ and μ as binders, the notion of free and bound names and variables is defined as usual, and we accept Barendregt's convention to keep free and bound names and variables distinct, using (silent) α-conversion whenever necessary.

We write $x \in M$ ($\alpha \in M$) if x (α) occurs in M, either free of bound, and call a term *closed* if it has no free names or variables. We will call the pseudo-terms of the shape $[\alpha]M$ *commands*, and write C, and treat them as terms for reasons of brevity, whenever convenient.

As with Implicative Intuitionistic Logic, the reduction rules for the terms that represent the proofs correspond to proof contractions, but in $\vdash_{\mathrm{F}}$. The reduction rules for the λ-calculus are the *logical* reductions, *i.e.* deal with the removal of a introduction-elimination pair for a type construct and in addition to these, Parigot expresses also the *structural* rules that change the focus of a proof, where elimination essentially deals with negation and takes place for a type constructor that appears in one of the alternative conclusions (the Greek variable is the name given to a subterm). Parigot therefore needs to express that the focus of the derivation (proof) changes (see the rules in Definition 1.9), and this is achieved by extending the syntax with two new constructs $[\alpha]M$ and $\mu\alpha.M$[3] that act as witness to *passivation* and *activation* of $\vdash_{\mathrm{F}}$, which together move the focus of the derivation, and together are called a *context switch*.

In $\lambda\mu$, reduction of terms is expressed via implicit substitution, and as usual, $M\{N/x\}$ stands for the (instantaneous) substitution of all occurrences of x in M by N. Two kinds of structural substitution are defined: the first is the standard one, where $M\{N{\cdot}\gamma/\alpha\}$ stands for the term obtained from M in which every command of the form $[\alpha]P$ is replaced by $[\gamma]PN$ (here γ is a fresh name). The second will be of use for CBV reduction, were $\{N{\cdot}\gamma/\alpha\}M$ stands for the term obtained from M in which every $[\alpha]P$ is replaced by $[\gamma]NP$.

They are formally defined by:

Definition 1.7 (STRUCTURAL SUBSTITUTION) *Right-structural substitution*, $M\{N{\cdot}\gamma/\alpha\}$, and *left-structural substitution*, $\{N{\cdot}\gamma/\alpha\}M$, are defined inductively over (pseudo) terms. The important cases are:

$$\begin{array}{ll} [\alpha]M\{N{\cdot}\gamma/\alpha\} \triangleq [\gamma](M\{N{\cdot}\gamma/\alpha\}N) & \{N{\cdot}\gamma/\alpha\}[\alpha]M \triangleq [\gamma]N(\{N{\cdot}\gamma/\alpha\}M) \\ [\beta]M\{N{\cdot}\gamma/\alpha\} \triangleq [\beta](M\{N{\cdot}\gamma/\alpha\})\ (\beta \neq \alpha) & \{N{\cdot}\gamma/\alpha\}[\beta]M \triangleq [\beta]\{N{\cdot}\gamma/\alpha\}M\ (\beta \neq \alpha) \end{array}$$

Parigot'92 only defines the first variant of these notions of structural substitutions (so does not use the prefix 'right'); the two notions are defined together, but rather informally, using a notion of contexts in [29].

We have the following notions of reduction on $\lambda\mu$. For the third, call by value, different variants exists in the literature; we adopt the one from [29].

Definition 1.8 ($\lambda\mu$ REDUCTION) *i*) The reduction rules of $\lambda\mu$ are:

$$\begin{array}{rrcll} \textit{logical}\ (\beta): & (\lambda x.M)N & \to & M\{N/x\} & \\ \textit{structural}\ (\mu): & (\mu\alpha.\mathrm{C})N & \to & \mu\gamma.\mathrm{C}\{N{\cdot}\gamma/\alpha\} & (\gamma\ \textit{fresh}) \\ \textit{erasing}\ (\theta): & \mu\alpha.[\alpha]M & \to & M & (\alpha \notin M) \\ \textit{renaming}\ (\rho): & [\beta]\mu\gamma.\mathrm{C} & \to & \mathrm{C}\{\beta/\gamma\} & \end{array}$$

ii) Evaluation contexts are defined as terms with a single hole $\lceil\,\rfloor$ by:

$$\mathtt{C} ::= \lceil\,\rfloor \mid \mathtt{C}M \mid M\mathtt{C} \mid \lambda x.\mathtt{C} \mid \mu\alpha.[\beta]\mathtt{C}$$

We write $\mathtt{C}\lceil M\rfloor$ for the term obtained by replacing the hole with the term M.

(Free, unconstrained) reduction $\to_{\beta\mu}$ on $\lambda\mu$-terms is defined through $\mathtt{C}\lceil M\rfloor \to_{\mathrm{N}} \mathtt{C}\lceil N\rfloor$ if $M \to N$ using either the β, μ, θ, or ρ-reductions rule.

[3] Notice that these constructs are *pseudo* terms in that they always occur together in terms.

iii) CBN *evaluation contexts* are defined as:

$$\mathsf{C_N} \ ::= \ \lceil\,\rfloor \mid \mathsf{C_N}M \mid \mu\alpha.[\beta]\mathsf{C_N}$$

CBN reduction $\rightarrow_{\text{N}}$ is defined through: $\mathsf{C_N}\lceil M\rfloor \rightarrow_{\text{N}} \mathsf{C_N}\lceil N\rfloor$ if $M \rightarrow N$ using either the β, μ, θ, or ρ-reduction rule.

iv) CBV *evaluation contexts* are defined through:

$$\mathsf{C_V} \ ::= \ \lceil\,\rfloor \mid \mathsf{C_V}M \mid V\mathsf{C_V} \mid \mu\alpha.[\beta]\mathsf{C_V}$$

CBV reduction $\rightarrow_{\text{V}}$ is defined through: $\mathsf{C_V}\lceil M\rfloor \rightarrow_{\text{V}} \mathsf{C_V}\lceil N\rfloor$ if $M \rightarrow N$ using either μ, θ, ρ, or:

$$\begin{array}{lrcl} (\beta_{\text{V}}): & (\lambda x.M)\,V & \rightarrow_{\text{N}} & M\{V/x\} \\ (\mu_{\text{V}}): & V\,(\mu\alpha.\mathrm{C}) & \rightarrow_{\text{N}} & \mu\gamma.\{V\cdot\gamma/\alpha\}\mathrm{C} \quad (\gamma \textit{ fresh}) \end{array}$$

Remark that, for rule (μ_{V}), $\mu\alpha.[\beta]N$ is not a value. Also, unlike for the λ-calculus, CBV reduction is not a sub-reduction system of $\rightarrow_{\beta\mu}$: the rule (μ_{V}) (and left-structural substitution) are not part of $\rightarrow_{\beta\mu}$. Both CBN and CBV constitute *reduction strategies* in that they pick exactly one $\beta\mu$-redex to contract; notice that a term might be in either CBN or CBV-normal form (*i.e.* reduction has stopped), but not need be that for $\rightarrow_{\beta\mu}$.

Type assignment for $\lambda\mu$ is defined below through inhabiting the inference rules of $\vdash_{\text{F}}$ with syntax; there is a *main*, or *active*, conclusion, labelled by a term, and the *alternative* conclusions are labelled by names α, β, *etc*. Judgements in $\lambda\mu$ are of the shape $\Gamma \vdash_{\lambda\mu} M : A \mid \Delta$, where Δ consists of pairs of Greek characters (the *names*) and types; the left-hand context Γ, as for the λ-calculus, contains pairs of Roman characters and types, and represents the types of the free term variables of M.

Definition 1.9 (TYPING RULES FOR $\lambda\mu$) *i*) Let φ range over a countable (infinite) set of type-variables. The set of types is defined by the grammar:

$$A,B \ ::= \ \varphi \mid A \rightarrow B$$

ii) A *context* (of term variables) Γ is a partial mapping from term variables to types, denoted as a finite set of *statements* $x{:}A$, such that the *subjects* of the statements (x) are distinct. We write Γ_1,Γ_2 for the *compatible* union of Γ_1 and Γ_2 (if $x{:}A_1 \in \Gamma_1$ and $x{:}A_2 \in \Gamma_2$, then $A_1 = A_2$), and write $\Gamma,x{:}A$ for $\Gamma,\{x{:}A\}$, $x \notin \Gamma$ if there exists no A such that $x{:}A \in \Gamma$, and $\Gamma \backslash x$ for $\Gamma \backslash \{x{:}A\}$.

iii) A *context of names* Δ (or *co-context*) is a partial mapping from *names* to types, denoted as a finite set of *statements* $\alpha{:}A$, such that the *subjects* of the statements (α) are distinct. Notions Δ_1,Δ_2, as well as $\Delta,\alpha{:}A$ and $\alpha \notin \Delta$ are defined as for Γ.

iv) The type assignment rules for $\lambda\mu$, adapted to our notation, are:

$$(\text{AX}): \frac{}{\Gamma,x{:}A \vdash x : A \mid \Delta} \qquad (\rightarrow\text{I}): \frac{\Gamma,x{:}A \vdash M : B \mid \Delta}{\Gamma \vdash \lambda x.M : A\rightarrow B \mid \Delta}\,(x \notin \Gamma) \qquad (\rightarrow\text{E}): \frac{\Gamma \vdash M : A\rightarrow B \mid \Delta \quad \Gamma \vdash N : A \mid \Delta}{\Gamma \vdash MN : B \mid \Delta}$$

$$(\mu): \frac{\Gamma \vdash M : B \mid \alpha{:}A,\beta{:}B,\Delta}{\Gamma \vdash \mu\alpha.[\beta]M : A \mid \beta{:}B,\Delta}\,(\alpha \notin \Delta) \qquad \frac{\Gamma \vdash M : A \mid \alpha{:}A,\Delta}{\Gamma \vdash \mu\alpha.[\alpha]M : A \mid \Delta}\,(\alpha \notin \Delta)$$

We will write $\Gamma \vdash_{\lambda\mu} M : A \mid \Delta$ for statements derivable in this system.

v) We extend Barendregt's convention on free and bound variables and names to judgements (for all the notions of type assignment we define here), so in $\Gamma,x{:}A \vdash_{\lambda\mu} M : B \mid \alpha{:}C,\Delta$, both x and α cannot appear bound in M.

We can think of $[\alpha]M$ as storing the type of M amongst the alternative conclusions by giving it the name α.

$$\dfrac{\dfrac{\begin{array}{c}\dfrac{\begin{array}{c}\boxed{D_1}\\ \Gamma \vdash M : A{\to}B \mid \alpha{:}A{\to}B,\gamma{:}D,\Delta\end{array}}{\Gamma \vdash \mu\gamma.[\alpha]M : D \mid \alpha{:}A{\to}B,\Delta}\,(\mu)\\ \boxed{D_2}\\ \Gamma \vdash \mathsf{C}\lceil\mu\gamma.[\alpha]M\rfloor : C \mid \alpha{:}A{\to}B,\Delta\end{array}}{\Gamma \vdash \mu\alpha.[\beta]\mathsf{C}\lceil\mu\gamma.[\alpha]M\rfloor : A{\to}B \mid \Delta}\,(\mu) \quad \begin{array}{c}\boxed{D_3}\\ \Gamma\vdash N:A\mid\Delta\end{array}}{\Gamma \vdash (\mu\alpha.[\beta]\mathsf{C}\lceil\mu\gamma.[\alpha]M\rfloor)\,N : B \mid \Delta}\,(\to\mathrm{E})$$

$$\dfrac{\begin{array}{c}\dfrac{\dfrac{\begin{array}{c}\boxed{D_1}\\ \Gamma \vdash M : A{\to}B \mid \delta{:}B,\gamma{:}D,\Delta\end{array} \quad \dfrac{\begin{array}{c}\boxed{D_3}\\ \Gamma\vdash N:A\mid\Delta\end{array}}{\Gamma \vdash N : A \mid \delta{:}B,\gamma{:}D,\Delta}\,(Wk)}{\Gamma \vdash MN : B \mid \delta{:}B,\gamma{:}D,\Delta}\,(\to\mathrm{E})}{\Gamma \vdash \mu\gamma.[\delta]MN : D \mid \delta{:}B,\Delta}\,(\mu)\\ \boxed{D_2'}\\ \Gamma \vdash \mathsf{C}\lceil\mu\gamma.[\delta]MN\rfloor : C \mid \delta{:}B,\Delta\end{array}}{\Gamma \vdash \mu\delta.[\beta]\mathsf{C}\lceil\mu\gamma.[\delta]MN\rfloor : B \mid \Delta}\,(\mu)$$

Figure 1. An example of structural reduction in $\lambda\mu$.

Notice that, if we erase all term information from the inference rules, we get the rules from $\vdash_{\mathrm{F}}$, but for the variants of (μ); these we can infer, however, so they are admissible.

$$\dfrac{\dfrac{\begin{array}{c}\boxed{}\\ \Gamma \vdash B \mid A,B,\Delta\end{array}}{\Gamma \vdash \bot \mid A,B,\Delta}\,(\textsc{pass})}{\Gamma \vdash A \mid B,\Delta}\,(\textsc{act}) \qquad \dfrac{\dfrac{\begin{array}{c}\boxed{}\\ \Gamma \vdash A \mid A,\Delta\end{array}}{\Gamma \vdash \bot \mid A,\Delta}\,(\textsc{pass})}{\Gamma \vdash A \mid \Delta}\,(\textsc{act})$$

The following result is standard and of use in the proofs below.

Lemma 1.10 (WEAKENING AND THINNING FOR $\vdash_{\lambda\mu}$) *The following rules for* weakening *and* thinning *are admissible for* $\vdash_{\lambda\mu}$*:*

$$(Wk):\ \dfrac{\Gamma \vdash M : A \mid \Delta}{\Gamma' \vdash M : A \mid \Delta'}\ (\Gamma \subseteq \Gamma', \Delta \subseteq \Delta') \qquad (Th):\ \dfrac{\Gamma \vdash M : A \mid \Delta}{\Gamma' \vdash M : A \mid \Delta'}\ \begin{array}{l}(\Gamma' = \{\, x{:}B \in \Gamma \mid x \in fv(M)\,\},\\ \ \Delta' = \{\, \alpha{:}B \in \Delta \mid \alpha \in fn(M)\,\})\end{array}$$

Proof: Standard. □

The following soundness result hold.

Theorem 1.11 ([6]) *If $M \to_{\beta\mu} N$, and $\Gamma \vdash_{\lambda\mu} M : A \mid \Delta$, then $\Gamma \vdash_{\lambda\mu} N : A \mid \Delta$.*

This result is in that paper also shown for CBV and CBN-reduction.

Example 1.12 We can represent the proof contraction in $\vdash_{\mathrm{F}}$ of Example 1.5 through:

$$\dfrac{\dfrac{\begin{array}{c}\boxed{}\\ \Gamma \vdash M : A{\to}B \mid \alpha{:}A{\to}B,\Delta\end{array}}{\Gamma \vdash \mu\alpha.[\alpha]M : A{\to}B \mid \Delta}\,(\mu) \quad \begin{array}{c}\boxed{}\\ \Gamma \vdash N : A \mid \Delta\end{array}}{\Gamma \vdash (\mu\alpha.[\alpha]M)\,N : B \mid \Delta}\,(\to\mathrm{E}) \qquad \dfrac{\dfrac{\begin{array}{c}\boxed{}\\ \Gamma \vdash M : A{\to}B \mid \alpha{:}B,\Delta\end{array} \quad \begin{array}{c}\boxed{}\\ \Gamma \vdash N : A \mid \alpha{:}B,\Delta\end{array}}{\Gamma \vdash MN : B \mid \alpha{:}B,\Delta}\,(\to\mathrm{E})}{\Gamma \vdash \mu\alpha.[\alpha]MN : B \mid \Delta}\,(\mu)$$

The general case for this kind of proof contraction can be illustrated by the derivations for the reduction

$$(\mu\alpha.[\beta]\mathsf{C}\lceil\mu\gamma.[\alpha]M\rfloor)\,N \ \to_{\beta\mu}\ \mu\delta.[\beta]\mathsf{C}\lceil\mu\gamma.[\delta]MN\rfloor$$

(where $\beta{:}C \in \Delta$) in Figure 1; the derivation D_2' is in structure equal to D_2, since that is decided by the syntactic structure of the context $\mathsf{C}\lceil\cdot\rfloor$ but contains $\mu\gamma.[\delta]MN$ rather than $\mu\gamma.[\alpha]M$.

The intuition behind the structural rule is given by de Groote [21]: *"in a $\lambda\mu$-term $\mu\alpha.M$ of type $A \to B$, only the subterms named by α are* really *of type $A \to B$ (...); hence, when such a μ-abstraction is applied to an argument, this argument must be passed over to the sub-terms named by α."* Remark that this is accurate, but hides the fact that the naming construction $[\alpha]M$ is actually a (hidden) instance of rule $(\neg\mathrm{E})$, so 'naming' is actually a kind of application.

Inhabiting the proof of Example 1.4 gives the term $\lambda x.\mu\alpha.[\alpha](x(\lambda y.\mu\beta.[\alpha]y))$. In [30], Parigot shows that 'double negation elimination' can be represented in $\lambda\mu$; as suggested above, $\bot$ is added as a pseudo-type to express negation $\neg A$ through $A\to\bot$, as well as contradiction.

Example 1.13 (Double negation elimination in $\lambda\mu$) Double negation elimination is shown in $\vdash_{\mathrm{NI}}$ by the proof on the left (where $\Gamma = \neg\neg C, \neg C$); we can also show this in $\vdash_{\mathrm{F}}$, as in the proof on the right, but since $\vdash_{\mathrm{F}}$ has no rules for negation, we need to add $\bot$ to express it, so write $C\to\bot$ for $\neg C$. Let $\Gamma' = (C\to\bot)\to\bot$:

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma\vdash\neg\neg C}\,(\textsc{ax}) \quad \dfrac{}{\Gamma\vdash\neg C}\,(\textsc{ax})}{\Gamma\vdash\bot}\,(\neg\textsc{e})}{\neg\neg C\vdash C}\,(\textsc{pbc})}{\vdash\neg\neg C\Rightarrow C}\,(\to\textsc{i})
\qquad
\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma'\vdash(C\to\bot)\to\bot\mid C}\,(\textsc{ax}) \quad \dfrac{\dfrac{\dfrac{}{\Gamma',C\vdash C\mid C}\,(\textsc{ax})}{\Gamma',C\vdash\bot\mid C}\,(\mathbf{PASS})}{\Gamma'\vdash C\to\bot\mid C}\,(\to\textsc{i})}{\Gamma'\vdash\bot\mid C}\,(\to\textsc{e})}{\Gamma'\vdash C\mid}\,(\mathbf{ACT})}{\vdash((C\to\bot)\to\bot)\to C\mid\bot}\,(\to\textsc{i})
$$

Notice that the rules $(\mathbf{PASS})$ and $(\mathbf{ACT})$ are not paired, while they are in $\lambda\mu$, and that the assumption $\Gamma\vdash_{\mathrm{NI}}\neg C$ gets replaced by the proof for $\Gamma'\vdash_{\mathrm{F}} C\to\bot\mid C$. Moreover, $(\neg\textsc{e})$ is represented through $(\to\textsc{i})$ and $(\to\textsc{e})$.

Double negation elimination can be represented in $\lambda\mu$ through $\lambda y.\mu\alpha.[\gamma]y\,(\lambda x.\mu\delta.[\alpha]x)$ [30].

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{y{:}(C\to\bot)\to\bot\vdash y:(C\to\bot)\to\bot\mid}\,(\textsc{ax}) \quad \dfrac{\dfrac{\dfrac{}{x{:}C\vdash x:C\mid\delta{:}\bot,\alpha{:}C,\gamma{:}\bot}\,(\textsc{ax})}{x{:}C\vdash\mu\delta.[\alpha]x:\bot\mid\alpha{:}C,\gamma{:}\bot}\,(\mu)}{\vdash\lambda x.\mu\delta.[\alpha]x:C\to\bot\mid\alpha{:}C,\gamma{:}\bot}\,(\to\textsc{i})}{y{:}(C\to\bot)\to\bot\vdash y\,(\lambda x.\mu\delta.[\alpha]x):\bot\mid\alpha{:}C,\gamma{:}\bot}\,(\to\textsc{e})}{y{:}(C\to\bot)\to\bot\vdash\mu\alpha.[\gamma]y\,(\lambda x.\mu\delta.[\alpha]x):C\mid\gamma{:}\bot}\,(\mu)}{\vdash\lambda y.\mu\alpha.[\gamma]y\,(\lambda x.\mu\delta.[\alpha]x):((C\to\bot)\to\bot)\to C\mid\gamma{:}\bot}\,(\to\textsc{i})
$$

This corresponds to the proof in $\vdash_{\mathrm{F}}$ above, but for the fact that extra calls to $(\mathbf{PASS})$ and $(\mathbf{ACT})$ are added inside the calls to (μ), as well as additional names of type $\bot$; notice that because of these extra rules this term is not closed as it has a free name γ.

