Stabilization—An Alternative to Double-Negation Translation for Classical Natural Deduction

Ralph Matthes
matthes@informatik.uni-muenchen.de
Lehr- und Forschungseinheit für theoretische Informatik
Institut für Informatik der Universität München
Oettingenstraße 67, D-80538 München

January 19, 2004

Abstract

A new proof of strong normalization of Parigot's second-order $\lambda\mu$-calculus is given by a reduction-preserving embedding into system F (second-order polymorphic λ-calculus). The main idea is to use the least stable supertype for any type. These non-strictly positive inductive types and their associated iteration principle are available in system F, and allow to give a translation vaguely related to CPS translations (corresponding to Kolmogorov's double-negation embedding of classical logic into intuitionistic logic). However, they simulate Parigot's μ-reductions whereas CPS translations hide them.

As a major advantage, this embedding does not use the idea of reducing stability ($\neg\neg A \to A$) to that for atomic formulae. Therefore, it even extends to positive fixed-point types.

The article expands on "Parigot's Second-Order $\lambda\mu$-Calculus and Inductive Types" (Conference Proceedings TLCA 2001, Springer LNCS 2044) by the author.

1 Introduction

If $\neg\neg A \to A$ is provable, then A is called stable. Stability of all A is the key characteristic of classical logic. In the framework of intuitionistic logic, classical features can be added in several ways. We concentrate on "reductio ad absurdum" (RAA), i. e., proof by contradiction: If falsity is provable from the negation of A, then A is provable.

Natural deduction systems can be formulated—via the Curry-Howard-isomorphism—as lambda calculi. Here, we are interested in second-order logic, hence in second-order (polymorphic) lambda-calculus, also called system F.

System F is a term rewrite system, consequently also its extension by RAA comes with rewrite rules: we may speak about operationalized classical logic. The most well known such system is Parigot's $\lambda\mu$-calculus, and it is strongly normalizing [Par93, Par97], i. e., admits no infinite reduction sequences starting with a typable term. Its μ-reduction rules describe how RAA for composite types can be reduced to RAA for the constituent types. In the case of implication, this corresponds to the fact that stability of $A \to B$ follows intuitionistically from stability of B.

In [Mat01], an extension of system F by a least stable supertype $\sharp A$—called the stabilization of A—for any type A, has been studied for the purpose of reproving strong normalization of Parigot's second-order $\lambda\mu$-calculus.

In the present article, this approach is refined to also cover $\lambda\mu$-calculus in the formulation of de Groote [dG94]. Also typing issues are considered, in the sense that typing à la Church is used instead of fully type-decorated terms. An important part of this article is the comparison with other approaches to prove strong normalization of second-order classical natural deduction from that of system F. For direct proofs of strong normalization through logical predicates—even capturing sums/disjunction with permutative/commuting conversions—see the account in [Mat04].

In a nutshell, [Mat01] has been extended

- to Church typing instead of full type annotations (with subtle problems leading to the notion of "refined embedding"—Definition 2—and associated substitution rules),
- to $\lambda\mu$-calculus in de Groote's formulation (with $\perp$ being a regular type),
- by a comparison of stabilization with double negation,
- by a motivation of μ-reduction rules and a presentation of the method by Joly [Jol97],
- by an explanation why translations fail with Curry-style typing or just with plain classical natural deduction (without Parigot's "special substitution"),
- by a discussion of the recent criticism of CPS-translations by Nakazawa and Tatsuta [NT03].

The article is structured as follows: In section 2, system F and second-order classical natural deduction are defined. Section 3 contains the definition of the system of $\mathsf{I}\sharp$ with the least stable supertype $\sharp A$ for every type A and its impredicative encoding in F. It also contains a comparison of $\sharp A$ and $\neg\neg A$. Section 4 is the central part: It is first shown that the more naive system $\mathsf{F}^{\neg}$ of classical natural deduction cannot be embedded into F, then $\lambda\mu$-calculus is defined and successfully embedded in $\mathsf{I}\sharp$. Plenty of discussion material is given in section 5: After a presentation of Joly's embedding, which is based on the idea of reducing stability to that for atomic formulae (= type variables), an extension to fixed-points is given where Joly's method cannot be seen to work—unlike our stabilization. Even least fixed-points are treated. Finally, the article by Nakazawa and Tatsuta is discussed under the heading "Embeddings in Continuation-Passing Style?".

Acknowledgments to Andreas Abel and Klaus Aehlig for fruitful discussions on the subject.

2 Second-Order Natural Deduction

In this article, natural deduction is always represented by the help of lambda terms. For second-order natural deduction, the basic—intuitionistic—system is F [Gir72], which is defined in the first part of this section. Mostly, notation is given that is used throughout the article.

In the second part, F is extended by "reductio ad absurdum" (RAA), which is again just a fixing of notation. In addition, an argument is given why F itself fails to be "classical", and the reduction rules for RAA—which are a variation on those of $\lambda\mu$-calculus [Par92]—are motivated by the reduction of stability for composite formulas/types to that for atoms/type variables.

2.1 System F

System F is the polymorphic lambda-calculus whose normalization has first been shown by Girard [Gir72]. In a proof-theoretic reading, F provides proof terms for intuitionistic second-order propositional logic in natural deduction style, and also considers proof transformation rules. Normalization for second-order systems does not yield the subformula property but nevertheless guarantees logical consistency and allows, e. g., to conclude that F is not classical—see below. In the programming language perspective, normalization is the calculation of values for terms inhabiting a concrete datatype. And F is a (functional) programming language in which datatypes can be represented [BB85].

We define system F and recall some of its basic metatheoretic properties.

Types. (Denoted by uppercase letters.) Metavariable X ranges over an infinite set TV of type variables.

$$A, B, C, D \quad ::= \quad X \mid \bot \mid A \to B \mid \forall X.A$$

The occurrences of X in A are bound in $\forall X.A$. We identify types whose de Bruijn representations coincide. Let $\mathsf{FV}(A)$ be the set of free type variables in A. The type $\bot$ just serves as a constant in F. It is used to define $\neg A := A \to \bot$. Note that intuitionistic falsum is already present in the pure system: $\forall X.X$ has "ex falsum quodlibet" by trivial universal elimination.

Terms. (Denoted by lowercase letters.) The metavariable x ranges over an infinite set of term variables.

$$r, s, t \quad ::= \quad x \mid \lambda x^A.t \mid r\,s \mid \Lambda X.t \mid r\,B$$

The occurrences of x in t are bound in $\lambda x^A.t$. Also, the occurrences of type variable X in t are bound in $\Lambda X.t$ We identify α-equivalent terms, that is, terms whose bound type and term variables are consistently renamed. Capture-avoiding substitution of a term s for a variable x in term r is denoted by $r[x := s]$. Analogously, we define type substitution $A[X := B]$ and substitution $t[X := B]$ of type B for type variable X in term t. We think of terms as parse trees according to the above grammar rules and insert parentheses in case of ambiguities, especially around the application $r\,s$. But we will omit them as much as possible and assume that application associates to the left. Hence, we write $rs_1 \dots s_n$ for the parenthesized term $(\dots(rs_1)\dots s_n)$—also for types instead of terms s_i. We abbreviate $\mathsf{id}^A := \lambda x^A.x$, the identity on A.

Contexts. They are finite sets of pairs $(x : A)$ of term variables and types and will be denoted by Γ. Term variables in a context Γ are assumed to be distinct, and the notation $\Gamma, x : A$ shall indicate $\Gamma \cup \{(x : A)\}$ where x does not occur in Γ. Write $\mathsf{FV}(\Gamma) := \bigcup_{(x:A)\in\Gamma} \mathsf{FV}(A)$.

Well-typed terms. $\Gamma \vdash t : A$ is inductively defined by

$$\frac{(x{:}A) \in \Gamma}{\Gamma \vdash x : A} \qquad \frac{\Gamma, x{:}A \vdash t : B}{\Gamma \vdash \lambda x^A.t : A \to B} \qquad \frac{\Gamma \vdash r : A \to B \qquad \Gamma \vdash s : A}{\Gamma \vdash r\,s : B}$$

as for simply-typed lambda-calculus, and by the two rules for the universal quantifier whose use is traced in the terms (unlike formulations in a style à la Curry):

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \Lambda X.t : \forall X.A} \text{ if } X \notin \mathsf{FV}(\Gamma) \qquad \frac{\Gamma \vdash r : \forall X.A}{\Gamma \vdash r\,B : A[X := B]}$$

A term t is *typable* if there are Γ and A such that $\Gamma \vdash t : A$. Write $\vdash t : A$ for $\emptyset \vdash t : A$. If there is a term t with $\vdash t : A$, the type A is *inhabited.* Evidently, if $\Gamma \vdash t : A$ and $\Gamma \subseteq \Gamma'$ for the context Γ', then $\Gamma' \vdash t : A$ as well, i. e., *weakening* is an admissible typing rule. In the sequel, this word will be reserved for the rule

$$\frac{\Gamma \vdash t : B}{\Gamma, x : A \vdash t : B}\; x \text{ not declared in } \Gamma$$

It is also easy to see that the following *cut* rules are admissible typing rules:

$$\frac{\Gamma, x : A \vdash t : B \qquad \Gamma \vdash s : A}{\Gamma \vdash t[x := s] : B} \qquad \frac{\Gamma \vdash t : B}{\Gamma[X := A] \vdash t[X := A] : B[X := A]}$$

Here, $\Gamma[X := A] := x_1 : B_1[X := A], \dots, x_n : B_n[X := A]$ for $\Gamma = x_1 : B_1, \dots, x_n : B_n$.

Reduction. The one-step reduction relation $t \longrightarrow t'$ between terms t and t' is defined as the closure of the following axioms under all term constructors.

$$\begin{array}{lll}(\lambda x^A.t)\, s & \longrightarrow_\beta & t[x := s] \\ (\Lambda X.t)\, B & \longrightarrow_\beta & t[X := B]\end{array}$$

We denote the transitive closure of $\longrightarrow$ by $\longrightarrow^+$ and the reflexive-transitive closure by $\longrightarrow^*$. It is clear that $r \longrightarrow r'$ implies $r[x := s] \longrightarrow r'[x := s]$ and $s[x := r] \longrightarrow^* s[x := r']$.

Normal terms. Inductively define the set NF of *normal forms* by:

- If, for all $i \in \{1, \dots, n\}$, either $s_i \in \mathsf{NF}$ or s_i is a type, then $x s_1 \dots s_n \in \mathsf{NF}$.
- If $t \in \mathsf{NF}$ then $\lambda x^A.t \in \mathsf{NF}$ and $\Lambda X.t \in \mathsf{NF}$.

It is easy to see that the terms in NF are exactly those terms t such that no t' exists with $t \longrightarrow t'$, i. e., the normal forms are the *normal terms*.

Metatheoretic properties.

Lemma 1 (Subject Reduction) *If $\Gamma \vdash t : A$ and $t \longrightarrow t'$ then $\Gamma \vdash t' : A$.*

Proof This only requires the cut rules above. Subject reduction would be much more difficult in a formulation with typing à la Curry. □

Lemma 2 (Local Confluence) F *is locally confluent, i. e., if $r \longrightarrow r_1, r_2$ then there is a term t such that $r_1, r_2 \longrightarrow^* t$.*

Proof There are no critical pairs. □

It is well known that F is even *confluent*, i. e., $\longrightarrow^*$ is locally confluent.

Lemma 3 (Strong Normalization) *If $\Gamma \vdash t : A$ then there is no infinite reduction sequence $t \longrightarrow t_1 \longrightarrow t_2 \longrightarrow \dots$ In other words, every typable term is strongly normalizing.*

This deep result is known since Tait's adaptation [Tai75] of Girard's proof of weak normalization [Gir72].

2.2 Classical Natural Deduction

It is well known that system F is not classical: There is no closed term of type $\neg\neg X \to X$, for X any type variable. (This is equivalent to the non-existence of a closed term of type $\forall X. \neg\neg X \to X$.)

A proof can be obtained as follows. If, to the contrary, there were such a term r_c, one would also get a closed term r_p of type $(\neg X \to X) \to X$, which amounts to the Peirce law: Set

$$r_p := \lambda x^{\neg X \to X}.\, r_c\, (\lambda y^{\neg X}.y\, (x\, y)).$$

By normalization of F, there would also be a closed *normal* term r of type $(\neg X \to X) \to X$. Several steps of reasoning about normal terms would yield that $r = \lambda x^{\neg X \to X}.\, x\, (\lambda z^X.s)$ with s a normal term, and $x : \neg X \to X, z : X \vdash s : \bot$. This is plainly impossible, by inspection of normal forms.[1]

Therefore, F has to be extended in order to be classical. Following Prawitz [Pra65, Pra71], system $\mathsf{F}^\neg$ will now be based on "reductio ad absurdum" (RAA): If one can derive absurdity from the assumption $\neg A$, then A obtains. A term notation for a first-order system in this spirit has been proposed by Rehof and Sørensen [RS94], and later on even for pure type systems

[1]This proof—with the detour through Peirce's law—also works in case that $\bot$ is set to $\forall Y.Y$, i. e., for the system with built-in *ex falsum quodlibet* $\forall X. \bot \to X$.

[BHS97]. While those authors use the letter Δ for the binder, our name μ is taken from Parigot's $\lambda\mu$-calculus [Par92].

The term syntax of $\mathsf{F}^\neg$ is that of F, plus

$$r, s, t \quad ::= \quad \ldots \mid \mu x^{\neg A}.r$$

As regards variable binding, $\mu x^{\neg A}.r$ is the same as $\lambda x^{\neg A}.r$.

The new typing rule models RAA:

$$\frac{\Gamma, x : \neg A \vdash r : \bot}{\Gamma \vdash \mu x^{\neg A}.r : A}$$

We also add the axiom schemes of μ-reduction to the reduction system:

$$\begin{array}{lll} (\mu x^{\neg(A \to B)}.r)\, s & \longrightarrow_\mu & \mu y^{\neg B}.\, r[x := \lambda z^{A \to B}.\, y\,(z\, s)] \\ (\mu x^{\neg \forall X.A}.r)\, B & \longrightarrow_\mu & \mu y^{\neg A[X:=B]}.\, r[x := \lambda z^{\forall X.A}.\, y\,(z\, B)] \end{array}$$

The new term closure will again be denoted by $\longrightarrow$.

The first rule is present in λ_Δ in [RS94], the second one is the canonical analogue for the second-order quantification, as has been studied for $\lambda\mu$-calculus in [Par97].

Motivation of the μ-rules. RAA is equivalent to stability, $\forall X.\, \neg\neg X \to X$, also called "duplex negatio affirmat": If one introduces a constant stab^A assuming $\neg\neg A \to A$, then $\mu x^{\neg A}.r$ can be defined as $\mathsf{stab}^A(\lambda x^{\neg A}.r)$. In the other direction, one may define the term stab^A of type $\neg\neg A \to A$ as $\mathsf{stab}^A := \lambda v^{\neg\neg A} \mu x^{\neg A}.\, v\, x$.

It is general knowledge that, in minimal logic, stability of $A \to B$ follows from stability of B. In a first-order system, this may be used to assume only constants $\mathsf{stab}^X : \neg\neg X \to X$ for type variables X, and to define the closed term

$$\mathsf{stab}^{A \to B} := \lambda v^{\neg\neg(A \to B)} \lambda w^A.\, \mathsf{stab}^B\, (\lambda y^{\neg B}.\, v\, (\lambda z^{A \to B}.\, y\, (z\, w)))$$

of type $\neg\neg(A \to B) \to A \to B$. The above-mentioned encoding of RAA immediately yields the first μ-rule:

$$\begin{array}{lll} (\mu x^{\neg(A \to B)}.r)\, s & = & \mathsf{stab}^{A \to B}\, (\lambda x^{\neg(A \to B)}.r)\, s \\ & \longrightarrow^2 & \mathsf{stab}^B \Big(\lambda y^{\neg B}.\, (\lambda x^{\neg(A \to B)}.r)\, (\lambda z^{A \to B}.\, y\, (z\, s))\Big) \\ & = & \mu y^{\neg B}.\, \underbrace{(\lambda x^{\neg(A \to B)}.r)\, (\lambda z^{A \to B}.\, y\, (z\, s))}_{\longrightarrow r[x := \lambda z^{A \to B}.\, y\, (z\, s)]} \end{array}$$

Therefore, the termination of simply-typed lambda-calculus with the first-order μ-reduction rule is very easy.