The proof transformation we hinted at above translates to the following (where $\neg C\in\Gamma$):

$$
\dfrac{\begin{array}{c}\dfrac{\begin{array}{c}\vdots\\ \Gamma\vdash\neg\neg C\end{array} \quad \dfrac{}{\Gamma\vdash\neg C}\,(\textsc{ax})}{\Gamma\vdash\bot}\,(\neg\textsc{e})\\ \vdots\\ \Gamma',\neg C\vdash\bot\end{array}}{\Gamma'\vdash C}\,(\textsc{pbc})
\qquad
\dfrac{\begin{array}{c}\dfrac{\begin{array}{c}\vdots\\ \Gamma\vdash M:(C\to\bot)\to\bot\mid\alpha{:}C,\gamma{:}\bot,\Delta\end{array} \quad \dfrac{\dfrac{\dfrac{}{\Gamma,y{:}C\vdash y:C\mid\delta{:}\bot,\alpha{:}C,\gamma{:}\bot,\Delta}\,(\textsc{ax})}{\Gamma,y{:}C\vdash\mu\delta.[\alpha]y:\bot\mid\alpha{:}C,\gamma{:}\bot,\Delta}\,(\mu)}{\Gamma\vdash\lambda y.\mu\delta.[\alpha]y:C\to\bot\mid\alpha{:}C,\gamma{:}\bot,\Delta}\,(\to\textsc{i})}{\Gamma\vdash M\,(\lambda y.\mu\delta.[\alpha]y):\bot\mid\alpha{:}C,\gamma{:}\bot,\Delta}\,(\to\textsc{e})\\ \vdots\\ \Gamma'\vdash\mathsf{C}\lceil M\,(\lambda y.\mu\delta.[\alpha]y)\rfloor:\bot\mid\alpha{:}C,\gamma{:}\bot,\Delta\end{array}}{\Gamma'\vdash\mu\alpha.[\gamma]\mathsf{C}\lceil M\,(\lambda y.\mu\delta.[\alpha]y)\rfloor:C\mid\gamma{:}\bot,\Delta}\,(\mu)
$$

so Parigot essentially replaces here an instance of the $(\textsc{ax})$ rule for $\Gamma,y{:}\neg C\vdash_{\mathrm{NI}} y:\neg C$ by a derivation for $\Gamma\vdash_{\lambda\mu}\lambda y.\mu\delta.[\alpha]y:C\to\bot\mid\alpha{:}C,\Delta$.[4] It is this what allows for the successful encoding of $\vdash_{\mathrm{NI}}$ in $\lambda\mu$.

We will see this kind of transformation play an important role later in the paper.

It is important to point out that the use of γ in the previous example creates an anomaly. Although it is a logical tautology, the $\lambda\mu$-term that is the witness for $((C\to\bot)\to\bot)\to C$ is *not* a closed term so the proof has an uncanceled assumption. Moreover, terms can have type $\bot$ without being typed with the equivalent of rule $(\neg\textsc{i})$, but using $(\to\textsc{e})$.

Several attempts have been made to rectify this. Parigot not only adds $\bot$ to the language of types in a side remark, but also allows for statements like $\gamma{:}\bot$ to be used without adding

[4] Summers [40] uses $\nu z.[y]\,(zN)$.

them explicitly to the co-context, so does not consider them 'real' assumptions. Ariola and Herbelin [2] define an extension of $\lambda\mu$, adding a special syntax construct $[\mathsf{tp}]M$, where tp acts as a 'continuation constant' and represent the outermost context of the term. In their system, the witness to $((C \to \bot) \to \bot) \to C$ is the term $\lambda y.\mu\alpha.[\mathsf{tp}]y(\lambda x.\mu\delta.[\alpha]x)$.

Another solution would be to detach, syntactically, passivation from activation, so to no longer insist that they strictly follow each other. That is the approach in de Groote and Saurin's $\Lambda\mu$-calculus [21, 38]; there the witness would be $\lambda y.\mu\alpha.y(\lambda x.[\alpha]x)$ which directly inhabits the proof in $\vdash_{\mathrm{F}}$ above. That variant of $\lambda\mu$ better expresses the logic of $\vdash_{\mathrm{F}}$, but one problem with $\Lambda\mu$ is that is not clear if (denotational) semantics can be defined for it, which is possible for $\lambda\mu$ [39, 7]. This is directly related to the fact that a μ-abstraction can now be applied to a term of type $\bot$ that is an application, rather than a term typed (implicitly) with rule $(\neg\mathrm{E})$.

1.3 The $\nu\lambda\mu$-calculus

In [40], Summers makes a strong case for inhabiting the rules of $\vdash_{\mathrm{NI}}$ directly and in full, and defines the calculus $\nu\lambda\mu$ by adding the rules for negation and their syntactic representation to a generalisation of $\lambda\mu$. He thereby extends the syntax with the construct $[M]N$ which is used to represent negation elimination, not just when M is a name, but also when the negated statement on the left is the result of a proof, and allows (μ) to be applied to assumption used on the right in $(\neg\mathrm{E})$. He also removes the distinction between names and variables, and brings all assumptions together in one context.

Definition 1.14 (Syntax of $\nu\lambda\mu$) The *$\nu\lambda\mu$-terms* we consider are defined over variables (Roman characters) by the grammar:

$$M,N \ ::= \ x \mid \lambda x.M \mid MN \mid \nu x.M \mid [M]N \mid \mu x.M$$

Type assignment (see Definition 1.16 below) will naturally allow μ-binding to terms of the shape $[P]Q$, but since $\bot$ is a type, variables and applications can have type $\bot$, allowing $\mu\alpha.y(\lambda x.[\alpha]x)$ to be typed; a term like $\mu x.\lambda y.P$ will not be typeable.

The reduction rules for $\nu\lambda\mu$ in [40] are largely defined, as can be expected, through term substitution as far as the constructors λ and ν are concerned, but contracting a μ redex now becomes more involved than in $\lambda\mu$, for the reasons we discussed in Example 1.2.

Definition 1.15 (Reduction in $\nu\lambda\mu$ [40]) *i*) The auxiliary notion of substitution $\{z{\cdot}N/x\}$[5] is defined inductively over the structure of terms, using the base cases

$$\begin{array}{rcll} x\{z{\cdot}N/x\} & = & \nu z.N & (y \neq x) \\ y\{z{\cdot}N/x\} & = & y & (y \neq x) \\ ([x]M)\{z{\cdot}N/x\} & = & N\{M\{z{\cdot}N/x\}/z\} & \end{array}$$

ii) The reduction rules of $\nu\lambda\mu$ are:

$$\begin{array}{rrcl} \lambda' : & (\lambda x.M)N & \to & \mu y.[\nu x.[y]M]N \\ \nu : & [\nu x.M]N & \to & M\{N/x\} \\ \mu{\to}_1 : & (\mu x.M)N & \to & \mu y.M\{z{\cdot}[y](zN)/x\} \\ \mu{\to}_2 : & N(\mu x.M) & \to & \mu y.M\{z{\cdot}[y](Nz)/x\} \end{array} \qquad \begin{array}{rrcl} \mu\neg_1 : & [\mu x.M]N & \to & M\{z{\cdot}[z]N/x\} \\ \mu\neg_2 : & [N]\mu x.M & \to & M\{z{\cdot}[N]z/x\} \\ \mu\nu : & \nu y.\mu x.M & \to & \nu y.M\{z{\cdot}z/x\} \\ \mu\mu : & \mu y.[\mu x.M] & \to & \mu y.M\{z{\cdot}z/x\} \\ \mu\eta : & \mu x.[x]M & \to & M \quad (x \notin M) \end{array}$$

Evaluation contexts are defined by:

$$\mathsf{C} \ ::= \ \lceil\,\rfloor \mid \lambda x.\mathsf{C} \mid \mathsf{C}M \mid M\mathsf{C} \mid \nu x.\mathsf{C} \mid [\mathsf{C}]M \mid [M]\mathsf{C} \mid \mu x.\mathsf{C}$$

[5] [40] uses a slightly different notation.

(Free, unconstrained) reduction $\rightarrow_{\beta\mu}$ on $\mathcal{L}$-terms is defined through $\mathrm{C}\lceil M \rfloor \rightarrow_{\mathrm{N}} \mathrm{C}\lceil N \rfloor$ if $M \rightarrow N$ using either of the nine rules above.

It is clear that these reduction rules contain the CBV-rules as well in $(\mu{\rightarrow}_2)$ and $(\mu\neg_2)$. Thereby reduction is not confluent; we have a critical pair in the rules $(\mu{\rightarrow}_1)$ and $(\mu{\rightarrow}_2)$: the term $(\mu x.M)\,(\mu y.N)$ is reducible using both, and these reduction steps will (normally) result in different outcomes.

Definition 1.16 (TYPE ASSIGNMENT FOR $\nu\lambda\mu$) *i*) The set of types is defined by the grammar:

$$A,B \ ::= \ \bot \mid \varphi \mid A \rightarrow B \mid \neg A$$

A *context* (of term variables) Γ is defined as before.

ii) The type assignment rules for $\nu\lambda\mu$ are:

$$(\mathrm{AX}): \frac{}{\Gamma,x{:}A \vdash x : A} \qquad (\rightarrow\mathrm{I}): \frac{\Gamma,x{:}A \vdash M : B}{\Gamma \vdash \lambda x.M : A \rightarrow B} \qquad (\rightarrow\mathrm{E}): \frac{\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

$$(\mu): \frac{\Gamma,x{:}\neg A \vdash M : \bot}{\Gamma \vdash \mu x.M : A} \qquad (\neg\mathrm{I}): \frac{\Gamma,x{:}A \vdash M : \bot}{\Gamma \vdash \nu x.M : \neg A} \qquad (\neg\mathrm{E}): \frac{\Gamma \vdash M : \neg A \quad \Gamma \vdash N : A}{\Gamma \vdash [M]N : \bot}$$

We will write $\Gamma \vdash_{\nu\lambda\mu} M : A$ for statements derivable in this system.

Because all inference rules of $\vdash_{\mathbf{NI}}$ are inhabited by term information 'as is', it is immediately clear that all proofs in $\vdash_{\mathbf{NI}}$ have a term representation in $\nu\lambda\mu$.

Example 1.17 In this calculus, the witness for double negation elimination becomes:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{y{:}\neg\neg C,x{:}\neg C \vdash y : \neg\neg C}\,(\mathrm{AX}) \quad \dfrac{}{y{:}\neg\neg C,x{:}\neg C \vdash x : \neg C}\,(\mathrm{AX})}{y{:}\neg\neg C,x{:}\neg C \vdash [y]x : \bot}\,(\neg\mathrm{E})}{y{:}\neg\neg C \vdash \mu x.[y]x : C}\,(\mu)}{\vdash \lambda y.\mu x.[y]x : (\neg\neg C) \rightarrow C}\,(\rightarrow\mathrm{I})$$

The presence of reduction rules $\mu\nu$ and $\mu\mu$ in Definition 1.15 is remarkable, since they do not correspond to proof contractions in a proof system that uses $\bot$ only to represent conflict. Both rule (μ) and $(\neg\mathrm{I})$ are only applicable to a statement of the shape $\Gamma \vdash_{\nu\lambda\mu} M : \bot$; the rule (μ) above them implies an assumption of the shape $\neg\bot$, which is allowed since in [40], $\bot$ is a type, so the following are valid derivations.

$$\dfrac{\dfrac{\Gamma,y{:}A,x{:}\neg\bot \vdash M : \bot}{\Gamma,y{:}A \vdash \mu x.M : \bot}\,(\mu)}{\Gamma \vdash \nu y.\mu x.M : \neg A}\,(\neg\mathrm{I}) \qquad \dfrac{\dfrac{\Gamma,y{:}\neg A,x{:}\neg\bot \vdash M : \bot}{\Gamma,y{:}\neg A \vdash \mu x.M : \bot}\,(\mu)}{\Gamma \vdash \mu y.\mu x.M : A}\,(\mu)$$

Moreover, treating $\bot$ as a type gives that negation is represented in two different ways in $\nu\lambda\mu$. In all, after the choices made by Summers, the calculus is rather too permissive.

Below, we will choose to not treat $\bot$ as a type.

Example 1.18 We can inhabit the proof for $(A \rightarrow B) \rightarrow \neg B \rightarrow \neg A$ in both $\vdash_{\nu\lambda\mu}$ and $\vdash_{\lambda\mu}$. Let $\Gamma = x{:}A{\rightarrow}B, y{:}\neg B, z{:}A$, and $\Gamma' = x{:}A{\rightarrow}B, y{:}B{\rightarrow}\bot, z{:}A$, then we can construct:

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash y : \neg B}\,(\textsc{Ax}) \quad \dfrac{\dfrac{}{\Gamma \vdash x : A\to B}\,(\textsc{Ax}) \quad \dfrac{}{\Gamma \vdash z : A}\,(\textsc{Ax})}{\Gamma \vdash xz : B}\,(\to\textsc{E})}{\Gamma \vdash [y]\,(xz) : \bot}\,(\neg\textsc{E})}{\Gamma\backslash z \vdash \nu z.[y]\,(xz) : \neg A}\,(\neg\textsc{I})}{x{:}A\to B \vdash \lambda y.\nu z.[y]\,(xz) : \neg B \to \neg A}\,(\to\textsc{I})}{\vdash \lambda xy.\nu z.[y]\,(xz) : (A\to B) \to \neg B \to \neg A}\,(\to\textsc{I})
\qquad
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma' \vdash y : B\to\bot \mid} \quad \dfrac{\dfrac{}{\Gamma' \vdash x : A\to B \mid} \quad \dfrac{}{\Gamma' \vdash z : A \mid}}{\Gamma' \vdash xz : B \mid}\,(\to\textsc{E})}{\Gamma' \vdash y(xz) : \bot \mid}\,(\to\textsc{E})}{\Gamma'\backslash z \vdash \lambda z.y(xz) : A\to\bot \mid}\,(\to\textsc{I})}{x{:}A\to B \vdash \lambda yz.y(xz) : (B\to\bot)\to A\to\bot \mid}\,(\to\textsc{I})}{\vdash \lambda xyz.y(xz) : (A\to B)\to(B\to\bot)\to A\to\bot \mid}\,(\to\textsc{I})
$$

Since this property holds in intuitionistic logic, it is no surprise that the co-context is not needed in $\vdash_{\lambda\mu}$.

We can also show the counterpart for $\vdash_{\mathrm{NI}} (\neg B \to \neg A) \to A \to B$. Let $\Gamma = x{:}\neg B \to \neg A, y{:}A, z{:}\neg B$.

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash x : \neg B \to \neg A}\,(\textsc{Ax}) \quad \dfrac{}{\Gamma \vdash z : \neg B}\,(\textsc{Ax})}{\Gamma \vdash xz : \neg A}\,(\to\textsc{E}) \quad \dfrac{}{\Gamma \vdash y : A}\,(\textsc{Ax})}{\Gamma \vdash [xz]\,y : \bot}\,(\neg\textsc{E})}{x{:}\neg B \to \neg A, y{:}A \vdash \mu z.[xz]\,y : B}\,(\mu)}{x{:}\neg B \to \neg A \vdash \lambda y.\mu z.[xz]\,y : A \to B}\,(\to\textsc{I})}{\vdash \lambda xy.\mu z.[xz]\,y : (\neg B \to \neg A) \to A \to B}\,(\to\textsc{I})
$$

We can do a similar thing in $\vdash_{\lambda\mu}$ (with $\Gamma = x{:}(B\to\bot) \to A\to\bot, y{:}A$) :

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash x : (B\to\bot)\to A\to\bot \mid}\,(\textsc{Ax}) \quad \dfrac{\dfrac{\dfrac{}{\Gamma, z{:}B \vdash z : B \mid \delta{:}\bot, \alpha{:}B, \gamma{:}\bot}\,(\textsc{Ax})}{\Gamma, z{:}B \vdash \mu\delta.[\alpha]z : \bot \mid \alpha{:}B, \gamma{:}\bot}\,(\mu)}{\Gamma \vdash \lambda z.\mu\delta.[\alpha]z : B \to \bot \mid \alpha{:}B, \gamma{:}\bot}\,(\to\textsc{I})}{\Gamma \vdash x(\lambda z.\mu\delta.[\alpha]z) : A\to\bot \mid \alpha{:}B, \gamma{:}\bot}\,(\to\textsc{E}) \quad \dfrac{}{\Gamma \vdash y : A \mid \alpha{:}B, \gamma{:}\bot}\,(\textsc{Ax})}{\Gamma \vdash x\,(\lambda z.\mu\delta.[\alpha]z)y : \bot \mid \alpha{:}B, \gamma{:}\bot}\,(\to\textsc{E})}{\Gamma \vdash \mu\alpha.[\gamma]x\,(\lambda z.\mu\delta.[\alpha]z)y : B \mid \gamma{:}\bot}\,(\mu)}{\Gamma\backslash y \vdash \lambda y.\mu\alpha.[\gamma]x\,(\lambda z.\mu\delta.[\alpha]z)y : A \to B \mid \gamma{:}\bot}\,(\to\textsc{I})}{\vdash \lambda xy.\mu\alpha.[\gamma]x\,(\lambda z.\mu\delta.[\alpha]z)y : ((B\to\bot)\to A\to\bot) \to A \to B \mid \gamma{:}\bot}\,(\to\textsc{I})
$$

Notice that the same kind of transformation has been applied to replace the negated assumption $\Gamma \vdash_{\mathcal{L}} z : \neg B$ on the right, and that again the witness for the property is not a closed term in $\lambda\mu$ (γ is free).

2 The $\mathcal{L}$-calculus

We now present the calculus $\mathcal{L}$ we introduce in this paper, which can be seen as a variant of $\lambda\mu$ that gives a Curry-Howard interpretation to the logical system below, which corresponds to $\vdash_{\mathrm{F}}$, extended with negation by treating it as a first-class citizen. Our aim is to fully represent proofs in $\vdash_{\mathrm{NI}}$ in a natural way, but defining a calculus with a notion of reduction that is confluent.

We call this calculus $\mathcal{L}$ in honour of Mendelson's formal axiomatic theory L for the propositional calculus [26]. Adapted to our notation, L is defined through:

Definition 2.1 (L cf. [26]) *i*) (Well formed) formulas are defined through the grammar:

$$A, B \ ::= \ p \mid \neg A \mid A \to B$$

ii) If A, B and C are formulas of L, then the following are axioms of L:

$(A1)$: $A \to B \to A$.

$(A2)$: $(A{\to}B{\to}C){\to}(A{\to}B){\to}A{\to}C$.
$(A3)$: $(\neg B{\to}\neg A){\to}(\neg B{\to}A){\to}B$.

iii) The only rule of inference of L is *modus ponens* $(\to\mathrm{E})$: B is a direct consequence of A and $A{\to}B$.

The first two rules form the axiom-schemes for intuitionistic implicational logic; the third rule renders the system classical. For example, using these three rules it is possible to show $\neg\neg C{\to}C$ (for details, see [26], Lemma 1.11).

The attentive reader will recognise the types of the combinators K and S of Curry's Combinatory Logic [14, 15] in the first two axioms; this is the origin of the Curry-Howard isomorphism [14]. Of course here we follow Church's approach, by defining an extended λ-calculus.