We might extend our definition to the second order as follows:

$$\mathsf{stab}^{\forall X.A} := \lambda v^{\neg\neg \forall X.A} \Lambda X.\, \mathsf{stab}^A\, (\lambda y^{\neg A}.\, v\, (\lambda z^{\forall X.A}.\, y\, (z\, X))),$$

which yields a closed term of type $\neg\neg(\forall X.A) \to \forall X.A$ if stab^A is closed and of type $\neg\neg A \to A$. Again with our encoding of RAA, we would get

$$\begin{array}{lll} (\mu x^{\neg \forall X.A}.r)\, B & = & \mathsf{stab}^{\forall X.A}\, (\lambda x^{\neg \forall X.A}.r)\, B \\ & \longrightarrow^2 & \mathsf{stab}^A[X := B] \Big(\lambda y^{\neg A[X:=B]}.\, (\lambda x^{\neg \forall X.A}.r)\, (\lambda z^{\forall X.A}.\, y\, (z\, B))\Big) \\ & = & \mu y^{\neg A[X:=B]}.\, \underbrace{(\lambda x^{\neg \forall X.A}.r)\, (\lambda z^{\forall X.A}.\, y\, (z\, B))}_{\longrightarrow r[x := \lambda z^{\forall X.A}.\, y\, (z\, B)]} \end{array}$$

In the passage from the last but one line to the last line, we assumed that $\mathsf{stab}^A[X := B] = \mathsf{stab}^{A[X:=B]}$, which can certainly not be taken for granted! stab^X in $\mathsf{stab}^X[X := B]$ is a constant while stab^B is a defined term if B is non-atomic. An easy remedy seems to be a non-standard definition of substitution, namely $\mathsf{stab}^X[X := B] := \mathsf{stab}^B$. But this would give a non-trivial extension of F whose normalization would have to be established, hence no reduction to F would have been obtained.

One might want to repair the situation by assuming stability of X in the context and hence taking variables instead of constants. Unfortunately, the above definition of $\mathsf{stab}^{\forall X.A}$ does not allow stab^A to be typed with X free in the typing context, hence rules out this idea. Nevertheless, Joly [Jol97] found an embedding of $\mathsf{F}^\neg$ into F that allows to infer strong normalization for $\mathsf{F}^\neg$ from that of F, and that essentially follows our failed motivation of the μ-reduction rules. This will be reported in more detail in section 5.1.

A less ambitious but correct motivation for the second-order rule can nevertheless be given, just by using the "other direction" in the equivalence of RAA and stability: stab^A is only an abbreviation for $\lambda v^{\neg\neg A}\mu x^{\neg A}.\, v\, x$, and we require that $\mathsf{stab}^{\forall X.A}$ and the term

$$\lambda v^{\neg\neg\forall X.A}\Lambda X.\ \mathsf{stab}^A\,(\lambda y^{\neg A}.\, v\,(\lambda z^{\forall X.A}.\, y\,(z\,X))),$$

that defined $\mathsf{stab}^{\forall X.A}$ above, are equivalent with respect to $=_\beta$, which is the symmetric, transitive and reflexive closure of β-reduction $\longrightarrow$ of F. We then calculate

$$\begin{array}{lll}
(\mu x^{\neg\forall X.A}.r)\,B & =_\beta & (\mu x^{\neg\forall X.A}.(\lambda x^{\neg\forall X.A}.\,r)\,x)\,B \\
& =_\beta & (\lambda v^{\neg\neg\forall X.A}\mu x^{\neg\forall X.A}.vx)\,(\lambda x^{\neg\forall X.A}.r)\,B \\
& = & \mathsf{stab}^{\forall X.A}\,(\lambda x^{\neg\forall X.A}.r)\,B \\
& =_\beta & \Big(\lambda v^{\neg\neg\forall X.A}\Lambda X.\ \mathsf{stab}^A\,(\lambda y^{\neg A}.\,v\,(\lambda z^{\forall X.A}.\,y\,(z\,X)))\Big)\,(\lambda x^{\neg\forall X.A}.r)\,B \\
& =_\beta & \mathsf{stab}^A[X:=B]\Big(\lambda y^{\neg A[X:=B]}.\,(\lambda x^{\neg\forall X.A}.r)\,(\lambda z^{\forall X.A}.\,y\,(z\,B))\Big) \\
& = & (\lambda v^{\neg\neg A[X:=B]}\mu y^{\neg A[X:=B]}.vy)\,\Big(\lambda y^{\neg A[X:=B]}.\,(\lambda x^{\neg\forall X.A}.r)\,(\lambda z^{\forall X.A}.\,y\,(z\,B))\Big) \\
& =_\beta & \mu y^{\neg A[X:=B]}.\ \underbrace{(\lambda x^{\neg\forall X.A}.r)\,(\lambda z^{\forall X.A}.\,y\,(z\,B))}_{=_\beta\, r[x:=\lambda z^{\forall X.A}.\,y\,(z\,B)]}
\end{array}$$

Note that this second approach to a motivation does not say anything about normalization. We record that second-order classical natural deduction is not only more expressive than first-order classical natural deduction (this is vacuously true!), but that the reduction to the respective intuitionistic subsystem is essentially more complicated in the second-order case than for first order.

Lemma 4 *Subject reduction holds for* $\mathsf{F}^\neg$.

Proof The first attempt for a motivation of the μ-reduction rules already justifies subject reduction for these rules, the β-reduction rules can be separately treated for F alone, and the term closure does not raise any difficulty. $\square$

Lemma 5 (Local Confluence) $\mathsf{F}^\neg$ *is locally confluent.*

Proof As for F, there are no critical pairs. $\square$

Therefore, strong normalization, as shown below, yields confluence for typable terms by Newman's lemma. Certainly, the method of complete developments by Takahashi [Tak95] can be extended to show even confluence for all terms.

3 Stabilization in Iterative Style

In this section, the essential new tool (first published in [Mat01]) for the reduction of second-order classical logic to second-order intuitionistic logic is defined: An extension $\mathsf{I}\sharp$ of F with types of the form $\sharp A$ that represents the *least* stable supertype of A. This minimality give rise to an iteration principle that explains the heading "stabilization in iterative style". Although this is the only way stabilization is treated in the present article, this name is chosen in order to distinguish it from another approach to stabilization capable of capturing sums with permutative conversions, to be reported elsewhere.

The first part is devoted to the definition of $\mathsf{I}\sharp$, and to its (impredicative) justification in terms of F, whose strong normalization is inherited through an embedding. In the second part, we find a retract embedding of $\neg\neg A$ into $\sharp A$, giving a link to the Kolmogorov translation on which CPS-translations are based.

3.1 Definition of System $\mathsf{I}\sharp$

We define system $\mathsf{I}\sharp$ as extension of system F by stabilization types with iteration. The type system of $\mathsf{I}\sharp$ is

$$A, B, C, D \quad ::= \quad X \mid \bot \mid A \to B \mid \forall X.A \mid \sharp A$$

The term system and its typing reflects that $\sharp A$ is stable, that A embeds into $\sharp A$ and that $\sharp A$ is the least type with these properties.[2] We extend the grammar of terms with

$$r, s, t \quad ::= \quad \ldots \mid \mathsf{emb} \mid \mathsf{stab} \mid \mathsf{It}_\sharp^C(r, s, t)$$

The definition of $\Gamma \vdash t : A$ for F is extended by three clauses

$$\frac{}{\Gamma \vdash \mathsf{emb} : \forall X. X \to \sharp X} \qquad \frac{}{\Gamma \vdash \mathsf{stab} : \forall X. \neg\neg\sharp X \to \sharp X}$$

$$\frac{\Gamma \vdash r : \sharp A \qquad \Gamma \vdash s_1 : A \to C \qquad \Gamma \vdash s_2 : \neg\neg C \to C}{\Gamma \vdash \mathsf{It}_\sharp^C(r, s_1, s_2) : C}$$

The one-step relation $\longrightarrow$ of $\mathsf{I}\sharp$ is defined with the axiom schemes $\longrightarrow_\beta$ of system F plus the schemes

$$\begin{array}{lcl}
\mathsf{It}_\sharp^C(\mathsf{emb}\, A\, t, s_1, s_2) & \longrightarrow_\sharp & s_1\, t \\
\mathsf{It}_\sharp^C(\mathsf{stab}\, A\, t, s_1, s_2) & \longrightarrow_\sharp & s_2 \Big(\lambda y^{\neg C}. t\, (\lambda z^{\sharp A}. y\, \mathsf{It}_\sharp^C(z, s_1, s_2))\Big)
\end{array}$$

In the last rule, we assume that $y \neq z$ and that y, z not free in r, s_1, s_2.