We will base $\mathcal{L}$ on a variant of the system $\vdash_{\mathrm{F}}$ defined below; notice that, because we use negation explicitly, as in $\nu\lambda\mu$ we no longer have to separate the negated formulas from the non-negated ones.

$$(\mathrm{AX}): \frac{}{\Gamma,A\vdash A} \qquad (\to\mathrm{I}): \frac{\Gamma,A\vdash B}{\Gamma\vdash A\to B} \qquad (\to\mathrm{E}): \frac{\Gamma\vdash A\to B \quad \Gamma\vdash A}{\Gamma\vdash B}$$

$$(\neg\mathrm{I}): \frac{\Gamma,A\vdash M}{\Gamma\vdash\neg A} \qquad (\neg\mathrm{E}): \frac{\Gamma\vdash\neg A \quad \Gamma\vdash A}{\Gamma\vdash\bot} \qquad (\mathbf{ACT}): \frac{\Gamma,\neg A\vdash\bot}{\Gamma\vdash A} \qquad (\mathbf{PASS}): \frac{\Gamma,\neg A\vdash A}{\Gamma\vdash\bot}$$

Notice that A in rules $(\mathbf{ACT})$ and $(\mathbf{PASS})$ can be a negated formula. The rule $(\mathbf{PASS})$ could be omitted, since, as before, we can derive:

$$\frac{\dfrac{}{\Gamma,\neg A\vdash\neg A}\,(\mathrm{AX}) \qquad \dfrac{\qquad}{\Gamma,\neg A\vdash A}}{\Gamma\vdash\bot}\,(\neg\mathrm{E})$$

We keep the rule, however, since we want to preserve the fact that in rule $(\mathbf{ACT})$ we only cancel a negated assumption that was used on the left in $(\neg\mathrm{E})$; notice that that characteristic is not expressed in the logic, but will be once we represent the structure of proofs through syntax.

Notation: We write $V^{\mathbf{F}}$ for the set of all finite sequences of elements of V, with ϵ representing the empty sequence, and use the notation $\vec{v}$ for elements of $V^{\mathbf{F}}$.

Definition 2.2 (Syntax of $\mathcal{L}$) *i*) The set of $\mathcal{L}$-*terms* we consider is defined over variables (Roman characters) and names (Greek characters) by the grammar:

$$M,N ::= x \mid \lambda x.M \mid MN \mid \nu x.M \mid [M]N \mid \mu\alpha.M \mid [\alpha]N$$

ii) Recognising ν, λ, and μ as binders, the notion of free and bound variables and names is defined as usual, and we accept Barendregt's convention to keep free and bound variables and names distinct, using (silent) α-conversion whenever necessary. We write $x\in M$ or $\alpha\in M$ if x or α occurs in M, either free of bound, and call a term *closed* if it has no free variables or names.

iii) We write $M\vec{N}$ for the term $MN_1\cdots N_n$ when $\vec{N}=N_1\cdots N_n$ (so not for $M(N_1\cdots N_n)$ as the notation might suggest), and $M\epsilon=M$.

Notice that α is not a term. Since $\bot$ is not a type, type assignment (see Definition 2.5 below) will only allow μ-binding to terms of the shape $[\alpha]Q$ or $[P]Q$, so staying close to $\lambda\mu$.

We will use $\mathcal{L}$ for the set of terms defined above, as well as for the system based on that, including the reduction and type assignment rules. In $\mathcal{L}$, reduction of terms is expressed via three types of implicit substitution. As usual, $M\{N/x\}$ stands for the (instantaneous) substitution of all occurrences of x in M by N. The definition of structural substitution for $\mathcal{L}$ is defined as for $\lambda\mu$ (Definition 1.7), but with small modifications.

Definition 2.3 (STRUCTURAL SUBSTITUTION IN $\mathcal{L}$) *Structural substitution*, $M\{N{\cdot}\gamma/\alpha\}$ and *insertion* $M\{N/\alpha\}$ are defined inductively over terms. We give the main cases:

$$\begin{array}{rclrcl}
([\alpha]M)\{N{\cdot}\gamma/\alpha\} & \triangleq & [\gamma](M\{N{\cdot}\gamma/\alpha\}N) & ([\alpha]M)\{N/\alpha\} & \triangleq & [M]N \\
([\beta]M)\{N{\cdot}\gamma/\alpha\} & \triangleq & [\beta](M\{N{\cdot}\gamma/\alpha\}) \quad (\beta \neq \alpha) & ([\beta]M)\{N/\alpha\} & \triangleq & [\beta](M\{N/\alpha\}) \ (\beta \neq \alpha)
\end{array}$$

We will write $M\{\vec{N}{\cdot}\gamma/\alpha\}$ for $M\{N_1{\cdot}\gamma_1/\alpha\}\{N_2{\cdot}\gamma_2/\gamma_1\}\cdots\{N_n{\cdot}\gamma/\gamma_{n-1}\}$.

We have the following notion of reduction on $\mathcal{L}$.

Definition 2.4 ($\mathcal{L}$ REDUCTION) *i*) The reduction rules of $\mathcal{L}$ are:

$$\begin{array}{lrcll lrcll}
\beta: & (\lambda x.M)N & \to & M\{N/x\} & & \delta: & [\mu\alpha.M]N & \to & M\{N/\alpha\} & \\
\nu: & [\nu x.M]N & \to & M\{N/x\} & & \theta: & \mu\alpha.[\alpha]M & \to & M & (\alpha \notin M) \\
\mu: & (\mu\alpha.M)N & \to & \mu\gamma.M\{N{\cdot}\gamma/\alpha\} & (\gamma \textit{ fresh}) & \rho: & [\beta]\mu\gamma.M & \to & M\{\beta/\gamma\} &
\end{array}$$

Evaluation contexts are defined by:

$$\mathsf{C} \ ::= \ \lceil\,\rfloor \mid \lambda x.\mathsf{C} \mid \mathsf{C}M \mid M\mathsf{C} \mid \nu x.\mathsf{C} \mid [\mathsf{C}]M \mid [M]\mathsf{C} \mid \mu x.\mathsf{C} \mid [\alpha]\mathsf{C}$$

Reduction $\to_{\mathcal{L}}$ on $\mathcal{L}$-terms is defined through $\mathsf{C}\lceil M\rfloor \to_{\mathrm{N}} \mathsf{C}\lceil N\rfloor$ if $M \to N$ using either the β, ν, μ, δ, θ, or ρ-reduction rule. As usual, we will use $\to_{\mathcal{L}}^{=}$ for the reflexive closure, and $\to_{\mathcal{L}}^{*}$ for the reflexive, transitive closure of $\to_{\mathcal{L}}$.

Judgements in $\mathcal{L}$ are of the shape $\Gamma \vdash_{\mathcal{L}} M : A$, where Γ consists of pairs of Greek or Roman characters (the *variables* and *names*) and their types. Type assignment is defined through:

Definition 2.5 (TYPE ASSIGNMENT FOR $\mathcal{L}$) *i*) The set of types $\mathcal{T}_{\mathcal{L}}$ is defined by the grammar:

$$A,B \ ::= \ \varphi \mid A \to B \mid \neg A$$

where '$\to$' associates to the right and '$\neg$' binds stronger than '$\to$'. Notice that $\perp$ is not a type. If $A = \neg B$, we call A a *negated* type, and if $A = \neg B$, but $B \neq \neg C$, we call A a *single negated* type. If $A = \neg\neg B$, we call A a *double negated type*, where B could be a negated type as well.

ii) A *context* (of term variables) Γ is defined as a partial mapping from term variables to types (which can be negated) and names to negated types, denoted as a finite set of *statements* $x{:}A$ and $\alpha{:}\neg B$, such that the *subjects* of the statements are distinct.

We define $\overline{\Gamma}$ through:

$$\begin{array}{rcl}
\overline{\emptyset} & = & \emptyset \\
\overline{\Gamma,x{:}A} & = & \overline{\Gamma},A \\
\overline{\Gamma,\alpha{:}\neg A} & = & \overline{\Gamma},\neg A
\end{array}$$

iii) The type assignment rules for $\mathcal{L}$ are:

$$(\mathrm{AX}): \frac{}{\Gamma,x{:}A \vdash x : A} \qquad (\to\mathrm{I}): \frac{\Gamma,x{:}A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B} \qquad (\to\mathrm{E}): \frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

$$(\neg\mathrm{I}): \frac{\Gamma,x{:}A \vdash M : \perp}{\Gamma \vdash \nu x.M : \neg A} \qquad (\neg\mathrm{E}): \frac{\Gamma \vdash M : \neg A \quad \Gamma \vdash N : A}{\Gamma \vdash [M]N : \perp} \qquad (\mu): \frac{\Gamma,\alpha{:}\neg A \vdash M : \perp}{\Gamma \vdash \mu\alpha.M : A} \qquad (\mathrm{N}): \frac{\Gamma \vdash N : A}{\Gamma,\alpha{:}\neg A \vdash [\alpha]N : \perp}$$

We will write $\Gamma \vdash_{\mathcal{L}} M : A$ for statements derivable in this system.

Notice that, in rule (N), $\alpha{:}\neg A$ is added to the context; this allows for that statement to already occur there. In all the rules where a variable or name is bound, by our variable convention it does not occur in the context in the conclusion. The notation $\overline{\Gamma}$ will be used in Corollary 4.3.

Example 2.6 In this calculus, $\lambda y.\mu\alpha.[]\,[y](\nu x.[\alpha]x)$ is the witness for double negation elimination:

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{y{:}\neg\neg C,\alpha{:}\neg C \vdash y : \neg\neg C}\,(\textsc{ax}) \qquad \dfrac{\dfrac{\dfrac{}{y{:}\neg\neg C,x{:}C \vdash x : C}\,(\textsc{ax})}{y{:}\neg\neg C,x{:}C,\alpha{:}\neg C \vdash [\alpha]x : \bot}\,(\textsc{n})}{y{:}\neg\neg C,\alpha{:}\neg C \vdash \nu x.[\alpha]x : \neg C}\,(\neg\textsc{i})}{y{:}\neg\neg C,\alpha{:}\neg C \vdash [y]\,(\nu x.[\alpha]x) : \bot}\,(\neg\textsc{e})}{y{:}\neg\neg C \vdash \mu\alpha.[y]\,(\nu x.[\alpha]x) : C}\,(\mu)}{\vdash \lambda y.\mu\alpha.[y]\,(\nu x.[\alpha]x) : (\neg\neg C)\to C}\,(\to\textsc{i})
$$

Observe that $\lambda y.\mu\alpha.[y]\,(\nu x.[\alpha]x)$ is a closed term.

It is also straightforward to find untypeable terms. For example, we cannot type a term like $[\lambda x.M]\,N$ since type assignment would require a negated type for $\lambda x.M$, nor $\mu\alpha.\lambda x.M$ since that would require $\bot$ for $\lambda x.M$, nor $\nu y.\lambda x.M$, $\mu\alpha.MN$, $\mu\alpha.\nu x.M$, etc.

It will be clear that, once allowing Greek characters for variables as well, the rule $(\textsc{n})$ is admissible in $\vdash_{\nu\lambda\mu}$, as was the case above for rule $(\textsc{pass})$. Observe that, if $\Gamma,\alpha{:}\neg A \vdash_{\mathcal{L}} M : \bot$, in order for the derivation for $\Gamma \vdash_{\mathcal{L}} \mu\alpha.M : A$ to be used as a subderivation, either $A = B{\to}C$, or $A = \neg B$, for some B and C.

We will now show that types are preserved under reduction. For this we need a weakening result.

Lemma 2.7 (Weakening and thinning for $\vdash_{\mathcal{L}}$) *The following rules are admissible for $\vdash_{\mathcal{L}}$:*

$$
(Wk):\ \dfrac{\Gamma \vdash M : A}{\Gamma' \vdash M : A}\,(\Gamma \subseteq \Gamma') \qquad (Th):\ \dfrac{\Gamma \vdash M : A}{\Gamma' \vdash M : A}\,(\Gamma' = \{\, x{:}B \in \Gamma \mid x \in fv(M)\,\})
$$

Proof: Standard. □

Notice that, by our extension of Barendregt's convention in Definition 1.9, Γ' cannot contain statements for the bound names and variables in M.

Example 2.8 We illustrate the reduction rule δ:

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma,x{:}A \vdash P : \bot}{\Gamma \vdash \nu x.P : \neg A}\,(\neg\textsc{i})}{\Gamma,\alpha{:}\neg\neg A \vdash [\alpha]\,\nu x.P : \bot}\,(\textsc{n})}{\Gamma,\alpha{:}\neg\neg A \vdash M : \bot}}{\Gamma \vdash \mu\alpha.M : \neg A}\,(\mu) \qquad \Gamma \vdash N : A}{\Gamma \vdash [\mu\alpha.M]\,N : \bot}\,(\neg\textsc{e})
\qquad
\dfrac{\dfrac{\dfrac{\Gamma,x{:}A \vdash P : \bot}{\Gamma \vdash \nu x.P : \neg A}\,(\neg\textsc{i}) \qquad \Gamma \vdash N : A}{\Gamma \vdash [\nu x.P]\,N : \bot}\,(\neg\textsc{e})}{\Gamma \vdash M\{N/\alpha\} : \bot}
$$

It might have been more natural, similar to the approach of [40], to define

$$
([\alpha]M)\,\{N/\alpha\} \ \triangleq\ [\nu z.[z]N]\,M
$$

which would have created the subterm $[\nu z.[z]N]\,P$ in the derivation above; however, notice that $[\nu z.[z]N]\,P \to_{\mathcal{L}} [P]\,N$ and that $\{\nu z.[z]N/\alpha\}$ only ever gets applied 'to the left'.

We will now show that type assignment is closed under reduction. First we show results for the three notions of term substitution.

Lemma 2.9 (Substitution lemma) *i) If $\Gamma,x{:}B \vdash_{\mathcal{L}} M : A$ and $\Gamma \vdash_{\mathcal{L}} L : B$, then $\Gamma \vdash_{\mathcal{L}} M\{L/x\} : A$.*

ii) If $\Gamma,\alpha{:}\neg(B\to C) \vdash_{\mathcal{L}} M : A$ and $\Gamma \vdash_{\mathcal{L}} L : B$, then $\Gamma,\gamma{:}\neg C \vdash_{\mathcal{L}} M\{L{\cdot}\gamma/\alpha\} : A$.

iii) If $\Gamma,\alpha{:}\neg\neg B \vdash_{\mathcal{L}} M : A$ and $\Gamma \vdash_{\mathcal{L}} L : B$, then $\Gamma \vdash_{\mathcal{L}} M\{N/\alpha\} : A$.

Proof: *i*) Standard, by induction on the definition of term substitution.

ii) By induction on the definition of structural substitution. All cases follow straightforwardly, except for:

$(([\alpha]N)\{L{\cdot}\gamma/\alpha\} \triangleq [\gamma]N\{L{\cdot}\gamma/\alpha\})$: Then by rule $(\textsc{n})$, $A = \bot$ and $\Gamma,\alpha{:}\neg(B\to C) \vdash_{\mathcal{L}} N : B\to C$. Then, by induction, we have $\Gamma,\gamma{:}\neg C \vdash_{\mathcal{L}} N\{L{\cdot}\gamma/\alpha\} : B\to C$. Since we know that $\Gamma \vdash_{\mathcal{L}} L : B$, we can construct:

$$\dfrac{\dfrac{\Gamma,\gamma{:}\neg C \vdash N\{L{\cdot}\gamma/\alpha\} : B\to C \qquad \dfrac{\Gamma \vdash L : B}{\Gamma,\gamma{:}\neg C \vdash L : B}\,(Wk)}{\Gamma,\gamma{:}\neg C \vdash (N\{L{\cdot}\gamma/\alpha\})\,L : C}\,(\to\textsc{e})}{\Gamma,\gamma{:}\neg C \vdash [\gamma](N\{L{\cdot}\gamma/\alpha\})\,L : \bot}\,(\textsc{n})$$

iii) By induction on the definition of insertion. All cases follow straightforwardly, except for:

$(([\alpha]M)\{N/\alpha\} \triangleq [M]N)$: Then $A = \bot$, and the derivation is of the shape:

$$\dfrac{\Gamma,\alpha{:}\neg\neg B \vdash P : \neg B}{\Gamma,\alpha{:}\neg\neg A \vdash [\alpha]P : \bot}\,(\textsc{n})$$

By induction we have $\Gamma \vdash_{\mathcal{L}} P\{N/\alpha\} : \neg B$, and we can construct:

$$\dfrac{\Gamma \vdash P\{N/\alpha\} : \neg B \qquad \Gamma \vdash N : B}{\Gamma \vdash [P\{N/\alpha\}]N : \bot}\,(\neg\textsc{e})$$

□

Notice that, the structural substitution $\{N{\cdot}\gamma/\alpha\}$ gets performed by building an application with any subterm P that is named α, resulting in $[\gamma]PN$ of type B. Moreover, the insertion $\{N/\alpha\}$ gets performed for typed terms towards a name that has a double negated type, which disappears.

We will now show that type assignment respects reduction:

Theorem 2.10 (Soundness) *If $\Gamma \vdash_{\mathcal{L}} M : A$, and $M \to_{\mathcal{L}} N$, then $\Gamma \vdash_{\mathcal{L}} N : A$.*

Proof: By induction on the definition of $\to_{\mathcal{L}}$, where we focus on the basic reduction rules.

(β): Then $M \equiv (\lambda x.P)\,Q \to_{\textsc{n}} P\{Q/x\} \equiv N$. The derivation for $\Gamma \vdash_{\mathcal{L}} (\lambda x.P)\,Q : A$ is shaped like

$$\dfrac{\dfrac{\Gamma,x{:}B \vdash P : A}{\Gamma \vdash \lambda x.P : B\to A}\,(\to\textsc{i}) \qquad \Gamma \vdash Q : B}{\Gamma \vdash (\lambda x.P)\,Q : A}\,(\to\textsc{e})$$

In particular, we have $\Gamma,x{:}B \vdash_{\mathcal{L}} P : A$ and $\Gamma \vdash_{\mathcal{L}} Q : B$. Then we have $\Gamma \vdash_{\mathcal{L}} P\{Q/x\} : A$ by Lemma 2.9.

(ν): Then $M \equiv [\nu x.P]\,Q \to_{\textsc{n}} P\{Q/x\} \equiv N$. Then $A = \bot$ and derivation for $\Gamma \vdash_{\mathcal{L}} [\nu x.P]\,Q : \bot$ is shaped like

$$\dfrac{\dfrac{\Gamma,x{:}B \vdash P : \bot}{\Gamma \vdash \nu x.P : \neg B}\,(\neg\textsc{i}) \qquad \Gamma \vdash Q : B}{\Gamma \vdash [\nu x.P]\,Q : \bot}\,(\neg\textsc{e})$$

In particular, we have $\Gamma,x{:}B \vdash_{\mathcal{L}} P : \bot$ and $\Gamma \vdash_{\mathcal{L}} Q : B$. Then, by Lemma 2.9, we have $\Gamma \vdash_{\mathcal{L}} P\{Q/x\} : \bot$.

(μ): Then $M \equiv (\mu\alpha.P)\,Q \to_{\textsc{n}} \mu\gamma.P\{Q{\cdot}\gamma/\alpha\} \equiv N$. The derivation for $(\mu\alpha.P)\,Q$ is shaped like

$$\dfrac{\dfrac{\Gamma,\alpha{:}\neg(B\to A)\vdash P:\bot}{\Gamma\vdash \mu\alpha.P:B\to A}\,(\mu) \qquad \Gamma\vdash Q:B}{\Gamma\vdash(\mu\alpha.P)\,Q:A}\,(\to\mathrm{E})$$

In particular, we have $\Gamma,\alpha{:}\neg(B\to A)\vdash_{\mathcal{L}} P:\bot$ and $\Gamma\vdash_{\mathcal{L}} Q:B$. Then by Lemma 2.9, we have $\Gamma,\gamma{:}\neg A\vdash_{\mathcal{L}} P\{Q{\cdot}\gamma/\alpha\}:\bot$, and applying rule (μ) gives the result.

(δ): Then $M\equiv[\mu\alpha.P]\,Q\to_{\mathrm{N}} P\{Q/\alpha\}\equiv N$. Then $A=\bot$ and derivation for $\Gamma\vdash_{\mathcal{L}}[\mu\alpha.P]\,Q:\bot$ is shaped like

$$\dfrac{\dfrac{\Gamma,\alpha{:}\neg\neg B\vdash P:\bot}{\Gamma\vdash \mu\alpha.P:\neg B}\,(\neg\mathrm{I}) \qquad \Gamma\vdash Q:B}{\Gamma\vdash[\mu\alpha.P]\,Q:\bot}\,(\neg\mathrm{E})$$

In particular, we have $\Gamma,\alpha{:}\neg\neg B\vdash_{\mathcal{L}} P:\bot$ and $\Gamma\vdash_{\mathcal{L}} Q:B$. Then, by Lemma 2.9, we have $\Gamma\vdash_{\mathcal{L}} P\{Q/\alpha\}:\bot$.

(θ): Then $M\equiv\mu\alpha.[\alpha]P\to_{\mathrm{N}} P\equiv N$ with $\alpha\notin M$. The derivation for $\mu\alpha.[\alpha]P$ is shaped like

$$\dfrac{\dfrac{\Gamma\vdash P:A}{\Gamma,\alpha{:}\neg A\vdash[\alpha]P:\bot}\,(\mathrm{N})}{\Gamma\vdash\mu\alpha.[\alpha]P:A}\,(\mu)$$

We have $\Gamma\vdash_{\mathcal{L}} P:A$ through a sub-derivation.

(ρ): Then $M\equiv[\beta]\mu\gamma.M\to_{\mathrm{N}} M\{\beta/\gamma\}\equiv N$. The derivation for $[\beta]\mu\gamma.P$ is shaped like

$$\dfrac{\dfrac{\Gamma,\beta{:}\neg B,\gamma{:}\neg B\vdash P:\bot}{\Gamma,\beta{:}\neg B\vdash\mu\gamma.P:B}\,(\mu)}{\Gamma,\beta{:}\neg B\vdash[\beta]\mu\gamma.P:\bot}\,(\mathrm{N})$$

So in particular, replacing all occurrences of γ by β, we obtain $\Gamma,\beta{:}\neg B\vdash_{\mathcal{L}} P\{\beta/\gamma\}:\bot$. $\square$

3 Confluence

In this section we will show that reduction in $\mathcal{L}$ satisfies the Church-Rosser property, *i.e.* is confluent. This property is defined as follows:

Definition 3.1 (Diamond and Church-Rosser Properties [8]) Let R be binary relation on a set V.

i) R satisfies the *diamond property* if for all $t,u,v\in V$, if $t\,R\,u$ and $t\,R\,v$, then there exists $w\in V$ such that $u\,R\,w$ and $v\,R\,w$.

ii) R satisfies the *Church-Rosser property* (is *confluent*) if its reflexive, transitive closure R^* satisfies the diamond property.

This immediately implies that if a relation is confluent, then so is its transitive closure.

The standard approach to showing confluence is that of Tait and Martin-Löf (see [8, 34]) by defining a notion of *parallel reduction* that is based on the standard reduction, which is a reflexive relation defined (in the case of β-reduction) through the rules:

$$\dfrac{}{x\Rightarrow x} \qquad \dfrac{M\Rightarrow M'}{\lambda x.M\Rightarrow\lambda x.M'} \qquad \dfrac{M\Rightarrow M' \quad N\Rightarrow N'}{MN\Rightarrow M'N'} \qquad \dfrac{M\Rightarrow M' \quad N\Rightarrow N'}{(\lambda x.M)\,N\Rightarrow M'\{N'/x\}}$$

By the last rule, '$\Rightarrow$' encompasses '$\to_\beta$'; also, if N reduces to N', then $(\lambda x.M)\,N$ reduces to $M\{N'/x\}$, so all contractions in the various copies of N inside $M\{N/x\}$ are contracted simultaneously when contracting the redex $(\lambda x.M)\,N$; we are even allowed to contract a redex in M, and contracting all these together is considered a *single step* in '$\Rightarrow$'. The proof of confluence for β-reduction then contains of showing that $\Rightarrow$ satisfies the diamond property, and that $\Rightarrow\ = \ \rightarrow_{\beta}^{*}$.

Using this technique, confluence has been claimed for $\lambda\mu$ in [30], but, as noticed in [36, 4], that proof was not complete. The main reason is that the proof overlooks the fact that, perhaps unexpectedly, reduction of one redex can remove another.

Example 3.2 Take the term $(\mu\alpha.[\alpha]\mu\beta.[\alpha]M)\,N$; contracting the outermost μ-redex $(\mu\alpha.\ldots)\,N$ destroys the innermost ρ-redex $\mu\alpha.[\alpha]\mu\beta.[\alpha]M$. The latter is contractable because the sub-term $\mu\beta.[\alpha]M$ is a μ-abstraction:

$$(\mu\alpha.[\alpha]\mu\beta.[\alpha]M)\,N \ \Rightarrow(\rho)\ \ (\mu\alpha.[\alpha]M)\,N$$

This is no longer true after the contraction of the outermost redex:

$$(\mu\alpha.[\alpha]\mu\beta.[\alpha]M)\,N \ \Rightarrow(\mu)\ \ \mu\gamma.[\gamma](\mu\beta.[\gamma]MN)\,N$$

where N gets (also) placed as an argument to $\mu\beta.[\gamma]MN$, creating an application $(\mu\beta.[\gamma]MN)\,N$ which means that the result is no longer a μ-abstraction, thus destroying the ρ-redex. The resulting terms can be joined, but not through a single parallel reduction step, as would be required. So the diamond property does not hold for the standard notion of parallel reduction.

This problem was successfully addressed by Py [36] and later in [4] using a slightly different approach. We will follow the solution of the first here, using the modification of the definition of '$\Rightarrow$' as suggested by Aczel [1].

As in [4, 5, 7], we will not consider the extensional erasure reduction rule

$$\theta:\ \mu\alpha.[\alpha]M \ \rightarrow\ M \quad (\alpha \notin M)$$

Below we will need the property that we can change the order in which the four implicit substitution operations are performed. Notice that we can consider the substitution a binding operation for the variable or name involved, so for example the variable x in $M\{N/x\}\{P/y\}$ can be assumed to not occur in P.

Proposition 3.3 i) a) $M\{N/x\}\{P/y\} = M\{P/y\}\{N\{P/y\}/x\}$.
b) $M\{N/x\}\{P{\cdot}\delta/\alpha\} = M\{P{\cdot}\delta/\alpha\}\{N\{P{\cdot}\delta/\alpha\}/x\}$.
c) $M\{N/x\}\{P/\alpha\} = M\{P/\alpha\}\{N\{P/\alpha\}/x\}$.
d) $M\{N/x\}\{\delta/\alpha\} = M\{\delta/\alpha\}\{N\{\delta/\alpha\}/x\}$.
ii) a) $M\{N{\cdot}\gamma/\beta\}\{P/y\} = M\{P/y\}\{N\{P/y\}{\cdot}\gamma/\beta\}$.
b) $M\{N{\cdot}\gamma/\beta\}\{P{\cdot}\delta/\alpha\} = M\{P{\cdot}\delta/\alpha\}\{N\{P{\cdot}\delta/\alpha\}{\cdot}\delta/\beta\}$.
c) $M\{N{\cdot}\gamma/\beta\}\{P/\alpha\} = M\{P/\alpha\}\{N\{P/\alpha\}{\cdot}\gamma/\beta\}$.
d) $M\{N{\cdot}\gamma/\beta\}\{\delta/\alpha\} = M\{\delta/\alpha\}\{N\{\delta/\alpha\}{\cdot}\gamma/\beta\}$.
iii) a) $M\{N/\beta\}\{P/y\} = M\{P/y\}\{N\{P/y\}/\beta\}$.
b) $M\{N/\beta\}\{P{\cdot}\delta/\alpha\} = M\{P{\cdot}\delta/\alpha\}\{N\{P{\cdot}\delta/\alpha\}/\beta\}$.
c) $M\{N/\beta\}\{P/\alpha\} = M\{P/\alpha\}\{N\{P/\alpha\}/\beta\}$.
d) $M\{N/\beta\}\{\delta/\alpha\} = M\{\delta/\alpha\}\{N\{\delta/\alpha\}/\beta\}$.
iv) a) $M\{\gamma/\beta\}\{P/y\} = M\{P/y\}\{\gamma/\beta\}$.
b) $M\{\gamma/\beta\}\{P{\cdot}\delta/\alpha\} = M\{P{\cdot}\delta/\alpha\}\{\gamma/\beta\}\{P{\cdot}\delta/\alpha\}$.
c) $M\{\gamma/\beta\}\{P/\alpha\} = M\{P/\alpha\}\{\gamma/\beta\}\{P/\alpha\}$.
d) $M\{\gamma/\beta\}\{\delta/\alpha\} = M\{\delta/\alpha\}\{\gamma/\beta\}\{\delta/\alpha\}$.

Proof: Straightforward by induction on the definition of the four substitutions. $\square$

We now define a notion of parallel reduction for $\mathcal{L}$.

Definition 3.4 (Generalised Parallel reduction for $\mathcal{L}$ (cf. [1, 36])) We define *parallel reduction* on terms in $\mathcal{L}$ inductively by the rules:

$$
\begin{array}{llllll}
(1) & \dfrac{}{x \Rightarrow x}\ x & (5) & \dfrac{M \Rightarrow M' \quad N \Rightarrow N'}{MN \Rightarrow M'N'} & (9) & \dfrac{M \Rightarrow \lambda x.M' \quad N \Rightarrow N'}{MN \Rightarrow M'\{N'/x\}} \\[2ex]
(2) & \dfrac{M \Rightarrow M'}{\lambda x.M \Rightarrow \lambda x.M'} & (6) & \dfrac{M \Rightarrow M' \quad N \Rightarrow N'}{[M]N \Rightarrow [M']N'} & (10) & \dfrac{M \Rightarrow \mu\alpha.M' \quad N \Rightarrow N'}{MN \Rightarrow \mu\gamma.M'\{N'\cdot\gamma/\alpha\}}\ (\gamma\ \textit{fresh}) \\[2ex]
(3) & \dfrac{M \Rightarrow M'}{\mu\alpha.M \Rightarrow \mu\alpha.M'} & (7) & \dfrac{M \Rightarrow M'}{[\alpha]M \Rightarrow [\alpha]M'} & (11) & \dfrac{M \Rightarrow \nu x.M' \quad N \Rightarrow N'}{[M]\,N \Rightarrow M'\{N'/x\}} \\[2ex]
(4) & \dfrac{M \Rightarrow M'}{\nu x.M \Rightarrow \nu x.M'} & (8) & \dfrac{M \Rightarrow \mu\alpha.M'}{[\beta]M \Rightarrow M'\{\beta/\alpha\}} & (12) & \dfrac{M \Rightarrow \mu\alpha.M' \quad N \Rightarrow N'}{[M]N \Rightarrow M\{N'/\alpha\}}
\end{array}
$$

We write $M \Rightarrow_{\mathcal{L}} N$ if the statement $M \Rightarrow N$ is derivable using these rules.

It is easy to check that a term parallel reduces to itself, and under parallel reduction a term is considered to be in normal form if it only reduces to itself.

Notice, in particular, the change in the rule based on β-reduction, which changes from

$$\frac{M \Rightarrow M' \quad N \Rightarrow N'}{(\lambda x.M)\,N \Rightarrow M'\{N'/x\}} \quad \text{to} \quad \frac{M \Rightarrow \lambda x.M' \quad N \Rightarrow N'}{MN \Rightarrow M'\{N'/x\}}$$

It is this change that solves the problem mentioned.

Example 3.5 The problem signalled in Example 3.2 does not occur, since the diverging reduction steps

$$\frac{\mu\alpha.[\alpha]\mu\beta.P \Rightarrow \mu\alpha.[\alpha]\mu\beta.P \qquad N \Rightarrow N}{(\mu\alpha.[\alpha]\mu\beta.P)\,N \Rightarrow \mu\gamma.[\gamma](\mu\beta.P\{N\cdot\gamma/\alpha\})N}\,(10) \qquad \frac{\dfrac{\dfrac{\mu\beta.P \Rightarrow \mu\beta.P}{[\alpha]\mu\beta.P \Rightarrow P\{\alpha/\beta\}}\,(7)}{\mu\alpha.[\alpha]\mu\beta.P \Rightarrow \mu\alpha.P\{\alpha/\beta\}}\,(3) \qquad N \Rightarrow N}{(\mu\alpha.[\alpha]\mu\beta.P)\,N \Rightarrow (\mu\alpha.P\{\alpha/\beta\})\,N}\,(5)$$

can be joined:

$$\frac{\dfrac{\dfrac{\mu\beta.P\{N\cdot\gamma/\alpha\} \Rightarrow \mu\beta.P\{N\cdot\gamma/\alpha\} \qquad N \Rightarrow N}{(\mu\beta.P\{N\cdot\gamma/\alpha\})N \Rightarrow \mu\delta.P\{N\cdot\gamma/\alpha\}\{N\cdot\delta/\beta\}}\,(10)}{[\gamma](\mu\beta.P\{N\cdot\gamma/\alpha\})N \Rightarrow P\{N\cdot\gamma/\alpha\}\{N\cdot\delta/\beta\}\{\gamma/\delta\}}\,(7)}{\mu\gamma.[\gamma](\mu\beta.P\{N\cdot\gamma/\alpha\})N \Rightarrow \mu\gamma.P\{N\cdot\gamma/\alpha\}\{N\cdot\delta/\beta\}\{\gamma/\delta\}}\,(3) \qquad \frac{\mu\alpha.P\{\alpha/\beta\} \Rightarrow \mu\alpha.P\{\alpha/\beta\} \qquad N \Rightarrow N}{(\mu\alpha.P\{\alpha/\beta\})\,N \Rightarrow \mu\gamma.P\{\alpha/\beta\}\{N\cdot\gamma/\alpha\}}\,(10)$$

(notice that $\mu\gamma.P\{\alpha/\beta\}\{N\cdot\gamma/\alpha\} = \mu\gamma.P\{N\cdot\gamma/\alpha\}\{N\cdot\delta/\beta\}\{\gamma/\delta\}$).

It is straightforward to show that $\rightarrow^*_{\mathcal{L}}$ is the transitive closure of $\Rightarrow_{\mathcal{L}}$.

Lemma 3.6 $\rightarrow^*_{\mathcal{L}} = \Rightarrow^*_{\mathcal{L}}$.

Proof: First, since $M \Rightarrow_{\mathcal{L}} M$, for all M, by the presence of rules (9), (10), (11), and (12), we have $\rightarrow^{=}_{\mathcal{L}} \subseteq \Rightarrow_{\mathcal{L}}$, so also $\rightarrow^*_{\mathcal{L}} \subseteq \Rightarrow^*_{\mathcal{L}}$. Since in $\Rightarrow_{\mathcal{L}}$ we essentially contract any number of $\rightarrow_{\mathcal{L}}$-redexes in parallel (including zero or just one) we also have that $\Rightarrow_{\mathcal{L}} \subseteq \rightarrow^*_{\mathcal{L}}$. So in particular, $\Rightarrow_{\mathcal{L}}$ is a subset of a relation that is transitive, so its transitive closure is that as well, so $\Rightarrow^*_{\mathcal{L}} \subseteq \rightarrow^*_{\mathcal{L}}$. So $\rightarrow^*_{\mathcal{L}} = \Rightarrow^*_{\mathcal{L}}$. $\square$

The following property expresses that the four kinds of substitution are respected by $\Rightarrow_{\mathcal{L}}$.

Lemma 3.7 (Substitution Lemma) *If $P \Rightarrow_{\mathcal{L}} P'$ and $Q \Rightarrow_{\mathcal{L}} Q'$, then: 1) $P\{Q/z\} \Rightarrow_{\mathcal{L}} P'\{Q'/z\}$, 2) $P\{Q{\cdot}\gamma/z\} \Rightarrow_{\mathcal{L}} P'\{Q'{\cdot}\gamma/z\}$, 3) $P\{Q/\alpha\} \Rightarrow_{\mathcal{L}} P'\{Q'/\alpha\}$, and 4) $P\{\beta/\alpha\} \Rightarrow_{\mathcal{L}} P'\{\beta/\alpha\}$.*

Proof: *i*) By induction on the definition of $\Rightarrow_{\mathcal{L}}$, where we focus on the first parallel reduction.

(1): $P\{Q/z\} = z\{Q/z\} = Q \Rightarrow_{\mathcal{L}} Q' = z\{Q'/z\} = P'\{Q'/z\}$, and
$P\{Q/z\} = y\{Q/z\} = y \Rightarrow_{\mathcal{L}} y = y\{Q'/z\} = P'\{Q'/z\}$ if $y \neq z$.

(9): Then $MN \Rightarrow_{\mathcal{L}} M'\{N'/x\}$ follows from $M \Rightarrow_{\mathcal{L}} \lambda x.M'$ and $N \Rightarrow_{\mathcal{L}} N'$. By induction, we have $M\{Q/z\} \Rightarrow_{\mathcal{L}} (\lambda x.M')\{Q'/z\}$ and $N\{Q/z\} \Rightarrow_{\mathcal{L}} N'\{Q'/z\}$. So we can infer:

$$\frac{M\{Q/z\} \Rightarrow_{\mathcal{L}} (\lambda x.M')\{Q'/z\} \qquad N\{Q/z\} \Rightarrow_{\mathcal{L}} N'\{Q'/z\}}{M\{Q/z\}N\{Q/z\} \Rightarrow_{\mathcal{L}} M'\{Q'/z\}\{N'\{Q'/z\}/x\}}\ (9)$$

By Lemma 3.3, we have $M'\{Q'/z\}\{N'\{Q'/z\}/x\} = M'\{N'/x\}\{Q'/z\}$.

(10): Then $MN \Rightarrow_{\mathcal{L}} \mu\gamma.M'\{N'{\cdot}\gamma/\alpha\}$ follows from $M \Rightarrow_{\mathcal{L}} \mu\alpha.M'$ and $N \Rightarrow_{\mathcal{L}} N'$. By induction, $M\{Q/z\} \Rightarrow_{\mathcal{L}} (\mu\alpha.M')\{Q'/z\}$ and $N\{Q/z\} \Rightarrow_{\mathcal{L}} N'\{Q'/z\}$. So we can infer:

$$\frac{M\{Q/z\} \Rightarrow_{\mathcal{L}} (\mu\alpha.M')\{Q'/z\} \qquad N\{Q/z\} \Rightarrow_{\mathcal{L}} N'\{Q'/z\}}{M\{Q/z\}N\{Q/z\} \Rightarrow_{\mathcal{L}} \mu\gamma.M'\{Q'/z\}\{N'\{Q'/z\}{\cdot}\gamma/\alpha\}}\ (10)$$

By Lemma 3.3, we have $\mu\gamma.M'\{Q'/z\}\{N'\{Q'/z\}{\cdot}\gamma/\alpha\} = \mu\gamma.M'\{N'{\cdot}\gamma/\alpha\}\{Q'/z\}$.

(11): Then $[M]N \Rightarrow_{\mathcal{L}} M'\{N'/x\}$ follows from $M \Rightarrow_{\mathcal{L}} \nu x.M'$ and $N \Rightarrow_{\mathcal{L}} N'$. By induction, $M\{Q/z\} \Rightarrow_{\mathcal{L}} (\nu x.M')\{Q'/z\}$ and $N\{Q/z\} \Rightarrow_{\mathcal{L}} N'\{Q'/z\}$. So we can infer:

$$\frac{M\{Q/z\} \Rightarrow_{\mathcal{L}} (\nu x.M')\{Q'/z\} \qquad N\{Q/z\} \Rightarrow_{\mathcal{L}} N'\{Q'/z\}}{[M\{Q/z\}]N\{Q/z\} \Rightarrow_{\mathcal{L}} M'\{Q'/z\}\{N'\{Q'/z\}/x\}}\ (9)$$

By Lemma 3.3, we have $M'\{Q'/z\}\{N'\{Q'/z\}/x\} = M'\{N'/x\}\{Q'/z\}$.

(12): Then $[M]N \Rightarrow_{\mathcal{L}} M'\{N'/\alpha\}$ follows from $M \Rightarrow_{\mathcal{L}} \mu\alpha.M'$ and $N \Rightarrow_{\mathcal{L}} N'$. By induction, $M\{Q/z\} \Rightarrow_{\mathcal{L}} (\mu\alpha.M')\{Q'/z\}$ and $N\{Q/z\} \Rightarrow_{\mathcal{L}} N'\{Q'/z\}$. So we can infer:

$$\frac{M\{Q/z\} \Rightarrow_{\mathcal{L}} (\mu\alpha.M')\{Q'/z\} \qquad N\{Q/z\} \Rightarrow_{\mathcal{L}} N'\{Q'/z\}}{[M\{Q/z\}]N\{Q/z\} \Rightarrow_{\mathcal{L}} M'\{Q'/z\}\{N'\{Q'/z\}/\alpha\}}\ (12)$$

By Lemma 3.3, we have $M'\{Q'/z\}\{N'\{Q'/z\}/\alpha\} = M'\{N'/\alpha\}\{Q'/z\}$.

The other cases all follow by induction.

ii) , (3) and (4) Very similar. □

The following property expresses the interaction between the syntactic structure of terms and $\Rightarrow_{\mathcal{L}}$.