These reduction rules also justify to call s_1 and s_2 the first and second *step term* of $\mathsf{It}_\sharp^C(r, s_1, s_2)$, respectively. The rules will be explained through an embedding in the proof of Lemma 9 below.

Lemma 6 *Subject reduction holds for* $\mathsf{I}\sharp$.

Proof The $\longrightarrow_\sharp$ axioms evidently preserve types, closure under term formation rules works as usual. □

Lemma 7 $\mathsf{I}\sharp$ *is confluent.*

Proof By an immediate extension of the complete developments method by Takahashi [Tak95]. Since there are no critical pairs, local confluence is even trivially true. [3] □

Lemma 8 $\mathsf{I}\sharp$ *is strongly normalizing.*

[2] The author is aware that already Paulin-Mohring [PM96, p. 110] describes a similar idea: An inductively defined predicate may be "made classical" by adding stability as another clause to the definition. This turns the definition into a non-strictly positive one and enforces stability. Note, however, that non-strictly positive inductive definitions lead to inconsistencies in higher-order predicate logic, see the example reported in [PM96, p. 108]. In the framework of system F, we are in the fortunate situation that arbitrary types may be stabilized without harm to consistency.

[3] If there were also η rules, even local confluence would not hold for non-typable terms.

The proof will occupy the remainder of this section.

The idea of the proof is as follows: Although the second rule for $\longrightarrow_\sharp$ looks unfamiliar, it is just an instance of iteration over a non-strictly positive inductive type, and hence the reduction behaviour can be simulated within F.

Definition 1 *A type-respecting reduction-preserving embedding (embedding for short) of a term rewrite system $\mathcal{S}$ with typing into a term rewrite system $\mathcal{S}'$ with typing is a function $-'$ (the $-$ sign represents the indefinite argument of the function $'$) which assigns to every type A of $\mathcal{S}$ a type A' of $\mathcal{S}'$ and to every term r of $\mathcal{S}$ a term r' of $\mathcal{S}'$ such that the following implications hold:*

- *If $x_1 : A_1, \ldots, x_n : A_n \vdash r : A$ in $\mathcal{S}$ then $x_1 : A'_1, \ldots, x_n : A'_n \vdash r' : A'$ in $\mathcal{S}'$.*
- *If $r \longrightarrow s$ in $\mathcal{S}$, then $r' \longrightarrow^+ s'$ in $\mathcal{S}'$.*

Obviously, if there is an embedding of $\mathcal{S}$ into $\mathcal{S}'$, then strong normalization of $\mathcal{S}'$ is inherited by $\mathcal{S}$. In the sequel, we will write Γ' for $x_1 : A'_1, \ldots, x_n : A'_n$ if Γ is $x_1 : A_1, \ldots, x_n : A_n$.

Lemma 9 *There is an embedding of $\mathsf{I}\sharp$ into F.*

Proof By the very description, $\sharp A$ is nothing but the non-strictly positive inductive type $\mathsf{lfp}X.A + \neg\neg X$ with $X \notin \mathsf{FV}(A)$, formulated without the sum type. Its canonical polymorphic encoding would be

$$\forall X.((A + \neg\neg X) \to X) \to X$$

which can be simplified to

$$\forall X.(A \to X) \to (\neg\neg X \to X) \to X.$$

By iteration on the type A of $\mathsf{I}\sharp$ define the type A' of F. This shall be done homomorphically in all cases except:

$$(\sharp A)' := \forall X.(A' \to X) \to (\neg\neg X \to X) \to X \text{ with } X \notin \mathsf{FV}(A).$$

Clearly, $\mathsf{FV}(A') = \mathsf{FV}(A)$ and $(A[X := B])' = A'[X := B']$.

By iteration on the term r of $\mathsf{I}\sharp$ define the term r' of F. (Simultaneously, one has to show that the free variables of r' and r coincide). We only present the non-homomorphic cases.

- $\mathsf{emb}' := \Lambda Y \lambda x^Y \Lambda X \lambda x_1^{Y \to X} \lambda x_2^{\neg\neg X \to X}.x_1 x$
- $\mathsf{stab}' := \Lambda Y \lambda x^{\neg\neg(\sharp Y)'} \Lambda X \lambda x_1^{Y \to X} \lambda x_2^{\neg\neg X \to X}.x_2(\lambda y^{\neg X}.x(\lambda z^{(\sharp Y)'}.y(zXx_1x_2)))$
- $(\mathsf{It}_\sharp^C(r, s_1, s_2))' := r'C's'_1s'_2$.

It is fairly easy to see that these settings form an embedding in the sense of definition 1: The typings go well and, since $(t[x := s])' = t'[x := s']$ and $(t[X := A])' = t'[X := A']$, it is a trivial calculation to prove that

$$\begin{array}{lll}
((\lambda x^A.t)\, s)' & \longrightarrow^1 & (t[x := s])' \\
((\Lambda X.t)\, A)' & \longrightarrow^1 & (t[X := A])' \\
(\mathsf{It}_\sharp^C(\mathsf{emb}\, A\, t, s_1, s_2))' & \longrightarrow^5 & (s_1\, t)' \\
(\mathsf{It}_\sharp^C(\mathsf{stab}\, A\, t, s_1, s_2))' & \longrightarrow^5 & \Big(s_2\Big(\lambda y^{\neg C}.t\,(\lambda z^{\sharp A}.y\, \mathsf{It}_\sharp^C(z, s_1, s_2))\Big)\Big)'
\end{array}$$

Here, $\longrightarrow^n$ denotes n steps of reduction $\longrightarrow$ (in F). In fact, the right-hand sides of the $\longrightarrow_\sharp$-rules originally have been found just by reducing the translations of the left-hand sides. $\square$

Corollary 1 *System $\mathsf{I}\sharp$ is strongly normalizing.*

Proof Strong normalization is inherited through our embedding from F. $\square$

3.2 Double Negation versus Stabilization

The Kolmogorov translation is based on double negation $\neg\neg A$ of A which clearly yields a stable type (since every negated formula is stable already in minimal logic), and A implies $\neg\neg A$. In our embedding of classical natural deduction, we will use stabilization $\sharp A$ of A, which is the least stable supertype of A by definition. Hence, one would expect $\sharp A$ to be "smaller" than $\neg\neg A$. Perhaps surprisingly, we will find a retract in the other direction.

Lemma 10 *$\neg\neg A$ and $\sharp A$ are logically equivalent. Moreover and more precisely, $\neg\neg A$ can be embedded by a retract into $\sharp A$, i. e., there are terms p, q of $\mathsf{I}\sharp$ with $\vdash p : \forall X. \sharp X \to \neg\neg X$ and $\vdash q : \forall X.\neg\neg X \to \sharp X$ such that $p \circ q := \Lambda X \lambda x^{\neg\neg X}.pX(qXx) \longrightarrow^{+}_{\beta\sharp\eta} \Lambda X \lambda x^{\neg\neg X}.x$.*

Here, the index η indicates that another axiom scheme for the reduction is $\lambda x^A.rx \longrightarrow_\eta r$ for x not free in r.

Proof For p, we just have to formalize the above discussion: Set $p := \Lambda X \lambda x^{\sharp X}.\mathsf{It}_\sharp^{\neg\neg X}(x, s_1, s_2)$ with terms s_1, s_2 such that $\vdash s_1 : X \to \neg\neg X$ and $\vdash s_2 : \neg\neg\neg\neg X \to \neg\neg X$. This is achieved with:

$$\begin{array}{rcl} s_1 & := & \lambda x^X \lambda k^{\neg X}.kx \\ s_2 & := & \lambda y^{\neg\neg\neg\neg X} \lambda z^{\neg X}.y(\lambda u^{\neg\neg X}.uz) \end{array}$$

Set $q := \Lambda X \lambda x^{\neg\neg X}.\,\mathsf{stab}\,X\,t$ with a term t such that $x : \neg\neg X \vdash t : \neg\neg\sharp X$, namely the canonical lifting of $z : X \vdash \mathsf{emb}\,X\,z : \sharp X$ over the double negation:

$$t \quad := \quad \lambda y^{\neg\sharp X}.x(\lambda z^X.y(\mathsf{emb}\,X\,z)).$$

In the following calculation, we always indicate in the reduction steps which kind of axiom scheme has been used:

$$\begin{array}{rll} p \circ q & \longrightarrow^4_\beta & \Lambda X \lambda x^{\neg\neg X}.\mathsf{It}_\sharp^{\neg\neg X}(\mathsf{stab}\,X\,t, s_1, s_2) \\ & \longrightarrow_\sharp & \Lambda X \lambda x^{\neg\neg X}.s_2(\lambda y^{\neg\neg\neg X}.t(\lambda z_1^{\sharp X}.y\,\mathsf{It}_\sharp^{\neg\neg X}(z_1, s_1, s_2))) \\ & \longrightarrow_\beta & \Lambda X \lambda x^{\neg\neg X} \lambda z^{\neg X}.(\lambda y^{\neg\neg\neg X}.t(\lambda z_1^{\sharp X}.y\,\mathsf{It}_\sharp^{\neg\neg X}(z_1, s_1, s_2)))(\lambda u^{\neg\neg X}.uz) \\ & \longrightarrow_\beta & \Lambda X \lambda x^{\neg\neg X} \lambda z^{\neg X}.t(\lambda z_1^{\sharp X}.(\lambda u^{\neg\neg X}.uz)\,\mathsf{It}_\sharp^{\neg\neg X}(z_1, s_1, s_2)) \\ & \longrightarrow_\beta & \Lambda X \lambda x^{\neg\neg X} \lambda z^{\neg X}.t(\lambda z_1^{\sharp X}.\mathsf{It}_\sharp^{\neg\neg X}(z_1, s_1, s_2)\,z) \\ & \longrightarrow_\beta & \Lambda X \lambda x^{\neg\neg X} \lambda z^{\neg X}.x(\lambda z_2^X.(\lambda z_1^{\sharp X}.\mathsf{It}_\sharp^{\neg\neg X}(z_1, s_1, s_2)\,z)(\mathsf{emb}\,X\,z_2)) \\ & \longrightarrow_\beta & \Lambda X \lambda x^{\neg\neg X} \lambda z^{\neg X}.x(\lambda z_2^X.\mathsf{It}_\sharp^{\neg\neg X}(\mathsf{emb}\,X\,z_2, s_1, s_2)\,z) \\ & \longrightarrow_\sharp & \Lambda X \lambda x^{\neg\neg X} \lambda z^{\neg X}.x(\lambda z_2^X.s_1\,z_2\,z) \\ & \longrightarrow^2_\beta & \Lambda X \lambda x^{\neg\neg X} \lambda z^{\neg X}.x(\lambda z_2^X.z\,z_2) \\ & \longrightarrow_\eta & \Lambda X \lambda x^{\neg\neg X} \lambda z^{\neg X}.xz \longrightarrow_\eta \Lambda X \lambda x^{\neg\neg X}.x \end{array}$$

□

Remark 1 *The reduction behaviour of $q \circ p := \Lambda X \lambda x^{\sharp X}.qX(pXx)$ with p, q taken from the above proof is not perspicuous. There is no indication at all that $q \circ p$ behaves like the identity. E. g.,*

$$\begin{array}{rll} (q \circ p)X(\mathsf{emb}\,X\,x) & \longrightarrow^4_\beta & qX(\mathsf{It}_\sharp^{\neg\neg X}(\mathsf{emb}\,X\,x, s_1, s_2)) \\ & \longrightarrow_\sharp & qX(s_1 x) \\ & \longrightarrow_\beta & qX(\lambda k^{\neg X}.kx) \\ & \longrightarrow^2_\beta & \mathsf{stab}\,X\,t[x := \lambda k^{\neg X}.kx] \\ & \longrightarrow_\beta & \mathsf{stab}\,X\,(\lambda y^{\neg\sharp X}.(\lambda z^X.y(\mathsf{emb}\,X\,z))x) \\ & \longrightarrow_\beta & \mathsf{stab}\,X\,(\lambda y^{\neg\sharp X}.y(\mathsf{emb}\,X\,x)) \end{array}$$

The last term cannot be reduced to $\mathsf{emb}\,X\,x$, and the author has no knowledge of a reasonable extension of the reduction system that allows to reduce $q \circ p$ to $\Lambda X \lambda x^{\sharp X}.x$. (Already the rule $\mathsf{stab}\,A(\lambda x^{\neg\sharp A}.xt) \longrightarrow_\eta t$ for x not free in t would break local confluence.)

We conclude that, in a sense, $\sharp A$ is an operationally larger type than $\neg\neg A$. In section 4.3, this will be exploited in an embedding of $\lambda\mu$-calculus into $\mathsf{I}\sharp$ that does not erase the μ-reductions.

4 A New Embedding of Classical Natural Deduction

We would like to inherit strong normalization of $\mathsf{F}^\neg$ from F, by giving an embedding of $\mathsf{F}^\neg$ into $\mathsf{I}\sharp$. This will fail due to some "administrative" reductions. By switching from $\mathsf{F}^\neg$ to a formulation of $\lambda\mu$-calculus, an embedding can be achieved.

In the first part, we expound the problems with embeddings of $\mathsf{F}^\neg$ into $\mathsf{I}\sharp$: Firstly, typing information is needed in order to *define* the embedding. Secondly, one has to provide a retract embedding of $\bot$ into $\sharp\bot$. Thirdly, the attempted embedding turns out not to *simulate* the μ-reductions.

The second part defines system $\lambda\mu$, our variant of $\lambda\mu$-calculus, the third part exhibits a "refined embedding" of $\lambda\mu$ into $\mathsf{I}\sharp$ and discusses why this would not work for a formulation with typing à la Curry.

4.1 An Unsuccessful Attempt to Embed $\mathsf{F}^\neg$ into $\mathsf{I}\sharp$

We aim at an embedding $-'$ of $\mathsf{F}^\neg$ into $\mathsf{I}\sharp$. The type translation will at least have the following definition clauses:

$$\begin{aligned} X' &:= \sharp X \\ \bot' &:= \sharp\bot \\ (A \to B)' &:= \sharp(A' \to B') \end{aligned}$$

In this way, we ensure that every image A' is of the form $\sharp B$, hence the type expressing its stability is inhabited by $\mathsf{stab}\, B$. Since B is unique with $\sharp B = A'$, we define $A^* := B$ for that B and have $A' = \sharp A^*$. For the time being, we do not say anything about universal quantification.

We would like to define a term r' of $\mathsf{I}\sharp$ by iteration on the term r of $\mathsf{F}^\neg$ such that $-'$ becomes an embedding of these systems in the sense of definition 1. Since this will fail, we only discuss the syntax elements x and $\lambda x^A.t$.

We are forced to map variables to variables in order to get a substitution lemma that is needed for the simulation of β-reduction. The canonical choice is $x' := x$ and will allow us to prove $(t[x := s])' = t'[x := s']$. If we already know that $\Gamma', x : A' \vdash t' : B'$, we want to define $(\lambda x^A.t)'$ such that $\Gamma' \vdash (\lambda x^A.t)' : (A \to B)' = \sharp(A' \to B')$. This would be achieved by setting

$$(\lambda x^A.t)' := \mathsf{emb}(A' \to B')(\lambda x^{A'}.t').$$

Unfortunately, the type B' cannot be inferred from the input term $\lambda x^A.t$, unless one knows the context Γ. Hence, the notion of embedding in definition 1 is too narrow.