Proposition 3.8 *i) If $\lambda x.M \Rightarrow_{\mathcal{L}} L$, then $L \equiv \lambda x.N$ and $M \Rightarrow_{\mathcal{L}} N$.*

ii) If $\mu\alpha.M \Rightarrow_{\mathcal{L}} L$, then $L \equiv \mu\alpha.N$ and $M \Rightarrow_{\mathcal{L}} N$.

iii) If $\nu x.M \Rightarrow_{\mathcal{L}} L$, then $L \equiv \nu x.N$ and $M \Rightarrow_{\mathcal{L}} N$.

iv) If $MN \Rightarrow_{\mathcal{L}} L$, then either:

- *a) $L \equiv PQ$ with $M \Rightarrow_{\mathcal{L}} P$ and $N \Rightarrow_{\mathcal{L}} Q$, or*
- *b) $M \Rightarrow_{\mathcal{L}} \lambda x.P$, and $L = P\{Q/x\}$ with $N \Rightarrow_{\mathcal{L}} Q$, or*
- *c) $M \Rightarrow_{\mathcal{L}} \mu\alpha.P$, and $L = \mu\gamma.P\{Q{\cdot}\gamma/\alpha\}$ with $N \Rightarrow_{\mathcal{L}} Q$.*

v) If $[M]N \Rightarrow_{\mathcal{L}} L$, then either:

- *a) $L \equiv [P]Q$ with $M \Rightarrow_{\mathcal{L}} P$ and $N \Rightarrow_{\mathcal{L}} Q$, or*
- *b) $M \Rightarrow_{\mathcal{L}} \nu x.P$, and $L = P\{Q/x\}$ with $N \Rightarrow_{\mathcal{L}} Q$, or*
- *c) $M \Rightarrow_{\mathcal{L}} \mu\alpha.P$, and $L = P\{Q/\alpha\}$ with $N \Rightarrow_{\mathcal{L}} Q$.*

vi) If $[\alpha]M \Rightarrow_{\mathcal{L}} L$, then either:

- *a) $L \equiv [\alpha]P$ with $M \Rightarrow_{\mathcal{L}} P$, or*
- *b) $L \equiv P\{\alpha/\beta\}$ with $M \Rightarrow_{\mathcal{L}} \mu\beta.P$.*

Proof: Straightforward by the definition of $\Rightarrow_{\mathcal{L}}$. $\square$

We now show that $\Rightarrow_{\mathcal{L}}$ satisfies the diamond property. We will write '$P_1 \searrow P_3 \swarrow P_2$' for '$P_1 \Rightarrow_{\mathcal{L}} P_3$ and $P_2 \Rightarrow_{\mathcal{L}} P_3$'.

Theorem 3.9 *If $P_0 \Rightarrow_{\mathcal{L}} P_1$ and $P_0 \Rightarrow_{\mathcal{L}} P_2$ then there exists a P_3 such that $P_1 \searrow P_3 \swarrow P_2$.*

Proof: By induction on the definition of $\Rightarrow$, where we focus on the first parallel reduction. We only show the interesting cases.

(1): Then $P_0 \equiv x \Rightarrow x \equiv P_1$, and $P_2 = x$; take $P_3 = x$ as well.

(2), (3), (4): By induction.

(5): Then $P_0 \equiv M_0 N_0 \Rightarrow M_1 N_1 \equiv P_1$ because $M_0 \Rightarrow M_1$ and $N_0 \Rightarrow N_1$. By Proposition 3.8(*iv*), either:

($P_2 = M_2 N_2$, *with* $M_0 \Rightarrow M_2$ *and* $N_0 \Rightarrow N_2$): By induction there exists M_3, N_3 such that $M_1 \searrow M_3 \swarrow M_2$ and $N_1 \searrow N_3 \swarrow N_2$. Take $P_3 = M_3 N_3$.

($P_2 \equiv M_2\{N_2/x\}$ *with* $M_0 \Rightarrow \lambda x.M_2$ *and* $N_0 \Rightarrow N_2$): By induction there exists M_3, N_3 such that $M_1 \searrow M_3 \swarrow \lambda x.M_2$, and $N_1 \searrow N_3 \swarrow N_2$; by Proposition 3.8(*i*), $M_3 = \lambda x.M_3'$, and $M_2 \Rightarrow M_3'$. By Rule 9, we have $M_1 N_1 \Rightarrow M_3'\{N_3/x\}$, and by Lemma 3.7, we have $M_2\{N_2/x\} \Rightarrow M_3'\{N_3/x\}$.

($P_2 \equiv \mu\gamma.M_2\{N_2.\gamma/\alpha\}$ *with* $M_0 \Rightarrow \mu\alpha.M_2$ *and* $N_0 \Rightarrow N_2$): By induction there exists M_3, N_3 such that $M_1 \searrow M_3 \swarrow \mu\alpha.M_2$, and $N_1 \searrow N_3 \swarrow N_2$; by Proposition 3.8(*ii*), $M_3 = \mu\alpha.M_3'$, and $M_2 \Rightarrow M_3'$. By Rule 10, we have $M_1 N_1 \Rightarrow \mu\gamma.M_3'\{N_3\cdot\gamma/\alpha\}$, and by Lemma 3.7, we have $\mu\gamma.M_2\{N_2\cdot\gamma/\alpha\} \Rightarrow \mu\gamma.M_3'\{N_3\cdot\gamma/\alpha\}$.

(6): Then $P_0 \equiv [M_0]\,N_0 \Rightarrow [M_1]\,N_1 \equiv P_1$ because $M_0 \Rightarrow M_1$ and $N_0 \Rightarrow N_1$. By Proposition 3.8(*v*), either:

($P_2 \equiv [M_2]N_2$ *with* $M_0 \Rightarrow M_2$ *and* $N_0 \Rightarrow N_2$): By induction there exists M_3, N_3 such that $M_1 \searrow M_3 \swarrow M_2$ and $N_1 \searrow N_3 \swarrow N_2$. Take $P_3 = [M_3]\,N_3$.

($P_2 \equiv M_2\{N_2/x\}$ *with* $M_0 \Rightarrow \nu x.M_2$ *and* $N_0 \Rightarrow N_2$): By induction there exists M_3, N_3 such that $M_1 \searrow M_3 \swarrow \nu x.M_2$, and $N_1 \searrow N_3 \swarrow N_2$; by Proposition 3.8(*i*), $M_3 = \nu x.M_3'$, and $M_2 \Rightarrow M_3'$. By Rule (11), we have $[M_1]\,N_1 \Rightarrow M_3'\{N_3/x\}$ and by Lemma 3.7, we have $M_2\{N_2/x\} \Rightarrow M_3'\{N_3/x\}$.

($P_2 = M_2\{N_2/\alpha\}$ *with* $M_0 \Rightarrow \mu\alpha.M_2$ *and* $N_0 \Rightarrow N_2$): By induction there exists M_3, N_3 such that $M_1 \searrow M_3 \swarrow \mu\alpha.M_2$, and $N_1 \searrow N_3 \swarrow N_2$; by Proposition 3.8(*ii*), $M_3 = \mu\alpha.M_3'$, and $M_2 \Rightarrow M_3'$. By Rule (12), we have $[M_1]\,N_1 \Rightarrow M_3'\{N_3/\alpha\}$. By Lemma 3.7, we have $M_2\{N_2/\alpha\} \Rightarrow M_3'\{N_3\cdot\gamma/\alpha\}$.

(7): Then $P_0 \equiv [\beta]M_0 \Rightarrow [\beta]M_1 \equiv P_1$ because $M_0 \Rightarrow M_1$. By Proposition 3.8(*vi*), either:

($P_2 \equiv [\beta]M_2$ *with* $M_0 \Rightarrow M_2$): By induction, there exists M_3 such that $M_1 \searrow M_3 \swarrow M_2$; then by Rule (7) also $[\beta]M_1 \searrow [\beta]M_3 \swarrow [\beta]M_2$.

($P_2 \equiv M_2\{\beta/\alpha\}$ *with* $M_0 \Rightarrow \mu\alpha.M_2$): By induction, there exists M_3 such that $M_1 \searrow M_3 \swarrow \mu\alpha.M_2$; then by Proposition 3.8(*ii*), $M_3 \equiv \mu\alpha.M_3'$, $M_1 \equiv \mu\alpha.M_1'$. By Lemma 3.7 we have that $M_2\{\beta/\alpha\} \Rightarrow M_3'\{\beta/\alpha\}$. Since $\mu\alpha.M_1' \Rightarrow \mu\alpha.M_3'$, by Rule (7) also $[\beta]\mu\alpha.M_1' \Rightarrow M_3'\{\beta/\alpha\}$.

(8): Then $P_0 \equiv [\beta]M_0 \Rightarrow M_1\{\beta/\alpha\} \equiv P_1$ because $M_0 \Rightarrow \mu\alpha.M_1$. By Proposition 3.8(*vi*), either:

($P_2 \equiv [\beta]M_2$ *with* $M_0 \Rightarrow M_2$): By induction, there exists M_3 such that $\mu\alpha.M_1 \searrow M_3 \swarrow M_2$; then by Proposition 3.8(*ii*), $M_3 \equiv \mu\alpha.M_3'$ and $M_2 \equiv \mu\alpha.M_2'$. By Lemma 3.7 we have that $M_1\{\beta/\alpha\} \Rightarrow M_3'\{\beta/\alpha\}$. Since $\mu\alpha.M_2' \Rightarrow \mu\alpha.M_3'$, by Rule (7) also $[\beta]\mu\alpha.M_2' \Rightarrow M_3'\{\beta/\alpha\}$.

($P_2 \equiv M_2\{\beta/\alpha\}$ *with* $M_0 \Rightarrow \mu\alpha.M_2$): By induction, there exists M_3 such that $\mu\alpha.M_1 \searrow M_3 \swarrow \mu\alpha.M_2$; then by Proposition 3.8(*ii*), $M_3 \equiv \mu\alpha.M_3'$, and $M_1 \searrow M_3' \swarrow M_2$. Then by Lemma 3.7, also $M_1\{\beta/\alpha\} \searrow M_3'\{\beta/\alpha\} \swarrow M_2\{\beta/\alpha\}$.

(9): Then $P_0 \equiv M_0 N_0 \Rightarrow M_1\{N_1/x\} \equiv P_1$ because $M_0 \Rightarrow \lambda x.M_1$ and $N_0 \Rightarrow N_1$. By Proposition 3.8(*iv*), either:

($P_2 \equiv M_2 N_2$ *with* $M_0 \Rightarrow M_2$ *and* $N_0 \Rightarrow N_2$): By induction there exists M_3, N_3 such that $\lambda x.M_1 \searrow M_3 \swarrow M_2$, and $N_1 \searrow N_3 \swarrow N_2$; then by Proposition 3.8(*i*), $M_2 \equiv \lambda x.M'_2$ and $M_3 \equiv \lambda x.M'_3$ and $M_1 \searrow M'_3 \swarrow M'_2$. Since $M_2 \Rightarrow \lambda x.M'_3$ and $N_2 \Rightarrow N_3$, by Rule (9), $M_2 N_2 \Rightarrow M'_3\{N_3/x\}$. We have $M_1\{N_1/x\} \Rightarrow M'_3\{N_3/x\}$ by Lemma 3.7.

($P_2 = M_2\{N_2/x\}$ *with* $M_0 \Rightarrow \lambda x.M_2$ *and* $N_0 \Rightarrow N_2$): By induction there exists M_3, N_3 such that $\lambda x.M_1 \searrow M_3 \swarrow \lambda x.M_2$, and $N_1 \searrow N_3 \swarrow N_2$; then by Proposition 3.8(*i*), $M_3 = \lambda x.M'_3$, and $M_1 \searrow M'_3 \swarrow M_2$. We have $M_1\{N_1/x\} \searrow M'_3\{N_3/x\} \swarrow M_2\{N_2/x\}$ by Lemma 3.7.

($P_2 = \mu\gamma.M_2\{N_2{\cdot}\gamma/\alpha\}$ *with* $M_0 \Rightarrow \mu\alpha.M_2$ *and* $N_0 \Rightarrow N_2$): By induction there exists M_3 such that $\lambda x.M_1 \searrow M_3 \swarrow \mu\alpha.M_2$; this is impossible.

(10): Then $P_0 \equiv M_0 N_0 \Rightarrow \mu\gamma.M_1\{N_1{\cdot}\gamma/\alpha\} \equiv P_1$ because $M_0 \Rightarrow \mu\alpha.M_1$ and $N_0 \Rightarrow N_1$. By Proposition 3.8(*iv*), either:

($P_2 \equiv M_2 N_2$ *with* $M_0 \Rightarrow M_2$ *and* $N_0 \Rightarrow N_2$): By induction there exists M_3, N_3 such that $\mu\alpha.M_1 \searrow M_3 \swarrow M_2$, and $N_1 \searrow N_3 \swarrow N_2$; then by Proposition 3.8(*ii*), $M_2 \equiv \mu\alpha.M'_2$ and $M_3 \equiv \mu\alpha.M'_3$ and $M_1 \searrow M'_3 \swarrow M'_2$. Since $M_2 \Rightarrow \mu\alpha.M'_3$ and $N_2 \Rightarrow N_3$, by Rule (10), $M_2 N_2 \Rightarrow \mu\gamma.M'_3\{N_3{\cdot}\gamma/\alpha\}$. We have $\mu\gamma.M_1\{N_1{\cdot}\gamma/\alpha\} \Rightarrow \mu\gamma.M'_3\{N_3{\cdot}\gamma/\alpha\}$ by Lemma 3.7.

($P_2 = M_2\{N_2/x\}$ *with* $M_0 \Rightarrow \lambda x.M_2$ *and* $N_0 \Rightarrow N_2$): By induction there exists M_3 such that $\mu\alpha.M_1 \searrow M_3 \swarrow \lambda x.M_2$; this is impossible.

($P_2 = \mu\gamma.M_2\{N_2{\cdot}\gamma/\alpha\}$ *with* $M_0 \Rightarrow \mu\alpha.M_2$ *and* $N_0 \Rightarrow N_2$): By induction there exists M_3, N_3 such that $\mu\alpha..M_1 \searrow M_3 \swarrow \mu\alpha..M_2$, and $N_1 \searrow N_3 \swarrow N_2$; then by Proposition 3.8(*i*), $M_3 = \mu\alpha..M'_3$, and $M_1 \searrow M'_3 \swarrow M_2$. We have $M_1\{N_1{\cdot}\gamma/\alpha\} \searrow M'_3\{N_3{\cdot}\gamma/\alpha\} \swarrow M_2\{N_3{\cdot}\gamma/\alpha\}$ by Lemma 3.7.

(11): Then $P_0 \equiv [M_0]\,N_0 \Rightarrow M_1\{N_1/x\} \equiv P_1$ because $M_0 \Rightarrow \nu x.M_1$ and $N_0 \Rightarrow N_1$. By Proposition 3.8(*v*), either:

($P_2 \equiv [M_2]N_2$ *with* $M_0 \Rightarrow M_2$ *and* $N_0 \Rightarrow N_2$): Then by induction there exists M_3, N_3 such that $\nu x.M_1 \searrow M_3 \swarrow M_2$, and $N_1 \searrow N_3 \swarrow N_2$; then by Proposition 3.8(*iii*), $M_2 \equiv \nu x.M'_2$ and $M_3 \equiv \nu x.M'_3$ and $M_1 \searrow M'_3 \swarrow M'_2$. Since $M_2 \Rightarrow \nu x.M'_3$ and $N_2 \Rightarrow N_3$, by Rule (11), $[M_2]N_2 \Rightarrow M'_3\{N_3/x\}$. We have $M_1\{N_1/x\} \Rightarrow M'_3\{N_3/x\}$ by Lemma 3.7.

($P_2 = M_2\{N_2/x\}$ *with* $M_0 \Rightarrow \nu x.M_2$ *and* $N_0 \Rightarrow N_2$): By induction there exists M_3, N_3 such that $\nu x.M_1 \searrow M_3 \swarrow \nu x.M_2$, and $N_1 \searrow N_3 \swarrow N_2$; then by Proposition 3.8(*iii*), $M_3 = \nu x.M'_3$, and $M_1 \searrow M'_3 \swarrow M_2$. We have $M_1\{N_1/x\} \searrow M'_3\{N_3/x\} \swarrow M_2\{N_2/x\}$ by Lemma 3.7.

($P_2 = \mu\gamma.M_2\{N_2{\cdot}\gamma/\alpha\}$ *with* $M_0 \Rightarrow \mu\alpha.M_2$ *and* $N_0 \Rightarrow N_2$): By induction there exists M_3 such that $\nu x.M_1 \searrow M_3 \swarrow \mu\alpha.M_2$; this is impossible.

(12): Then $P_0 \equiv [M_0]\,N_0 \Rightarrow M_1\{N_1/\alpha\} \equiv P_1$ because $M_0 \Rightarrow \mu\alpha.M_1$ and $N_0 \Rightarrow N_1$. By Proposition 3.8(*v*), either:

($P_2 \equiv [M_2]N_2$ *with* $M_0 \Rightarrow M_2$ *and* $N_0 \Rightarrow N_2$): Then by induction there exists M_3, N_3 such that $\mu\alpha.M_1 \searrow M_3 \swarrow M_2$, and $N_1 \searrow N_3 \swarrow N_2$; then by Proposition 3.8(*iii*), $M_2 \equiv \mu\alpha.M'_2$ and $M_3 \equiv \mu\alpha.M'_3$ and $M_1 \searrow M'_3 \swarrow M'_2$. Since $M_2 \Rightarrow \mu\alpha.M'_3$ and $N_2 \Rightarrow N_3$, by Rule (11), $[M_2]N_2 \Rightarrow M'_3\{N_3/\alpha\}$. We have $M_1\{N_1/\alpha\} \Rightarrow M'_3\{N_3/\alpha\}$ by Lemma 3.7.

($P_2 = M_2\{N_2/x\}$ *with* $M_0 \Rightarrow \nu x.M_2$ *and* $N_0 \Rightarrow N_2$): By induction there exists M_3 such that $\mu\alpha.M_1 \searrow M_3 \swarrow \nu x.M_2$; this is impossible.

($P_2 = M_2\{N_2/\alpha\}$ *with* $M_0 \Rightarrow \mu\alpha.M_2$ *and* $N_0 \Rightarrow N_2$): By induction there exists M_3, N_3 such that $\mu\alpha.M_1 \searrow M_3 \swarrow \mu\alpha.M_2$, and $N_1 \searrow N_3 \swarrow N_2$; then by Proposition 3.8(*iii*), $M_3 = \mu\alpha.M'_3$, and $M_1 \searrow M'_3 \swarrow M_2$. We have $M_1\{N_1/\alpha\} \searrow M'_3\{N_3/\alpha\} \swarrow M_2\{N_2/\alpha\}$ by Lemma 3.7. $\square$

We can now state our main result.

Theorem 3.10 (Confluence) *Reduction in* $\rightarrow_{\mathcal{L}}$ *is confluent.*

Proof: By Theorem 3.9, we have that $\Rrightarrow_{\mathcal{L}}$ satisfies the diamond property, and by Lemma 3.6 that $\rightarrow^*_{\mathcal{L}}$ is the transitive closure of $\Rrightarrow_{\mathcal{L}}$. Then by Definition 3.1, $\rightarrow_{\mathcal{L}}$ is confluent. $\square$

4 Representing $\vdash_{\text{NI}}$ in $\vdash_{\mathcal{L}}$

In this section we will show that all statements provable in $\vdash_{\text{NI}}$ have a witness in $\vdash_{\mathcal{L}}$. We achieve this result by first defining a mapping for terms from $\nu\lambda\mu$ to $\mathcal{L}$; this will deal with a necessary transformation of derivations when establishing a relation between typeability in $\nu\lambda\mu$ and $\mathcal{L}$. What we use here is the transformation:

$$
\dfrac{\dfrac{\dfrac{\dfrac{\boxed{}}{\Gamma \vdash_{\nu\lambda\mu} M : \neg\neg C} \qquad \dfrac{}{y{:}\neg C \vdash_{\nu\lambda\mu} y : \neg C}\,(\text{AX})}{\Gamma \vdash_{\nu\lambda\mu} [M]y : \bot}\,(\neg\text{E})}{\dfrac{\boxed{}}{\Gamma \vdash_{\nu\lambda\mu} \mathsf{C}\lceil [M]y \rfloor : \bot}}}{\Gamma \vdash_{\nu\lambda\mu} \mu y.\mathsf{C}\lceil [M]y \rfloor : C}\,(\mu)
\quad \text{into} \quad
\dfrac{\dfrac{\dfrac{\dfrac{\boxed{}}{\Gamma \vdash_{\mathcal{L}} M : \neg\neg C} \qquad \dfrac{\dfrac{\dfrac{}{y{:}C \vdash_{\mathcal{L}} y : C}\,(\text{AX})}{y{:}C,\alpha{:}\neg C \vdash_{\mathcal{L}} [\alpha]y : \bot}\,(\text{N})}{\alpha{:}\neg C \vdash_{\mathcal{L}} \nu y.[\alpha]y : \neg C}\,(\neg\text{I})}{\Gamma \vdash_{\mathcal{L}} [M]\,(\nu y.[\alpha]y) : \bot}\,(\neg\text{E})}{\dfrac{\boxed{}}{\Gamma \vdash_{\mathcal{L}} \mathsf{C}\lceil [M]\,(\nu y.[\alpha]y) \rfloor : \bot}}}{\Gamma \vdash_{\mathcal{L}} \mu\alpha.\mathsf{C}\lceil [M]\,(\nu y.[\alpha]y) \rfloor : C}\,(\mu)
$$

Remark that in the first, there is no subterm that has type C, whereas in the second, there is. So we can deal with (PBC) towards an assumption that is not on the left.

Definition 4.1 We define a mapping $\ulcorner \cdot \lrcorner_V : \nu\lambda\mu \rightarrow \mathcal{L}$ inductively over terms.

$$
\begin{array}{rcll}
\ulcorner x \lrcorner_V & = & \nu x.[\alpha]x & (\alpha/x \in V) \\
\ulcorner x \lrcorner_V & = & x & (x \notin V) \\
\ulcorner \mu x.M \lrcorner_V & = & \mu\alpha.\ulcorner M \lrcorner_{V,\alpha/x}
\end{array}
\qquad
\begin{array}{rcl}
\ulcorner \lambda x.M \lrcorner_V & = & \lambda x.\ulcorner M \lrcorner_V \\
\ulcorner MN \lrcorner_V & = & \ulcorner M \lrcorner_V \ulcorner N \lrcorner_V \\
\ulcorner \nu x.M \lrcorner_V & = & \nu x.\ulcorner M \lrcorner_V \\
\ulcorner [M]\,N \lrcorner_V & = & [\ulcorner M \lrcorner_V]\,\ulcorner N \lrcorner_V
\end{array}
$$

and define

$$
\begin{array}{rcll}
\underline{x} & = & \emptyset \\
\underline{\lambda x.M} = \underline{\nu x.M} & = & \underline{M} \\
\underline{MN} = \underline{[M]N} & = & \underline{M} \cup \underline{N} \\
\underline{\mu x.M} & = & \underline{M} \cup \{\alpha/x\} & (\alpha \textit{ fresh})
\end{array}
\qquad
\begin{array}{rcll}
\ulcorner \Gamma, x{:}A \lrcorner_V & = & \ulcorner \Gamma \lrcorner_V, \alpha{:}A & (\alpha/x \in V) \\
\ulcorner \Gamma, x{:}A \lrcorner_V & = & \ulcorner \Gamma \lrcorner_V, x{:}A & (\alpha/x \notin V) \\
\ulcorner \emptyset \lrcorner_V & = & \emptyset
\end{array}
$$

Remark that, if $\alpha/x \in V$ then x was bound under μ, so in a derivation will have a negated type. Notice that *all* occurrences of term variables that occur in V are replaced by $\ulcorner \cdot \lrcorner_V$, even if they appear on the left in $(\neg\text{E})$: that this is not problematic, can be illustrated by the following:

$$
\ulcorner \mu y.[y]\,M \lrcorner_\epsilon \;=\; \mu\alpha.[\ulcorner y \lrcorner_{\alpha/y}]\,\ulcorner M \lrcorner_{\alpha/y} \;=\; \mu\alpha.[\nu y.[\alpha]y]\,\ulcorner M \lrcorner_{\alpha/y} \;\rightarrow_{\mathcal{L}}\; \mu\alpha.[\alpha]\,\ulcorner M \lrcorner_{\alpha/y}
$$

so the substitutions on the left-hand side do not affect the result, but just create a slightly more complicated proof than would be necessary.

We can now show a representation result, which essentially shows that, although the inference rules of $\vdash_{\nu\lambda\mu}$ and $\vdash_{\mathcal{L}}$ differ significantly in their applicability of the rule (μ), which represents the proof rule (PBC), they can witness the same results in $\vdash_{\text{NI}}$. This result does not establish a rule-to-rule mapping of the correspondence between the systems, but states that logical judgements that are provable in one system are also provable in the other. We know that every provable judgement in $\vdash_{\nu\lambda\mu}$ corresponds directly to a provable statement in $\vdash_{\text{NI}}$, and vice versa, and with the correspondence we show here, we get that this also holds between $\vdash_{\mathcal{L}}$ and $\vdash_{\text{NI}}$.

We first establish a relation between typeability in $\nu\lambda\mu$ and $\mathcal{L}$.

Theorem 4.2 *i) If $\Gamma \vdash_{\mathcal{L}} M : A$, then $\Gamma \vdash_{\nu\lambda\mu} M : A$.*

ii) If $\Gamma \vdash_{\nu\lambda\mu} M : A$, and $V = \underline{M}$, then $\ulcorner\Gamma\lrcorner_V \vdash_{\mathcal{L}} \ulcorner M\lrcorner_V : A$.

Proof: *i*) Since, once allowing Greek characters for variables as well, rule (N) can be omitted and $\vdash_{\mathcal{L}}$ is a sub-inference system of $\vdash_{\nu\lambda\mu}$.

ii) By induction on the definition of $\vdash_{\nu\lambda\mu}$.

(AX): Then $\Gamma = \Gamma', x{:}A$; we have two cases:

$(\alpha/x \in V)$: Then $A = \neg B$, $\Gamma = \Gamma', x{:}\neg B$, so $\ulcorner\Gamma\lrcorner_V = \ulcorner\Gamma'\lrcorner_V, \alpha{:}\neg B$ and $\ulcorner x\lrcorner_V = \nu x.[\alpha]x$. We can derive:

$$\dfrac{\dfrac{\dfrac{}{\ulcorner\Gamma'\lrcorner_V, x{:}B \vdash x : B}\,(\mathrm{AX})}{\ulcorner\Gamma'\lrcorner_V, \alpha{:}\neg B, x{:}B \vdash [\alpha]x : \bot}\,(\mathrm{N})}{\ulcorner\Gamma'\lrcorner_V, \alpha{:}\neg B \vdash \nu x.[\alpha]x : \neg B}\,(\neg\mathrm{I})$$

$(x \notin V)$: Then $\Gamma = \Gamma', x{:}A$, so $x{:}A \in \ulcorner\Gamma\lrcorner_V$ and $\ulcorner x\lrcorner_V = x$; the result follows by rule (AX).

(μ): Then $M = \mu x.N$, $\ulcorner M\lrcorner_V = \mu\alpha.\ulcorner N\lrcorner_{V,\alpha/x}$ and the derivation for $\Gamma \vdash_{\nu\lambda\mu} M : A$ is shaped like:

$$\dfrac{\Gamma, x{:}\neg A \vdash N : \bot}{\Gamma \vdash \mu x.N : A}\,(\mu)$$

By induction we have $\ulcorner\Gamma\lrcorner_V, \alpha{:}\neg A \vdash_{\mathcal{L}} \ulcorner N\lrcorner_{V,\alpha/x} : \bot$; the result follows by rule (μ).

$(\to\mathrm{I})$, $(\to\mathrm{E})$, $(\neg\mathrm{I})$, *and* $(\neg\mathrm{E})$: Straightforward by induction. $\square$

Moreover, we now have have that every provable judgement in $\vdash_{\mathrm{NI}}$ can be inhabited in $\vdash_{\mathcal{L}}$.

Corollary 4.3 If $\Gamma \vdash_{\mathrm{NI}} A$, if and only if there exist Γ' and $M \in \mathcal{L}$ such that $\overline{\Gamma'} = \Gamma$ and $\Gamma' \vdash_{\mathcal{L}} M : A$.

We will illustrate the expressiveness of $\vdash_{\mathcal{L}}$.

Example 4.4 We can witness $(\neg B \to \neg A) \to A \to B$ in $\vdash_{\mathcal{L}}$ (where $\Gamma = x{:}\neg B \to \neg A, y{:}A, \alpha{:}\neg B$):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash x : \neg B \to \neg A}\,(\mathrm{AX}) \quad \dfrac{\dfrac{\dfrac{}{\Gamma, z{:}B \vdash z : B}\,(\mathrm{AX})}{\Gamma, z{:}B \vdash [\alpha]z : \bot}\,(\mathrm{N})}{\Gamma \vdash \nu z.[\alpha]z : \neg B}\,(\neg\mathrm{I})}{\Gamma \vdash x\,(\nu z.[\alpha]z) : \neg A}\,(\to\mathrm{E}) \quad \dfrac{}{\Gamma \vdash y : A}\,(\mathrm{AX})}{\Gamma \vdash [x\,(\nu z.[\alpha]z)]\,y : \bot}\,(\neg\mathrm{E})}{x{:}\neg B \to \neg A, y{:}A \vdash \mu\alpha.[x\,(\nu z.[\alpha]z)]\,y : B}\,(\mu)}{x{:}\neg B \to \neg A \vdash \lambda y.\mu\alpha.[x\,(\nu z.[\alpha]z)]\,y : A \to B}\,(\to\mathrm{I})}{\vdash \lambda xy.\mu\alpha.[x\,(\nu z.[\alpha]z)]\,y : (\neg B \to \neg A) \to A \to B}\,(\to\mathrm{I})$$

We can show Mendelson's Axiom 3 in $\vdash_{\mathrm{NI}}$. Let $\Gamma = \neg B {\to} \neg A, \neg B {\to} A$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, \neg B \vdash \neg B {\to} \neg A}\,(\mathrm{AX}) \quad \dfrac{}{\Gamma, \neg B \vdash \neg B}\,(\mathrm{AX})}{\Gamma, \neg B \vdash \neg A}\,(\to\mathrm{E}) \quad \dfrac{\dfrac{}{\Gamma, \neg B \vdash \neg B {\to} A}\,(\mathrm{AX}) \quad \dfrac{}{\Gamma, \neg B \vdash \neg B}\,(\mathrm{AX})}{\Gamma, \neg B \vdash A}\,(\to\mathrm{E})}{\Gamma, \neg B \vdash \bot}\,(\neg\mathrm{E})}{\Gamma \vdash B}\,(\mathrm{PBC})}{\neg B {\to} \neg A \vdash (\neg B {\to} A) {\to} B}\,(\to\mathrm{I})}{\vdash (\neg B {\to} \neg A) {\to} (\neg B {\to} A) {\to} B}\,(\to\mathrm{I})$$

This proof gets represented in $\nu\lambda\mu$ by the term $\lambda xy.\mu z.[xz]\,(yz)$.

Interpreting this into $\mathcal{L}$ gives (where $\Gamma = x{:}\neg B {\to} \neg A, y{:}\neg B {\to} A, \alpha{:}\neg B$):

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{(Ax)}
\UnaryInfC{$\Gamma \vdash x : \neg B \to \neg A$}
\AxiomC{}
\RightLabel{(Ax)}
\UnaryInfC{$\Gamma, z{:}B \vdash z : B$}
\RightLabel{(N)}
\UnaryInfC{$\Gamma, z{:}B \vdash [\alpha]z : \bot$}
\RightLabel{$(\neg\mathrm{I})$}
\UnaryInfC{$\Gamma \vdash \nu z.[\alpha]z : \neg B$}
\RightLabel{$(\to\mathrm{E})$}
\BinaryInfC{$\Gamma \vdash x\,(\nu z.[\alpha]z) : \neg A$}
\AxiomC{}
\RightLabel{(Ax)}
\UnaryInfC{$\Gamma \vdash y : \neg B \to A$}
\AxiomC{}
\RightLabel{(Ax)}
\UnaryInfC{$\Gamma, z{:}B \vdash z : B$}
\RightLabel{(N)}
\UnaryInfC{$\Gamma, z{:}B \vdash [\alpha]z : \bot$}
\RightLabel{$(\neg\mathrm{I})$}
\UnaryInfC{$\Gamma \vdash \nu z.[\alpha]z : \neg B$}
\RightLabel{$(\to\mathrm{E})$}
\BinaryInfC{$\Gamma \vdash y\,(\nu z.[\alpha]z) : A$}
\RightLabel{$(\neg\mathrm{E})$}
\BinaryInfC{$\Gamma \vdash [x\,(\nu z.[\alpha]z)]\,(y\,(\nu z.[\alpha]z)) : \bot$}
\RightLabel{(μ)}
\UnaryInfC{$\Gamma\backslash\alpha \vdash \mu\alpha.[x\,(\nu z.[\alpha]z)]\,(y\,(\nu z.[\alpha]z)) : B$}
\RightLabel{$(\to\mathrm{I})$}
\UnaryInfC{$x{:}\neg B\to\neg A \vdash \lambda y.\mu\alpha.[x\,(\nu z.[\alpha]z)]\,(y\,(\nu z.[\alpha]z)) : (\neg B\to A)\to B$}
\RightLabel{$(\to\mathrm{I})$}
\UnaryInfC{$\vdash \lambda xy.\mu\alpha.[x\,(\nu z.[\alpha]z)]\,(y\,(\nu z.[\alpha]z)) : (\neg B\to\neg A)\to(\neg B\to A)\to B$}
\end{prooftree}
$$

5 Strong normalisation

In this section we shall prove that every term typeable in $\vdash_{\mathcal{L}}$ is strongly normalisable; this will be done using the technique of *reducibility candidates*, as first defined by Girard [19], based on work by Tait [42]. We will follow Parigot's application [33] of the reducibility method, but adapted to negation and applications like $[M]N$.

Definition 5.1 *i*) We use $\mathcal{SN}(M)$ to express that M is strongly normalisable (all reduction paths starting from M are of finite length), and $\mathcal{SN} = \{\, M \in \mathcal{L} \mid \mathcal{SN}(M)\,\}$.

ii) As in [33], we write $V^{\mathbf{F}}$ for the set of all finite sequences of elements of V, with ϵ representing the empty sequence, and use the notation $\vec{v}$ for elements of $V^{\mathbf{F}}$.

Proposition 5.2 The following properties hold of $\mathcal{SN}$:

i) $\mathcal{SN}(xM_1\cdots M_n)$ (with $n \geq 0$) and $\mathcal{SN}(M')$ if and only if $\mathcal{SN}(xM_1\cdots M_nM')$.

ii) If $\mathcal{SN}(M)$, then $\mathcal{SN}(\lambda x.M)$.

iii) If $\mathcal{SN}(M)$, then $\mathcal{SN}(\nu x.M)$.

iv) If $\mathcal{SN}(M)$ then $\mathcal{SN}(\mu\alpha.M)$.

v) If $\mathcal{SN}(Mx)$ then $\mathcal{SN}(M)$.

vi) If $\mathcal{SN}(M\{N/x\}\vec{P})$ and $\mathcal{SN}(N)$, then

vii) If $\mathcal{SN}(M\{N/x\}\vec{P})$ and $\mathcal{SN}(N)$, then $\mathcal{SN}([\nu x.M]\,N\vec{P})$.

viii) If $\mathcal{SN}(M\{N{\cdot}\gamma/\alpha\}\vec{P})$ and $\mathcal{SN}(N)$, then $\mathcal{SN}((\mu\alpha.M)\,N\vec{P})$.

ix) If $\mathcal{SN}(M\{N/\alpha\}\vec{P})$ and $\mathcal{SN}(N)$, then $\mathcal{SN}([\mu\alpha.M]\,N\vec{P})$.

x) If $\mathcal{SN}(M[\beta/\alpha])$, then $\mathcal{SN}([\beta]\,(\mu\alpha.M))$.

The idea is to assign to each type A a set of strongly normalisable terms $\mathit{Red}(A)$, and to show that every term typeable with A is an element of $\mathit{Red}(A)$, and thereby strongly normalisable. Central to the predicate is the notion of functional construction, which states that a set is in $\mathit{Red}(A\to B)$, if it is contained of terms that return terms in $\mathit{Red}(B)$ when applied to terms in $\mathit{Red}(A)$.

The latter is expressed through the following:

Definition 5.3 ([33]) *i*) The *functional construction* $\Rightarrow$: $\wp\mathcal{L} \times \wp\mathcal{L} \to \wp\mathcal{L}$ is defined through:

$$A \Rightarrow B \;\triangleq\; \{\, M \in \mathcal{L} \mid \forall N \in A\ (MN \in B)\,\}$$

ii) $\Rightarrow$ is generalised to $\Rightarrow^{\mathbf{F}}$ through:

$$A \Rightarrow^{\mathbf{F}} B \;\triangleq\; \{\, M \in \mathcal{L} \mid \forall \vec{N} \in A^{\mathbf{F}}\ (M\vec{N} \in B)\,\}$$

Using functional construction, reducibility candidates can be easily defined.

Definition 5.4 ([33]) The set $\mathit{Red} \subseteq \wp\mathcal{L}$ of *reducibility candidates* is inductively defined through:

i) $\mathcal{SN} \in Red$;

ii) If $A \in Red$ and $B \in Red$, then $A \Rightarrow B \in Red$.

We write $Red(A)$ for $A \in Red$.

The next result states that all terms that are reducible in A are also strongly normalisable, and that all variables are reducible in any type.

Lemma 5.5 ([33]) *If $Red(A)$, then $A \subseteq \mathcal{SN}$ and A contains the λ-variables.*

Proof: We prove *1) $A \subseteq \mathcal{SN}$ and 2) for all $\vec{N} \in \mathcal{SN}^{\mathbf{F}}$, $x\vec{N} \in A$* by induction on the definition of *Red*.

i) $(A = \mathcal{SN})$: Immediate.

$(A = B \Rightarrow C)$: Take $M \in A$, then $\forall N \in A\ (MN \in B)$, so by induction (1), $\forall N \in A\ (\mathcal{SN}(MN))$. Take the λ-variable x, then by induction (2), $x \in B$ and therefore $Mx \in C$, so by induction $\mathcal{SN}(Mx)$, and therefore by Proposition 5.2(v) $\mathcal{SN}(M)$.

ii) $(A = \mathcal{SN})$: By Proposition 5.2(i).

$(A = B \Rightarrow C)$: Let $N' \in B$, then $\mathcal{SN}(N')$ by induction (1). Take $\overrightarrow{\mathcal{SN}(N_i)}$, then by Proposition 5.2(i) $\mathcal{SN}(x\vec{N}N')$, and $x\vec{N}N' \in C$ by induction (2). So $x\vec{N} \in A$. □

Lemma 5.6 ([33]) *If $Red(A)$, then there exists $B \subseteq \mathcal{SN}^{\mathbf{F}}$ such that $A = B \Rightarrow^{\mathbf{F}} \mathcal{SN}$.*

Proof: By induction on the definition of *Red*.

$(A = \mathcal{SN})$: Notice that $\mathcal{SN} \triangleq \{\epsilon\} \Rightarrow^{\mathbf{F}} \mathcal{SN}$.

$(A = C \Rightarrow D)$: By induction we have $D = E \Rightarrow^{\mathbf{F}} \mathcal{SN}$ for some E, and therefore $A = F \Rightarrow^{\mathbf{F}} \mathcal{SN}$ where $F = \{ M\vec{N} \mid M \in C, \vec{N} \in E \}$. □

Definition 5.7 For every $A \in Red$, $A^{\perp}$ is defined as the greatest $B \subseteq \mathcal{SN}^{\mathbf{F}}$ such that $A = B \Rightarrow^{\mathbf{F}} \mathcal{SN}$.

Notice that, since $A \subseteq \mathcal{SN}$, if $M \in A$, then $M \in \mathcal{SN}$ and $M\epsilon \in \mathcal{SN}$, so $\epsilon \in A^{\perp}$.

Parigot remarks that membership of ϵ is essential. He says [33] "*It allows to go from an arbitrary reducibility candidate A to $\mathcal{SN}$ and back, without knowing anything about A. This property is used for the case of the rule where one switches from one formula to another. Contrary to the case of a λ where one knows that the type is an arrow, in the one of a μ one has an arbitrary type, but it can be considered as some kind of arrow $A^{\perp} \Rightarrow^{\mathbf{F}} \mathcal{SN}$ whose number of arguments is unknown (possibly zero).*" (notation adapted).