Definition 2 *A context-sensitive type-respecting reduction-preserving embedding (refined embedding for short) of a term rewrite system $\mathcal{S}$ with typing into a term rewrite system $\mathcal{S}'$ with typing is a function $-'$ which assigns to every type A of $\mathcal{S}$ a type A' of $\mathcal{S}'$, and for every context Γ of $\mathcal{S}$ a function $-'_\Gamma$ which assigns to every term r of $\mathcal{S}$ typable in context Γ, a term r'_Γ of $\mathcal{S}'$ such that the following implications hold:*

- *If $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ and $\Gamma \vdash r : A$ in $\mathcal{S}$ then $x_1 : A'_1, \ldots, x_n : A'_n \vdash r'_\Gamma : A'$ in $\mathcal{S}'$.*
- *If r is typable in context Γ and $r \longrightarrow s$ in $\mathcal{S}$, then $r'_\Gamma \longrightarrow^+ s'_\Gamma$ in $\mathcal{S}'$.*

Clearly, the above definition is well-formed in case $\mathcal{S}$ has subject reduction: s'_Γ is defined if r'_Γ is. We use the notation Γ' as described for definition 1.

Even with this refined notion, our attempt to define an embedding of $\mathsf{F}^\neg$ into $\mathsf{I}\sharp$ will fail. Again, we do not treat universal quantification and only discuss the syntax elements x, $\lambda x^A.t$, rs and $\mu x^{\neg A}.r$.

We set $x'_\Gamma := x$. If $\lambda x^A.t$ is typable in context Γ, then $\Gamma, x : A \vdash t : B$ for a unique type B. Set $\Delta := \Gamma, x : A$ and

$$(\lambda x^A.t)'_\Gamma := \mathsf{emb}(A' \to B')(\lambda x^{A'}.t'_\Delta).$$

In a proof that this definition respects types (the first condition in the definition), one would have $\Gamma', x : A' \vdash t'_\Delta : B'$ by induction hypothesis, hence $\Gamma' \vdash (\lambda x^A.t)'_\Gamma : (A \to B)'$.

With the (inductive) assumptions $\Gamma' \vdash r'_\Gamma : (A \to B)' = \sharp(A' \to B')$ and $\Gamma' \vdash s'_\Gamma : A'$ in mind, we set

$$(rs)'_\Gamma := \mathsf{It}_\sharp^{B'}(r'_\Gamma, \lambda z^{A' \to B'}.zs'_\Gamma, \mathsf{stab}\, B^*)$$

with z not free in s. Since $\Gamma' \vdash \lambda z^{A' \to B'}.zs'_\Gamma : (A' \to B') \to B'$ and $\Gamma' \vdash \mathsf{stab}\, B^* : \neg\neg B' \to B'$, we get $\Gamma' \vdash (rs)'_\Gamma : B'$. The definition is well-formed since A and B are uniquely determined by Γ and r: The type $A \to B$ is the unique type of r in context Γ.

The crucial case is as follows: Setting $\Delta := \Gamma, x : \neg A$, we may assume that $\Gamma', x : (\neg A)' \vdash r'_\Delta : \bot'$ and want to define $(\mu x^{\neg A}.r)'_\Gamma$ such that $\Gamma' \vdash (\mu x^{\neg A}.r)'_\Gamma : A'$. Unfolding the definitions, our assumption reads $\Gamma', x : \sharp(A' \to \sharp\bot) \vdash r'_\Delta : \sharp\bot$. First, we need to go back and forth between $\sharp\bot$ and $\bot$:

Lemma 11 *The type $\bot$ can be embedded by a retract into $\sharp\bot$, i. e., there are terms p, q with $\vdash p : \sharp\bot \to \bot$ and $\vdash q : \bot \to \sharp\bot$ such that $p(qx) \longrightarrow^+ x$. (Note that we do not even need η reductions here.)*

Proof Set $q := \mathsf{emb}\,\bot$ and $p := \lambda x^{\sharp\bot}.\mathsf{It}_\sharp^\bot(x, \mathsf{id}^\bot, \lambda z^{\neg\neg\bot}.z\,\mathsf{id}^\bot)$. Calculate $p(qx) = p(\mathsf{emb}\,\bot\, x) \longrightarrow^2 \mathsf{id}^\bot\, x \longrightarrow x$. □

With these terms p, q we define $q \circ^A y := \lambda z^{A'}.q(yz)$ (think of $y : \neg A'$ in the context, hence $q \circ^A y : A' \to \sharp\bot$) and

$$(\mu x^{\neg A}.r)'_\Gamma := \mathsf{stab}\, A^*\Big(\lambda y^{\neg A'}.p\Big(r'_\Delta[x := \mathsf{emb}(A' \to \sharp\bot)(q \circ^A y)]\Big)\Big).$$

This is type-correct since, we have $y : \neg A' \vdash \mathsf{emb}(A' \to \sharp\bot)(q \circ^A y) : \sharp(A' \to \sharp\bot)$, hence $\Gamma', y : \neg A' \vdash r'_\Delta[x := \mathsf{emb}(A' \to \sharp\bot)(q \circ^A y)] : \sharp\bot$ by the cut rule, finally, $\Gamma' \vdash \lambda y^{\neg A'}.p(r'_\Delta[x := \mathsf{emb}(A' \to \sharp\bot)(q \circ^A y)]) : \neg\neg A'$.

For the syntax captured by these definitions, we can prove that weakening and cut also hold for the prospective embedding, in the following sense:

- If $\Gamma \vdash t : B$ and x is not declared in Γ, then $t'_\Gamma = t'_{\Gamma, x:A}$.
- If $\Gamma, x : A \vdash t : B$ and $\Gamma \vdash s : A$ then $(t[x := s])'_\Gamma = t'_{\Gamma, x:A}[x := s'_\Gamma]$.

Although ordinary β-reduction would be simulated, we demonstrate that, for $(\mu x^{\neg(A \to B)}.r)s$ typable in Γ, we do *not* have

$$L := ((\mu x^{\neg(A \to B)}.r)s)'_\Gamma \longrightarrow^+ (\mu y^{\neg B}.r[x := \lambda z^{A \to B}.y(zs)])'_\Gamma =: R$$

Abbreviate $\mathsf{emb}(A' \to \sharp\bot)$ by e^A and set $\Delta := \Gamma, x : \neg(A \to B)$.

$$\begin{array}{lcl}
L & = & \mathsf{It}_\sharp^{B'}\Big(\mathsf{stab}(A \to B)^*\Big(\lambda y^{\neg(A \to B)'}.p(r'_\Delta[x := e^{A \to B}(q \circ^{A \to B} y)])\Big), \lambda z^{A' \to B'}.zs'_\Gamma, \mathsf{stab}\, B^*\Big) \\
 & \longrightarrow_\sharp & \mathsf{stab}\, B^*\Big(\lambda y^{\neg B'}.\Big(\lambda y^{\neg(A \to B)'}.p(r'_\Delta[x := e^{A \to B}(q \circ^{A \to B} y)])\Big)(\lambda z^{(A \to B)'}.y(zs)'_{\Gamma, z:A \to B})\Big) \\
 & \longrightarrow_\beta & \mathsf{stab}\, B^*\Big(\lambda y^{\neg B'}.p\Big(r'_\Delta\Big[x := e^{A \to B}\Big(\lambda z^{(A \to B)'}.q\Big((\lambda z^{(A \to B)'}.y(zs)'_{\Gamma, z:A \to B})z\Big)\Big)\Big]\Big)\Big) \\
 & \longrightarrow^* & \mathsf{stab}\, B^*\Big(\lambda y^{\neg B'}.p\Big(r'_\Delta\Big[x := e^{A \to B}\Big(\lambda z^{(A \to B)'}.q(y(zs)'_{\Gamma, z:A \to B})\Big)\Big]\Big)\Big) =: L^+ \\
R & = & \mathsf{stab}\, B^*\Big(\lambda y^{\neg B'}.p\Big(r'_\Delta[x := \underbrace{(\lambda z^{A \to B}.y(zs))'_{\Gamma, y:\neg B}}_{e^{A \to B}(\lambda z^{(A \to B)'}.(y(zs))'_{\Gamma, y:\neg B, z:A \to B})}][y := e^B(q \circ^B y)]\Big)\Big) \\
 & = & \mathsf{stab}\, B^*\Big(\lambda y^{\neg B'}.p\Big(r'_\Delta\Big[x := e^{A \to B}\Big(\lambda z^{(A \to B)'}.(y(zs))'_{\Gamma, y:\neg B, z:A \to B}[y := e^B(q \circ^B y)]\Big)\Big]\Big)\Big) \\
 & \longrightarrow^* & L^+
\end{array}$$