It is perhaps worthwhile to point out that, so far, there is no relation between typeability and reducibility, in that $Red(A)$ does not just contain terms that are typeable with A; for example, the term xx is in $Red(A)$, for any A, since $xx \in \mathcal{SN}$, but is not typeable in $\vdash_{\mathcal{L}}$. However, assume that term M is typeable with A, then we have a derivation that shows $\mathcal{D} : \Gamma \vdash_{\mathcal{L}} M : A$ and for every free variable x in M there will a type B such that $x{:}B \in \Gamma$, and $\mathcal{D}$ contains occurrences of rule (AX) showing $\Gamma' \vdash_{\mathcal{L}} x : B$. We will be capable of proving that then replacing x in M by elements of $Red(B)$, for every free variable of M, creates an element of $Red(A)$ and this will be sufficient for our purposes.

In [33], Parigot shows termination for typeable terms in $\lambda\mu$ enriched with quantification rules; in order to deal with the binding of type variables, he defines a notion of type interpretation that maps type variables onto reducible sets, extended naturally to quantification and (using functional construction) to arrow types. Here we do not need to deal with quantification, but find it convenient to continue on his path.

Definition 5.8 An *interpretation* ξ is a function from type variables to *Red*. Interpretations are extended to arbitrary formulas by:

$$\begin{array}{rcl} \xi(A\to B) & = & \xi(A) \Rightarrow \xi(B) \\ \xi(\neg A) & = & \xi(A) \Rightarrow \xi(\bot) \\ \xi(\bot) & = & \mathcal{SN} \end{array}$$

We shall now prove our strong normalisation result by showing that every term typeable with A is reducible in that type. For this, we need to prove a stronger property: we will now show that if we replace term-variables by reducible terms in a typeable term, then we obtain a reducible term.

Lemma 5.9 (REPLACEMENT LEMMA) • *Let* $\Gamma = \{\, x_1{:}B_1,\ldots,x_n{:}B_n,\alpha_1{:}\neg C_1,\ldots,\alpha_m{:}\neg C_m \,\}$.

- *Let for* $1 \leq i \leq n$, $N_i \in \xi(B_i)$, *and for all* $1 \leq j \leq m$, $\vec{L_j} \in \xi(C_j)^{\perp}$ *if* $C_j = D_j{\to}E_j$, *and* $L_j \in \xi(D_j)$ *if* $C_j = \neg D_j$.
- *Let* $\{Q_j?/\alpha\}$ *stand for* $\{\vec{Q_j}\cdot\gamma_j/\alpha_j\}$ *if* $C_j = (D_j{\to}E_j)$, *or* $\{Q_j/\alpha\}$ *if* $C_j = \neg D_j$.

If $\Gamma \vdash_{\mathcal{L}} M : A$, *then* $M\{\overrightarrow{N_i/x_i}\}\{\overrightarrow{L_i?/\alpha_i}\} \in \xi(A)$.

Proof: By induction on the structure of derivations. We will use $\mathbf{S}$ for $\{\overrightarrow{N_i/x_i}\}\{\overrightarrow{L_i?/\alpha_i}\}$.

(AX): Then $M \equiv x_i$, for some $1 \leq j \leq n$, $B_i = A$, and $M\mathbf{S} \equiv x_i\mathbf{S} \equiv N_i$. From the second assumption we have that $N_i \in \xi(A)$.

($\to$I): Then $M = \lambda x.N$, $A = F{\to}G$ and $\Gamma,x{:}F \vdash N : G$. Let $P \in \xi(F)$, then by Lemma 5.5 $\mathcal{SN}(P)$, and by induction $N\{P/x\}\mathbf{S} \in \xi(G)$. Let $\vec{Q} \in \xi(G)^{\perp}$, then $N\{P/x\}\mathbf{S}\vec{Q} \in \mathcal{SN}$ by Definition 5.7. Then by Proposition 5.2(*vi*) also $(\lambda x.N)P\mathbf{S}\vec{Q} \in \mathcal{SN}$. Therefore, by Definition 5.7, $(\lambda x.N)P\mathbf{S} \in \xi(G)$ and since $\mathbf{S}$ does not affect P, also $(\lambda x.N)\mathbf{S}P \in \xi(G)$; then by Definition 5.4 and 5.8, $(\lambda x.N)\mathbf{S} \in \xi(F{\to}G)$.

($\to$E): Then $M = PQ$ and there exists F such that $\Gamma \vdash P : F{\to}A$ and $\Gamma \vdash Q : F$. By induction, $P\mathbf{S} \in \xi(F{\to}A)$ and $Q\mathbf{S} \in \xi(F)$; by Definition 5.4 and 5.8 we have $P\mathbf{S}Q\mathbf{S} \in \xi(A)$, and $P\mathbf{S}Q\mathbf{S} \equiv (PQ)\mathbf{S}$.

($\neg$I): Then $M \equiv \nu y.P$, $A = \neg F$, and $\Gamma,y{:}F \vdash P : \bot$. Assume $Q \in \xi(F)$, then by induction, $P\mathbf{S}\{Q/y\} \in \xi(\bot)$, so by Definition 5.8, $\mathcal{SN}(P\mathbf{S}\{Q/y\})$. Then by Proposition 5.2(*vii*), we have $\mathcal{SN}([\nu y.P\mathbf{S}]\,Q)$, so by Definition 5.8, $[\nu y.P\mathbf{S}]\,Q \in \xi(\bot)$, so by Definition 5.8 $\nu y.P\mathbf{S} \in \xi(\neg F)$, and $\nu y.P\mathbf{S} \equiv (\nu y.P)\mathbf{S}$.

($\neg$E): Then $A = \bot$, $M \equiv [P]\,Q$, and there exists F such that $\Gamma \vdash P : \neg F$ and $\Gamma \vdash Q : F$. Then, by induction, we have $P\mathbf{S} \in \xi(\neg F)$ and $Q\mathbf{S} \in \xi(F)$. Then by Definition 5.4 and 5.8, $[P\mathbf{S}]\,Q\mathbf{S} \in \xi(\bot)$, and $[P\mathbf{S}]\,Q\mathbf{S} \equiv ([P]\,Q)\mathbf{S}$.

(N): Then $M = [\alpha_j]N$ with $1 \leq j \leq m$, $A = \bot$, and $\Gamma \vdash N : C_j$ with $\alpha{:}\neg C_j \in \Gamma$. By induction, $N\mathbf{S} \in \xi(C_j)$. Now either:

- ($C_j = D_j{\to}E_j$): Notice that $\{\vec{L_j}\cdot\gamma_j/\alpha_j\} \in \mathbf{S}$ and therefore $([\alpha_j]N)\mathbf{S} = [\gamma_j]N\mathbf{S}\vec{L_j}$. We have $\vec{L_j} \in \xi(C_j)^{\perp}$ by assumption, and therefore by Definition 5.7, $N\mathbf{S}\vec{L_j} \in \mathcal{SN}$, so also $[\gamma_j](N\mathbf{S})\vec{L_j} \in \mathcal{SN}$; then by Definition 5.8, $[\gamma_j](N\mathbf{S})\vec{L_j} \in \mathit{Red}\,(\bot)$.
- ($C_j = \neg F$): Now $V_j \in \xi(F)$ by assumption, and therefore by Definition 5.8, $[N\mathbf{S}]\,V_j \in \mathit{Red}\,(\bot)$, and since $\{Q/\alpha\} \in \mathbf{S}$, also $[N\mathbf{S}]\,V_j = ([\alpha]N)\mathbf{S}$.

(μ): Then $M = \mu\beta.N$ and $\Gamma,\beta{:}\neg A \vdash N : \bot$. Now either:

- ($A = D{\to}E$): Let $\vec{Q} \in \xi(A)^{\perp}$, then by induction $N\{\vec{Q}\cdot\gamma/\alpha\}\mathbf{S} \in \xi(\bot)$ and by Definition 5.4 and 5.8, $N\{\vec{Q}\cdot\gamma/\alpha\}\mathbf{S} \in \mathcal{SN}$. Then by Proposition 5.2(*viii*), $(\mu\alpha.N)\mathbf{S}\vec{Q} \in \mathcal{SN}$, so $(\mu\alpha.N)\mathbf{S} \in \xi(A)$.
- ($A = \neg D$): Assume $Q \in \xi(D)$, then by induction $P\mathbf{S}\{Q/\alpha\} \in \xi(\bot)$ and by Definition 5.4 and 5.8, $P\mathbf{S}\{Q/\alpha\} \in \mathcal{SN}$. Then by Proposition 5.2, we have $[\mu\alpha.P\mathbf{S}]\,Q \in \mathcal{SN}$, so by

Definition 5.8 $[\mu\alpha.P\mathbf{S}]\,Q \in \xi(\bot)$, so by Definition 5.4 $(\mu\alpha.P)\mathbf{S} \in \xi(\neg D)$. $\square$

We can now prove the main result.

Theorem 5.10 (STRONG NORMALISATION) *Any term typeable in '$\vdash_{\mathcal{L}}$' is strongly normalisable.*

Proof: Let $\Gamma = x_1{:}B_1,\ldots,x_n{:}B_n,\alpha_1{:}\neg C_1,\ldots,\alpha_m{:}\neg C_m$ such that $\Gamma \vdash_{\mathcal{L}} M : A$. By Lemma 5.5, for all $1 \leq i \leq n$, $x_i \in \xi(B_i)$ and $\epsilon \in \xi(C_j)^{\bot}$. Then, by 5.9, $M\{\overrightarrow{x_i/x_i}\}\{\overrightarrow{\epsilon?/\alpha_j}\} \in \mathit{Red}(A)$; strong normalisation for M then follows from Lemma 5.5. $\square$

6 Principal typing for $\mathcal{L}$

In this section, we will show that we can define a notion of principal typing for $\vdash_{\mathcal{L}}$. This is achieved in the standard way: we define notions of type substitutions and unification, that are used for the definition of the algorithm $pt_{\mathcal{L}}$ that calculates the principal typing for each term typeable in $\vdash_{\mathcal{L}}$.

Substitution is shown to be sound, *i.e.* maps inferable judgements to inferable judgements, and the algorithm is shown to be complete in that all inferable judgements for a term can be constructed from its principal typing.

Definition 6.1 (SUBSTITUTION AND UNIFICATION) *i*) *a*) The *substitution* $(\varphi \mapsto C)$, where φ is a type variable and C a type, is inductively defined[6] by:

$$\begin{array}{lcll}
(\varphi \mapsto C)\ \bot & = & \bot & \\
(\varphi \mapsto C)\ \varphi & = & C & \\
(\varphi \mapsto C)\ \varphi' & = & \varphi' & (\varphi' \neq \varphi) \\
(\varphi \mapsto C)\ A \to B & = & ((\varphi \mapsto C)\ A) \to ((\varphi \mapsto C)\ B) & \\
(\varphi \mapsto C)\ \neg A & = & \neg((\varphi \mapsto C)\ A) &
\end{array}$$

b) If S_1, S_2 are substitutions, then so is $S_1 \circ S_2$, where $S_1 \circ S_2\, A = S_1(S_2\, A)$.

c) $S\Gamma = \{\, x{:}SB \mid x{:}B \in \Gamma \,\} \cup \{\, \alpha{:}SB \mid \alpha{:}B \in \Gamma \,\}$.

d) $S\langle \Gamma, A\rangle = \langle S\Gamma, SA\rangle$.

e) If there exists a substitution S such that $SA = B$, then B is a *(substitution) instance* of A.

f) Id_S is the identity substitution that replaces all type variables by themselves.

ii) Unification of types is defined by:

$$\begin{array}{llllll}
\mathit{unify} & \varphi & \varphi & = & (\varphi \mapsto \varphi) & \\
\mathit{unify} & \varphi & B & = & (\varphi \mapsto B) & (\varphi \textit{ does not occur in } B) \\
\mathit{unify} & A & \varphi & = & \mathit{unify}\ \varphi\ A & \\
\mathit{unify} & (A \to B) & (C \to D) & = & S_2 \circ S_1 & \\
 & & \text{where} & S_1 & = & \mathit{unify}\ A\ C \\
 & & & S_2 & = & \mathit{unify}\ (S_1\, B)\ (S_1\, D) \\
\mathit{unify} & (\neg A) & (\neg C) & = & \mathit{unify}\ A\ C &
\end{array}$$

iii) The operation *unifyC* generalises *unify* to contexts:

[6] All algorithmic definitions in this section are presented in 'functional style', where calls are matched against the alternatives 'top-down', the first match is taken, and the result is undefined in case there is no match.

$$\begin{array}{l}
pt_{\mathcal{L}}\, x \;=\; \langle x{:}\varphi\,;\varphi\rangle \\
\quad \text{where } \varphi \textit{ is fresh} \\
pt_{\mathcal{L}}\, \lambda x.M \;=\; \langle \Pi\,;P\rangle \\
\quad \text{where } \langle \Pi'\,;P'\rangle \;=\; pt_{\mathcal{L}}\, M \\
\qquad\qquad \Pi\,;P \;=\; \begin{cases} \Pi'\backslash x\,; A\to P' & (x{:}A\in\Pi') \\ \Pi'\,;\varphi\to P' & (x\notin\Pi') \end{cases} \\
\qquad\qquad \varphi \textit{ is fresh} \\
pt_{\mathcal{L}}\, MN \;=\; S_2\circ S_1\,\langle \Pi_1\cup\Pi_2\,;\varphi\rangle \\
\quad \text{where } \langle \Pi_1\,;P_1\rangle \;=\; pt_{\mathcal{L}}\, M \\
\qquad\qquad \langle \Pi_2\,;P_2\rangle \;=\; pt_{\mathcal{L}}\, N \\
\qquad\qquad S_1 \;=\; \textit{unify}\ P_1\ P_2\to\varphi \\
\qquad\qquad S_2 \;=\; \textit{unifyC}\ (S_1\,\Pi_1)\ (S_1\,\Pi_2) \\
\qquad\qquad \varphi \textit{ is fresh} \\
pt_{\mathcal{L}}\, \nu x.M \;=\; \langle \Pi\,;P\rangle \\
\quad \text{where } \langle \Pi'\,;\bot\rangle \;=\; pt_{\mathcal{L}}\, M \\
\qquad\qquad \Pi\,;P \;=\; \begin{cases} \Pi'\backslash x\,; \neg A & (x{:}A\in\Pi') \\ \Pi'\,;\neg\varphi & (x\notin\Pi') \end{cases} \\
\qquad\qquad \varphi \textit{ is fresh}
\end{array}$$

$$\begin{array}{l}
pt_{\mathcal{L}}\, [M]N \;=\; S_2\circ S_1\,\langle \Pi_1\cup\Pi_2\,;\bot\rangle \\
\quad \text{where } \langle \Pi_1\,;P_1\rangle \;=\; pt_{\mathcal{L}}\, M \\
\qquad\qquad \langle \Pi_2\,;P_2\rangle \;=\; pt_{\mathcal{L}}\, N \\
\qquad\qquad S_1 \;=\; \textit{unify}\ P_1\ \neg P_2 \\
\qquad\qquad S_2 \;=\; \textit{unifyC}\ (S_1\,\Pi_1)\ (S_1\,\Pi_2) \\
pt_{\mathcal{L}}\, \mu\alpha.M \;=\; \langle \Pi\,;P\rangle \\
\quad \text{where } \langle \Pi'\,;\bot\rangle \;=\; pt_{\mathcal{L}}\, M \\
\qquad\qquad \Pi\,;P \;=\; \begin{cases} \Pi'\backslash \alpha\,; A & (\alpha{:}\neg A\in\Pi') \\ \Pi'\,;\varphi & (\alpha\notin\Pi') \end{cases} \\
\qquad\qquad \varphi \textit{ is fresh} \\
pt_{\mathcal{L}}\, [\alpha]N \;=\; \langle \Pi\,;\bot\rangle \\
\quad \text{where } \langle \Pi'\,;P'\rangle \;=\; pt_{\mathcal{L}}\, N \\
\qquad\qquad \Pi \;=\; \begin{cases} S\Pi' & (\alpha{:}\neg A\in\Pi') \\ \Pi',\alpha{:}\neg P' & (\alpha\notin\Pi') \end{cases} \\
\qquad\qquad S \;=\; \textit{unify}\ A\ P'
\end{array}$$

Figure 2. The algorithm $pt_{\mathcal{L}}$

$$\begin{array}{lcl}
\textit{unifyC}\ (\Gamma_1,x{:}A)\ (\Gamma_2,x{:}B) &=& S_2\circ S_1, \\
\qquad\qquad \text{where } S_1 &=& \textit{unify}\ A\ B \\
\qquad\qquad\qquad S_2 &=& \textit{unifyC}\ (S_1\,\Gamma_1)\ (S_1\,\Gamma_2) \\
\textit{unifyC}\ (\Gamma_1,x{:}A)\ \Gamma_2 &=& \textit{unifyC}\ \Gamma_1\ \Gamma_2 \quad (x\notin\Gamma_2) \\
\textit{unifyC}\ (\Gamma_1,\alpha{:}A)\ (\Gamma_2,\alpha{:}B) &=& S_2\circ S_1, \\
\qquad\qquad \text{where } S_1 &=& \textit{unify}\ A\ B \\
\qquad\qquad\qquad S_2 &=& \textit{unifyC}\ (S_1\,\Gamma_1)\ (S_1\,\Gamma_2) \\
\textit{unifyC}\ (\Gamma_1,\alpha{:}A)\ \Gamma_2 &=& \textit{unifyC}\ \Gamma_1\ \Gamma_2 \quad (\alpha\notin\Gamma_2) \\
\textit{unifyC}\ \emptyset\ \Gamma_2 &=& Id_S
\end{array}$$

This definition specifies *unify* as a partial function; if the side condition '*φ does not occur in B*' fails, no result is returned. So, for example, '*unify* $\varphi\ \varphi\to\varphi$' or '*unify* $(A\to B)\ \neg(C\to D)$' does not return a substitution.

If successful, unification returns the most general unifier, as stated by:

Proposition 6.2 ([37]) *For all A, B: if S_1 is a substitution such that $S_1\,A = S_1\,B$ (so then S_1 is a* unifier *of A and B), then there exist substitutions S_2 and S_3 such that $S_2 = \textit{unify}\ A\ B$ and $S_1 = S_3\circ S_2$.*

Lemma 6.3 (Soundness of substitution) *If $\Gamma\vdash_{\mathcal{L}} M : A$, then $S\Gamma\vdash_{\mathcal{L}} M : SA$.*

Proof: By straightforward induction on the structure of derivations. □

We now define a notion of principal typing for terms of $\mathcal{L}$.

Definition 6.4 The principal typing algorithm for $\vdash_{\mathcal{L}}$ is given in Figure 2.

We can show that the algorithm creates valid judgements:

Lemma 6.5 (Soundness of $pt_{\mathcal{L}}$) *If $pt_{\mathcal{L}}\, M = \langle \Pi\,;P\rangle$, then $\Pi\vdash_{\mathcal{L}} M : P$.*

Proof: By induction on the structure of terms, using Lemma 6.3.

We will now show the main result for $pt_{\mathcal{L}}$, which states that it calculates the most general typeing with respect to type substitution for all terms typeable in $\vdash_{\mathcal{L}}$.

Theorem 6.6 (Completeness of substitution.) *If $\Gamma \vdash_{\mathcal{L}} M : A$, then there exists context Π, type P, and substitution S such that: $pt_{\mathcal{L}}\, M = \langle \Pi\,; P \rangle$, $S\,\Pi \subseteq \Gamma$, and $S\,P = A$.*

Proof: By induction on the structure of terms in $\mathcal{L}$.

$(M \equiv x)$: Then, by rule (AX), $x{:}A \in \Gamma$, and $pt_{\mathcal{L}}\, x = \langle \{x{:}\varphi\}\,; \varphi \rangle$. Take $S = (\varphi \mapsto A)$.

$(M \equiv \lambda x.N)$: Then, by rule $(\to\text{I})$, there are C,D such that $A = C \to D$, and $\Gamma, x{:}C \vdash_{\mathcal{L}} N : D$. Then, by induction, there are Π', P' and S' such that $pt_{\mathcal{L}}\, N = \langle \Pi'\,; P' \rangle$, $S'\,\Pi' \subseteq \Gamma, x{:}C$, and $S\,P' = D$. Then either:

$(x \in fv(N))$: Then $x{:}C' \in \Pi'$, and $pt_{\mathcal{L}}\, \lambda x.N = \langle \Pi' \backslash x; C' {\to} P' \rangle$. Since $S'\,\Pi' \subseteq \Gamma, x{:}C$, in particular $S'\,C' = C$, $S'\,(\Pi' \backslash x) \subseteq \Gamma$, and $S'\,(C' {\to} P') = C \to D$. Take $\Pi = \Pi' \backslash x$, $P = C' {\to} P'$, and $S = S'$.