This is justified by the following reductions for every occurrence of x in r'_Δ (only in case x not free in r'_Δ, the desired $R = L^+$ holds true):

$$(y(zs))'_{\Gamma, y:\neg B, z:A \to B} = \mathsf{It}_\sharp^{\sharp\bot}(y, \lambda u^{B' \to \sharp\bot}.u(zs)'_{\Gamma, z:A \to B}, \mathsf{stab}\,\bot),$$

hence

$$\begin{array}{rcl}(y(zs))'_{\Gamma,y:\neg B,z:A\to B}[y:=e^B(q\circ^B y)] & = & \mathsf{It}^{\sharp\perp}_{\sharp}(e^B(q\circ^B y),\lambda u^{B'\to\sharp\perp}.u(zs)'_{\Gamma,z:A\to B},\mathsf{stab}\,\perp)\\ & \longrightarrow_{\sharp} & (\lambda u^{B'\to\sharp\perp}.u(zs)'_{\Gamma,z:A\to B})(q\circ^B y)\\ & \longrightarrow_{\beta} & (q\circ^B y)(zs)'_{\Gamma,z:A\to B}\\ & \longrightarrow_{\beta} & q(y(zs)'_{\Gamma,z:A\to B})\end{array}$$

The author does not see a way how to infer strong normalization of $\mathsf{F}^{\neg}$ from that of $\mathsf{I}\sharp$ due to this failure of simulation. The unwanted "administrative" reductions in R will be avoided if we do not replace x by $\lambda z^{A\to B}.y(zs)$ in the μ-reduction rule, but, loosely speaking, every occurrence of a subterm of the form $x\,t$ by $y(ts)$. This will only be a reasonable reduction relation if every occurrence of x is in a subterm of that form $x\,t$. Certainly, this cannot be expected. But we only need it for the variables x that are bound by μ. Hence, we introduce two name spaces for variables, one for the usual variables that may be bound by λ, hence called λ-variables, the other one for the variables possibly bound by μ, called μ-variables. And only for those μ-variables a, we will ensure that they occur only in a subterm of the form $a\,t$. Essentially, this gives Parigot's $\lambda\mu$-calculus [Par92]. However, like de Groote [dG94], we will not be so strict as to require that one can only μ-abstract over terms of that form $a\,t$. We even allow $\perp$ as a type former and hence stay more closely to the more recent presentation by de Groote [dG01].

4.2 Refined Classical Natural Deduction: $\lambda\mu$-Calculus

As mentioned above, Parigot's second-order typed $\lambda\mu$-calculus will be slightly extended in the style of de Groote [dG01]. However, we do not put the typing context for the μ-variables to the right-hand side of $\vdash$. Moreover, we use raw syntax in Church style.

The system of types of $\lambda\mu$ is just that of F (which is the same as that for $\mathsf{F}^{\neg}$). The term system of F is extended as follows: There is an infinite set of μ-variables, whose elements are denoted by a, b, c. They are supposed to be disjoint from the variables x, now called λ-variables. The additional formation rules for terms in $\lambda\mu$ are

$$r, s, t \quad ::= \quad \ldots \mid a\,t \mid \mu a^A.r$$

Hence, a μ-variable alone is no term, and every variable that will ever get bound by μ only occurs as left-hand side a of an application $a\,t$.

The contexts may also contain pairs $(a : A)$ of μ-variables and types, to be interpreted as a having type $\neg A$.

In addition to the typing rules of F, we have the rules involving the μ-variables

$$\frac{\Gamma, a : A \vdash t : A}{\Gamma, a : A \vdash a\,t : \perp} \qquad \frac{\Gamma, a : A \vdash r : \perp}{\Gamma \vdash \mu a^A.r : A}$$

Although a μ-variable is not a term, it can be turned into a term of the appropriate type: We have $a : A \vdash \lambda x^A.\,a\,x : \neg A$.

The axiom schemes for $\longrightarrow_\beta$ are taken from F. $\longrightarrow_\mu$ of $\mathsf{F}^{\neg}$ is changed to a rule that no longer has a lambda abstraction in the right-hand side. We need a special form of substitution which is called structural substitution by de Groote [dG01].

Definition 3 *A term context is derived from a term by replacing exactly one occurrence of a λ-variable in that term with the new symbol $\star$. This occurrence must not be in the scope of any binder (for simplicity). Term contexts will be communicated by the letter E, and the result of a textual substitution of a term r for $\star$ in E, will be denoted by $E[r]$ (and is a term).*

Definition 4 (Special Substitution) *For r a term, μ-variables a, b and a term context E, define the term $r[a\star := b\,E]$ as the result of replacing in r recursively every subterm of the form*

$a\,t$ by $b\,(E[t])$. The precise definition is by iteration on r. Every case is homomorphic with exception of

$$(c\,t)[a\star := b\,E] := \begin{cases} b\,(E[t[a\star := b\,E]]) & \text{, if } c = a \\ c\,(t[a\star := b\,E]) & \text{, else} \end{cases}$$

Lemma 12 *If $\Gamma, a : A \vdash r : C$ and $\Gamma, z : A \vdash E[z] : B$ then $\Gamma, b : B \vdash r[a\star := b\,E] : C$.* □

Add the axiom schemes of μ-reduction (with a "fresh" μ-variable b)

$$\begin{array}{lll} (\mu a^{A\to B}.r)\,s & \longrightarrow_\mu & \mu b^B.\,r[a\star := b\,(\star\,s)] \\ (\mu a^{\forall X.A}.r)\,B & \longrightarrow_\mu & \mu b^{A[X:=B]}.\,r[a\star := b\,(\star\,B)] \end{array}$$

Evidently, this rule just imposes a bit of a reduction strategy on the reduction relation of $\mathsf{F}^\neg$ since, with the only reasonable interpretation of $r[a := t]$, one has in $\lambda\mu$ that

$$r[a := \lambda z^A.b\,(E[z])] \longrightarrow^*_\beta r[a\star := b\,E].$$

Therefore, $\lambda\mu$ embeds into $\mathsf{F}^\neg$ with $-'$ the identity, using the following definition:

Definition 5 *An embedding of a $\lambda\mu$-calculus $\mathcal{S}$ into an extension $\mathcal{S}'$ of system F without μ-variables is a function $-'$ which assigns to every type A of $\mathcal{S}$ a type A' of $\mathcal{S}'$ and to every term r of $\mathcal{S}$ a term r' of $\mathcal{S}'$ such that the following implications hold:*

- *If $x_1 : A_1, \dots, x_n : A_n, a_1 : B_1, \dots, a_k : B_k \vdash r : A$ in $\mathcal{S}$, then one has the typing $x_1 : A'_1, \dots, x_n : A'_n, a_1 : \neg B'_1, \dots, a_k : \neg B'_k \vdash r' : A'$ in $\mathcal{S}'$.*
- *If $r \longrightarrow s$ in $\mathcal{S}$, then $r' \longrightarrow^+ s'$ in $\mathcal{S}'$.*

Here, we generally consider the μ-variables of $\mathcal{S}$ as term variables of $\mathcal{S}'$.

This definition shows the way to overcome the problems reported in section 4.1: The translated a will be thought of as having type $\neg A'$ and no longer the type $(\neg A)' = \sharp(A' \to \sharp\bot)$.

It is obvious that also an embedding in the sense of this definition allows to derive strong normalization of $\mathcal{S}$ from that of $\mathcal{S}'$. As for definition 1, we call the transformed context Γ' if the original context was Γ in the above implication.

4.3 Embedding $\lambda\mu$ into $\mathsf{I}\sharp$

It is known from [Par97] that $\lambda\mu$ is strongly normalizing. It also follows from strong normalization of $\mathsf{F}^\neg$, shown in [Mat04]. Here, we want to give an elementary proof through an embedding into $\mathsf{I}\sharp$, and hence only have to appeal to the now classical result that F is strongly normalizing. Another embedding into F has been found by Joly [Jol97]—see section 5.1—but that one does not seem to extend to fixed-points, as described in section 5.2. Translations in continuation-passing style neither reprove strong normalization of F, but are no embedding in the sense of simulation of all reductions—see section 5.3.