$(x \notin fv(N))$: Then $pt_{\mathcal{L}}\, \lambda x.N = \langle \Pi'\,; \varphi {\to} P' \rangle$, x does not occur in Π', and let φ not occur in $\langle \Pi'\,; P' \rangle$. Since $S'\,\Pi' \subseteq \Gamma, x{:}C$, in particular $S'\,\Pi' \subseteq \Gamma$. Take $S = S' \circ (\varphi \mapsto C)$, then, since φ does not occur in Π', also $S\Pi' \subseteq \Gamma$. Notice that $S\,(\varphi {\to} P') = C \to D$; take $\Pi = \Pi'$ and $P = \varphi {\to} P'$.

$(M \equiv QR)$: Then, by rule $(\to\text{E})$, there exists a B such that $\Gamma \vdash_{\mathcal{L}} Q : B {\to} A$ and $\Gamma \vdash_{\mathcal{L}} R : B$. By induction, there are S_1, S_2, $\langle \Pi_1\,; P_1 \rangle = pt_{\mathcal{L}}\, Q$ and $\langle \Pi_2\,; P_2 \rangle = pt_{\mathcal{L}}\, R$ (no type variables shared) such that $S_1\,\Pi_1 \subseteq \Gamma$, $S_2\,\Pi_2 \subseteq \Gamma$, $S_1\,P_1 = B {\to} A$ and $S_2\,P_2 = B$. Notice that S_1, S_2 do not interfere. Let φ be a fresh type variable and

$$\begin{array}{rcl} S_u & = & \mathit{unify}\ P_1\ (P_2 {\to} \varphi) \\ S_C & = & \mathit{unifyC}\ (S_u\,\Pi_1)\ (S_u\,\Pi_2) \\ pt_{\mathcal{L}}\, QR & = & S_C \circ S_u\, \langle \Pi_1 \cup \Pi_2\,; \varphi'_1 \cup \Delta'_2 \rangle \end{array}$$

We need to argue that $pt_{\mathcal{L}}\, QR$ is successful: since this can only fail on calls to unification (of P_1 and $P_2 {\to} \varphi$, or in the unification of the contexts), we need to argue that these are successful. Take $S_3 = S_2 \circ S_1 \circ (\varphi \mapsto A)$, then

$$\begin{array}{rcl} S_3\,P_1 & = & B {\to} A, \text{ and} \\ S_3(P_2 {\to} \varphi) & = & B {\to} A. \end{array}$$

so P_1 and $P_2 {\to} \varphi$ have a common instance $B {\to} A$, and by Proposition 6.2, S_u exists.

Notice that we have

$$\begin{array}{rcl} S_3\,\Pi_1 & \subseteq & \Gamma, \text{ and} \\ S_3\,\Pi_2 & \subseteq & \Gamma \end{array}$$

since Π_1 and Π_2 share no type-variables. Since Γ is a context, each term variable has only one type, and therefore S_3 is a unifier for Π_1 and Π_2, so we know that an S_4 exists which extends the substitution that unifies the contexts, even after being changed with S_u, so such that

$$\begin{array}{rcl} S_4(S_u\,\Pi_1) & \subseteq & \Gamma, \text{ and} \\ S_4(S_u\,\Pi_2) & \subseteq & \Gamma. \end{array}$$

So S_4 also unifies $S_u\,\Pi_1$ and $S_u\,\Pi_2$, so by Proposition 6.2 there exists a substitution S_5 such that $S_4 = S_5 \circ S_\Gamma \circ S_u$. Take $S = S_5$.

$(M \equiv \nu x.N)$: Then, by rule $(\neg\text{I})$, there exists C such that $A = \neg C$, and $\Gamma, x{:}C \vdash_{\mathcal{L}} N : \bot$. Then, by induction, there are Π' and S' such that $pt_{\mathcal{L}}\, N = \langle \Pi'\,; \bot \rangle$, and $S'\,\Pi' \subseteq \Gamma, x{:}C$. Then either:

$(x \in fv(N))$: Then $x{:}C' \in \Pi'$, and $pt_{\mathcal{L}}\, \nu x.N = \langle \Pi' \backslash x; \neg C' \rangle$. Since $S'\,\Pi' \subseteq \Gamma, x{:}C$, in particular $S'\,C' = C$, $S'\,(\Pi' \backslash x) \subseteq \Gamma$, and $S'\,(\neg C') = \neg C$. Take $\Pi = \Pi' \backslash x$, $P = \neg C'$, and $S = S'$.

$(x \notin fv(N))$: Then $pt_{\mathcal{L}}\, \lambda x.N = \langle \Pi'; \neg \varphi \rangle$, x does not occur in Π' where φ does not occur in $\langle \Pi' ; P' \rangle$. Since $S' \Pi' \subseteq \Gamma, x{:}C$, in particular $S' \Pi' \subseteq \Gamma$. Take $S = S' \circ (\varphi \mapsto C)$, then, since φ does not occur in Π', also $S\Pi' \subseteq \Gamma$. Notice that $S(\neg\varphi) = \neg C$; take $\Pi = \Pi'$ and $P = \neg\varphi$.

$(M \equiv [Q]R)$: Then $A = \bot$ and by rule $(\neg\mathrm{E})$ there exists a B such that $\Gamma \vdash_{\mathcal{L}} Q : \neg B$ and $\Gamma \vdash_{\mathcal{L}} R : B$. By induction, there are S_1, S_2, $\langle \Pi_1 ; P_1 \rangle = pt_{\mathcal{L}}\, Q$ and $\langle \Pi_2 ; P_2 \rangle = pt_{\mathcal{L}}\, R$ (no type variables shared) such that $S_1 \Pi_1 \subseteq \Gamma$, $S_2 \Pi_2 \subseteq \Gamma$, $S_1 P_1 = \neg B$ and $S_2 P_2 = B$. Notice that S_1, S_2 do not interfere. Let φ be a fresh type variable and

$$\begin{array}{rcl} S_u & = & \mathit{unify}\ P_1\ \neg P_2 \\ S_C & = & \mathit{unifyC}\ (S_u \Pi_1)\ (S_u \Pi_2) \\ pt_{\mathcal{L}}\, QR & = & S_C \circ S_u \langle \Pi_1 \cup \Pi_2 ; \bot \rangle \end{array}$$

As for the case $M = QR$, take $S_3 = S_2 \circ S_1 \circ (\varphi \mapsto A)$, then $S_3 P_1 = \neg B$, and $S_3 P_2 = B$, so P_1 and $\neg P_2$ have a common instance $\neg B$ and S_u exists. Since also $S_3 \Pi_1 \subseteq \Gamma$, and $S_3 \Pi_2 \subseteq \Gamma$, as above an S_4 exists such that $S_4(S_u \Pi_1) \subseteq \Gamma$, and $S_4(S_u \Pi_2) \subseteq \Gamma$ and by Proposition 6.2 there exists a substitution S_5 such that $S_4 = S_5 \circ S_\Gamma \circ S_u$. Take $S = S_5$.

$(M \equiv \mu\alpha.N)$: Then, by rule (μ), $\Gamma, \alpha{:}\neg A \vdash_{\mathcal{L}} N : \bot$. Then, by induction, there are Π' and S' such that $pt_{\mathcal{L}}\, N = \langle \Pi' ; \bot \rangle$, and $S' \Pi' \subseteq \Gamma, \alpha{:}\neg A$. Then either:

$(\alpha \in fv(N))$: Then $\alpha{:}\neg C \in \Pi'$, and $pt_{\mathcal{L}}\, \mu\alpha.N = \langle \Pi' \backslash \alpha ; C \rangle$. Since $S' \Pi' \subseteq \Gamma, \alpha{:}\neg A$, in particular $S' C = A$ and $S' (\Pi' \backslash \alpha) \subseteq \Gamma$. Take $\Pi = \Pi' \backslash \alpha$, $P = C'$, and $S = S'$.

$(\alpha \notin fv(N))$: Then $pt_{\mathcal{L}}\, \mu\alpha.N = \langle \Pi' ; \varphi \rangle$, α does not occur in Π' where φ does not occur in $\langle \Pi' ; P' \rangle$. Since $S' \Pi' \subseteq \Gamma, \alpha{:}\neg A$, in particular $S' \Pi' \subseteq \Gamma$. Take $S = S' \circ (\varphi \mapsto A)$, then, since φ does not occur in Π', also $S\Pi' \subseteq \Gamma$. Notice that $S(\varphi) = A$; take $\Pi = \Pi'$ and $P = \varphi$.

$(M \equiv [\alpha]N)$: Then $A = \bot$ and by rule (N) there exists a B such that $\alpha{:}\neg B \in \Gamma$ and $\Gamma \vdash_{\mathcal{L}} N : B$. By induction, there exists S_1, $\langle \Pi' ; P' \rangle = pt_{\mathcal{L}}\, N$ such that $S_1 \Pi' \subseteq \Gamma$, and $S_1 P' = B$. Then either:

$(\alpha \in fv(N))$: Let $\alpha{:}\neg C \in \Pi'$; take $S_2 = \mathit{unify}\ C\ P'$, then $pt_{\mathcal{L}}\, [\alpha]N = \langle S_2 \Pi' ; \bot \rangle$, and $\alpha{:}\neg S_2 C \in S_2 \Pi'$. Since $\alpha{:}\neg C \in \Pi'$ and $S_1 \Pi' \subseteq \Gamma$, we have that $S_1 \neg C = \neg B$ and $S_1 P' = B$, so S_2 is successful and there exists S_3 such that $S_1 = S_3 \circ S_2$, so $S_1 \Pi' = S_3(S_2 \Pi') \subseteq \Gamma$. Take $S = S_3$.

$(\alpha \notin fv(N))$: Then $pt_{\mathcal{L}}\, [\alpha]N = \langle \Pi', \alpha{:}\neg P' ; \bot \rangle$; take $S = S_1$. ☐

This last result shows the practicality of our notion of type assignment.

Conclusion and Future Work

We have presented $\mathcal{L}$ as an extension of Parigot's $\lambda\mu$-calculus by adding negation as a type constructor with the aim of representing proofs in $\vdash_{\mathrm{NI}}$, Classical Logic with implication, negation, and Proof by Contradiction. We gained a more expressive calculus, that no longer represents $\neg A$ through $A \rightarrow \bot$, but, more importantly, negation elimination is no longer represented by application, and negation introduction not by λ-abstraction, but through new syntactic constructs that represent negation introduction and elimination directly, thus getting a more faithful representation of proofs in $\vdash_{\mathrm{NI}}$.

We defined a notion of reduction that extends $\lambda\mu$'s logical and structural reduction rules with two new reduction rules, one dealing with a $(\neg\mathrm{I})-(\neg\mathrm{E})$-pair, the other when Proof by Contradiction gets applied agains an assumption that has a double negated type. We showed that type assignment is sound, in that assignable types are preserved under reduction. Using, as suggested by Py, Aczel's generalisation of Tait and Martin-Löf's notion of parallel reduction, we showed that reduction is confluent.

By it's nature, not all proofs in $\vdash_{\mathrm{NI}}$ can be represented in $\mathcal{L}$, but we have shown a completeness result in that all propositions that can be shown in $\vdash_{\mathbf{NI}}$ have a witness in $\mathcal{L}$. Following Parigot, using Girard's approach of reducibility candidates, we have shown that all typeable terms are strongly normalisable, and that type assignment for $\mathcal{L}$ enjoys the principal typing property.

In all, $\mathcal{L}$ satisfies all the properties that can be demanded for a calculus claiming to represent proofs and proof contraction in $\vdash_{\mathbf{NI}}$. By representing negation directly, it also severely enlarges the size of the set of **NEF** terms (a syntactic notion, not dependent on assigned types) to those *really* not representing negation. All these are important issues for the creation of theorem provers for Classical Logic.

Our motivation for our work was two-fold: enlarge the set of **NEF**-terms, and the fact that (implicative) $\lambda\mu$ does not fully represent negation. It is now fair to ask: "Is $\mathcal{L}$ the finished product?" In other words, can all logical connectors be expressed in $\mathcal{L}$? Although it is well known that conjunction $A \wedge B$ can be expressed through $\neg(A \to \neg B)$ and disjunction $A \vee B$ through $\neg A \to B$, this is not enough, since it does not answer the question of representability. In particular, is seems that rule $(\vee E)$ cannot be represented in $\vdash_{\mathbf{NI}}$, since that would require a combination of a limited version of *LEM* and $(\vee E)$ through the rule

$$\frac{\Gamma, A \vdash C \quad \Gamma, \neg A \vdash C}{\Gamma \vdash C}$$

We will leave this issue for future work.

Acknowledgements

I am very grateful to David Davies for asking me: "How does $\lambda\mu$ deal with negation?"

References

[1] P. Aczel. A general Church-Rosser theorem. Technical report, University Of Manchester, 1978.

[2] Z.M. Ariola and H. Herbelin. Minimal Classical Logic and Control Operators. In J.C.M. Baeten, J.K. Lenstra, J. Parrow, and G.J. Woeginger, editors, *Proceedings of Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003*, volume 2719 of *Lecture Notes in Computer Science*, pages 871–885. Springer Verlag, 2003.

[3] Z.M. Ariola, H. Herbelin, and A. Sabry. A Proof-Theoretic Foundation of Abortive Continuations. In *Proceedings of Higher-Order and Symbolic Computation, 2007*, pages 403–429, 2007.

[4] K. Baba, S. Hirokawa, and K. Fujita. Parallel Reduction in Type Free Lambda-mu-Calculus. In *Computing: The Australasian Theory Symposium (CATS 2001)*, volume 42 of *Electronic Notes in Theoretical Computer Science*, pages 52–66, 2001.

[5] S. van Bakel. Characterisation of Normalisation Properties for $\lambda\mu$ using Strict Negated Intersection Types. *ACM Transactions on Computational Logic*, 19, 2018.

[6] S. van Bakel. Exception Handling and Classical Logic. In E. Komendantskaya, editor, *Proceedings of the 21st International Symposium on Principles and Practice of Programming Languages, PPDP 2019, Porto, Portugal, October 7-9, 2019*, pages 21:1–21:14. ACM, 2019.

[7] S. van Bakel, F. Barbanera, and U. de'Liguoro. Intersection Types for the $\lambda\mu$-calculus. *Logical Methods in Computer Science*, 141(1), 2018.

[8] H. Barendregt. *The Lambda Calculus: its Syntax and Semantics*. North-Holland, Amsterdam, revised edition, 1984.

[9] Edwin Brady. Idris, a general-purpose dependently typed programming language: Design and implementation. *Journal of Functional Programming*, 23(5):552–593, September 2013.

[10] L.E.J. Brouwer. *Over de Grondslagen der Wiskunde*. PhD thesis, Faculteit der Wiskunde en Informatica, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands, 1907.

[11] L.E.J. Brouwer. De onbetrouwbaarheid der logische principes. *Tijdschrift voor Wijsbegeerte*, 2:152–158, 1908.

[12] L.E.J. Brouwer. Unreliability of the Logical Principles. In A. Heyting, editor, *Collected Works 1. Philosophy and Foundations of Mathematics*. North-Holland, Amsterdam, 1975.

[13] The Coq Development Team. *The Coq Reference Manual, Release 8.13.0*, 2021.

[14] H.B. Curry. Functionality in Combinatory Logic. In *Proc. Nat. Acad. Sci. U.S.A*, volume 20, pages 584–590, 1934.
[15] H.B. Curry and R. Feys. *Combinatory Logic*, volume 1. North-Holland, Amsterdam, 1958.
[16] D. Davies, S. van Bakel, and N. Wu. Candid: A Dependently Typed Programming Language with Control Operators for Classical Logic. Submitted, 2021.
[17] M. Felleisen and R Hieb. The revised report on the syntactic theories of sequential control and state. *Theoretical Computer Science*, 103(2), 1992.
[18] G. Gentzen. Untersuchungen über das Logische Schliessen. *Mathematische Zeitschrift*, 39(2):176–210 and 405–431, 1935.
[19] J.-Y. Girard. Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types. In J. Fenstad, editor, *2nd Scandinavian Logic Symposium*, pages 63–92. North Holland, 1971.
[20] T. Griffin. A formulae-as-types notion of control. In *Proceedings of the 17th Annual ACM Symposium on Principles of Programming Languages, Orlando (Fla., USA)*, pages 47–58, 1990.
[21] Ph. de Groote. On the Relation between the $\lambda\mu$-Calculus and the Syntactic Theory of Sequential Control. In *Proceedings of 5th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'94)*, volume 822 of *Lecture Notes in Computer Science*, pages 31–43. Springer Verlag, 1994.
[22] H. Herbelin. On the Degeneracy of Sigma-Types in Presence of Computational Classical Logic. In P. Urzyczyn, editor, *Typed Lambda Calculi and Applications, 7th International Conference, TLCA 2005, Nara, Japan, April 21-23, 2005, Proceedings*, volume 3461 of *Lecture Notes in Computer Science*, pages 209–220. Springer, 2005.
[23] H. Herbelin. A Constructive Proof of Dependent Choice, Compatible with Classical Logic. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*, pages 365–374. IEEE Computer Society, 2012.
[24] W.A. Howard. The Formula-as-Types Notion of Construction. In J.P. Seldin and J.R. Hindley, editors, *To H. B. Curry, Essays in Combinatory Logic, Lambda-Calculus and Formalism*, pages 479–490. Academic press, New York, 1980, 1980.
[25] P. Martin-Löf. *Intuitionistic type theory*, volume 1 of *Studies in proof theory*. Bibliopolis, 1984.
[26] E. Mendelson. *Introduction to Mathematical Logic*. Chapman and Hall, London, and J. Wiley, New York, 1964.
[27] É. Miquey, X. Montillet, and G. Munch-Maccagnoni. Dependent Type Theory in Polarised Sequent Calculus. Draft, 2020.
[28] U. Norell. *Towards a practical programming language based on dependent type theory*. PhD thesis, Department of Computer Science and Engineering, Chalmers University of Technology, SE-412 96 Göteborg, Sweden, September 2007.
[29] C.-H.L. Ong and C.A. Stewart. A Curry-Howard foundation for functional computation with control. In *Proceedings of the 24th Annual ACM Symposium on Principles Of Programming Languages*, pages 215–227, 1997.
[30] M. Parigot. An algorithmic interpretation of classical natural deduction. In *Proceedings of 3rd International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'92)*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201. Springer Verlag, 1992.
[31] M. Parigot. Classical Proofs as Programs. In *Kurt Gödel Colloquium*, pages 263–276, 1993. Presented at TYPES Workshop, at Băstad, June 1992.
[32] M. Parigot. Strong Normalization for Second Order Classical Natural Deduction. In *Proceedings of Eighth Annual IEEE Symposium on Logic in Computer Science, Montreal, Canada*, pages 39–46, 1993.
[33] M. Parigot. Proofs of Strong Normalisation for Second Order Classical Natural Deduction. *Journal of Symbolic Logic*, 62(4):1461–1479, December 1997.
[34] F. Pfenning. A Proof of the Church-Rosser Theorem and its Representation in a Logical Framework. Technical Report CMU-CS-92-186, Carnegie Mellon University, Pittsburgh, 1992.
[35] D. Prawitz. *Natural Deduction, A Proof-Theoretical Study*. Almqvist & Wiksell, 1965.
[36] W. Py. *Confluence en $\lambda\mu$-calcul*. Thèse de doctorat, Université de Savoie, 1998.
[37] J.A. Robinson. A Machine-Oriented Logic Based on Resolution Principle. *Journal of the ACM*, 12(1):23–41, 1965.
[38] A. Saurin. On the Relations between the Syntactic Theories of $\lambda\mu$-Calculi. In M. Kaminski and S. Martini, editors, *Computer Science Logic, 22nd International Workshop (CSL'08), Bertinoro, Italy*, volume 5213 of *Lecture Notes in Computer Science*, pages 154–168. Springer Verlag, September 16-19 2008.
[39] Th. Streicher and B. Reus. Classical logic: Continuation Semantics and Abstract Machines. *Journal of Functional Programming*, 11(6):543–572, 1998.
[40] A.J. Summers. *Curry-Howard Term Calculi for Gentzen-Style Classical Logic*. PhD thesis, Imperial

College London, 2008.
[41] M.E. Szabo, editor. *The Collected Papers of Gerhard Gentzen*. Studies in Logic and the Foundations of Mathematics. North-Holland, 1969.
[42] W. Tait. Intensional Interpretations of Functionals of Finite Type I. *Journal of Symbolic Logic*, 32(2):198–212, 1967.
[43] P. Wadler. Propositions as types. *Communications of the ACM*, 58(12):75–84, 2015.