This section presents an extension of [Mat01] from $\lambda\mu$-calculus in a formulation closer to Parigot to the present one with $\bot$ in the type system and in a precise Church-style typing discipline.

The attempted embedding of section 4.1 is adjusted to system $\lambda\mu$ and works out smoothly. Certainly, again, the naive notion of embedding (this time definition 5) is too narrow to be useful and has to be relativized to contexts:

Definition 6 *A context-sensitive type-respecting reduction-preserving embedding (refined embedding for short) of a $\lambda\mu$-calculus $\mathcal{S}$ into an extension $\mathcal{S}'$ of system F without μ-variables is a function $-'$ which assigns to every type A of $\mathcal{S}$ a type A' of $\mathcal{S}'$, and for every context Γ of $\mathcal{S}$ a function $-'_\Gamma$ which assigns to every term r of $\mathcal{S}$ typable in context Γ, a term r'_Γ of $\mathcal{S}'$ such that the following implications hold:*

- *If* $\Gamma = x_1 : A_1, \ldots, x_n : A_n, a_1 : B_1, \ldots, a_k : B_k$ *and* $\Gamma \vdash r : A$ *in* $\mathcal{S}$ *then, for*
$$\Gamma' := x_1 : A_1', \ldots, x_n : A_n', a_1 : \neg B_1', \ldots, a_k : \neg B_k',$$
one has $\Gamma' \vdash r'_\Gamma : A'$ *in* $\mathcal{S}'$.
- *If* r *is typable in context* Γ *and* $r \longrightarrow s$ *in* $\mathcal{S}$, *then* $r'_\Gamma \longrightarrow^+ s'_\Gamma$ *in* $\mathcal{S}'$.

Here, we generally consider the μ*-variables of* $\mathcal{S}$ *as term variables of* $\mathcal{S}'$.

Theorem 1 *There is a refined embedding of* $\lambda\mu$ *into* $\mathsf{I}\sharp$.

The proof forms the remainder of this section.

Definition 7 *By iteration on the type* A *of* $\lambda\mu$ *define the type* A^* *of* $\mathsf{I}\sharp$:

$$\begin{aligned} X^* &:= X \\ \bot^* &:= \bot \\ (A \to B)^* &:= \sharp A^* \to \sharp B^* \\ (\forall X.A)^* &:= \forall X.\, \sharp A^* \end{aligned}$$

Clearly, $\mathsf{FV}(A^*) = \mathsf{FV}(A)$. By an easy induction, one gets $(A[X := B])^* = A^*[X := B^*]$.

The translation of the types for the embedding is defined by $A' := \sharp A^*$. Trivially, $\mathsf{FV}(A') = \mathsf{FV}(A)$ and

$$\begin{aligned} X' &= \sharp X \\ \bot' &= \sharp\bot \\ (A \to B)' &= \sharp(A' \to B') \\ (\forall X.A)' &= \sharp\forall X.\, A' \end{aligned}$$

Therefore, the first three clauses are as in section 4.1. Important for the treatment of universal quantification will be the immediate consequence of the above substitution property:

$$(A[X := B])' = A'[X := B^*].$$

Definition 8 *By iteration on the term* r *of* $\lambda\mu$ *define the term* r'_Γ *of* $\mathsf{I}\sharp$ *whenever* r *is typable in context* Γ *in* $\lambda\mu$:

- $x'_\Gamma := x$.
- *If* $\Gamma \vdash \lambda x^A.t : A \to B$ *then* $(\lambda x^A.t)'_\Gamma := \mathsf{emb}(A' \to B')(\lambda x^{A'}.t'_{\Gamma,x:A})$.
- *If* $\Gamma \vdash r : A \to B$ *and* $\Gamma \vdash s : A$ *then* $(rs)'_\Gamma := \mathsf{It}_\sharp^{B'}(r'_\Gamma, \lambda z^{A' \to B'}.zs'_\Gamma, \mathsf{stab}\, B^*)$.
- *If* $\Gamma \vdash \Lambda X.t : \forall X.A$ *then* $(\Lambda X.t)'_\Gamma := \mathsf{emb}(\forall X.A')(\Lambda X.t'_\Gamma)$.
- *If* $\Gamma \vdash r : \forall X.A$ *then* $(rB)'_\Gamma := \mathsf{It}_\sharp^{(A[X:=B])'}(r'_\Gamma, \lambda z^{\forall X.A'}.zB^*, \mathsf{stab}\,(A[X := B])^*)$.
- *If* $\Gamma, a : A \vdash t : A$ *then* $(a\, t)'_{\Gamma,a:A} := q(a\, t'_{\Gamma,a:A})$.
- *If* $\Gamma \vdash \mu a^A.r : A$ *then* $(\mu a^A.r)'_\Gamma := \mathsf{stab}\, A^*(\lambda a^{\neg A'}.\, p\, r'_{\Gamma,a:A})$.

The first three defining clauses are taken from section 4.1, the fourth and fifth are the analogues for universal quantification. The last two rules use the terms q and p from Lemma 11 that gave a retract of $\bot$ into $\bot'$. (Note the simplification of the clause for μ in comparison with that of section 4.1.)

Remark 2 *If* $\Gamma \vdash \mu a^A.\, b\, t : A$ *then, for some type* B, *this comes from* $(b : B) \in \Gamma \cup \{(a : A)\}$ *and* $\Gamma, a : A \vdash t : B$. *Then,*

$$\begin{aligned} (\mu a^A.\, b\, t)'_\Gamma \quad &= \quad \mathsf{stab}\, A^*(\lambda a^{\neg A'}.\, p\,(q\,(b\, t'_{\Gamma,a:A}))) \\ &\longrightarrow^* \quad \mathsf{stab}\, A^*(\lambda a^{\neg A'}.\, b\, t'_{\Gamma,a:A}), \end{aligned}$$

by Lemma 11. The reduct thus achieved has originally been the definition of the translation of $\mu a^A.b\, t$ *in [Mat01]. It only treated* $\lambda\mu$*-calculus à la Parigot where* $\mu a^A.r$ *always has to be of the form* $\mu a^A.b\, t$. *It might be interesting to note that the present remark is the only place where the retract property in Lemma 11 is needed. Theorem 1 does not depend on it.*

Lemma 13 (Soundness) *If* $\Gamma \vdash r : A$ *then* $\Gamma' \vdash r'_\Gamma : A'$.

Proof By induction on $\Gamma \vdash r : A$. For variables, lambda abstraction and application, this has been done in section 4.1, for type application rB, one uses $(A[X := B])' = A'[X := B^*]$. The case of type abstraction $\Lambda X.t$ satisfies the proviso of the respective typing rule since the set of free type variables is not affected by the translation of types. The two rules involving μ-variables are easily treated by help of the induction hypothesis, e. g., $\Gamma', a : \neg A' \vdash (a\,t)'_{\Gamma,a:A} : \sharp\bot$ since $\Gamma', a : \neg A' \vdash a : \neg A'$ and $\Gamma', a : \neg A' \vdash t'_{\Gamma,a:A} : A'$, hence $\Gamma', a : \neg A' \vdash a\,t'_{\Gamma,a:A} : \bot$. □

Remark 3 (Church versus Curry) *The previous lemma would be incorrect for* $\lambda\mu$ *in a Curry-style typing discipline: There, from* $\Gamma \vdash t : A$ *and* X *not free in* Γ*, one would derive* $\Gamma \vdash t : \forall X.A$*—without changing the term* t*. Assume, we had* $\Gamma' \vdash t'_\Gamma : A'$ *by induction hypothesis. Then, we would be required to show* $\Gamma' \vdash t'_\Gamma : (\forall X.A)'$*. Certainly, we get* $\Gamma' \vdash t'_\Gamma : \forall X.\, A'$*. But* $(\forall X.A)' = \sharp(\forall X.\, A')$*, and the passage to the stabilization of a type cannot go unnoticed by the term system.*

As an attempt to resolve this problem, one might set $(\forall X.A)' := \forall X.\, A'$*. Among other nuisances, the rule of universal elimination becomes unsound: Assume* $\Gamma \vdash t : \forall X.\, X \to X$*. In Curry-style typing, this entails* $\Gamma \vdash t : (\forall Y.Y) \to (\forall Y.Y)$*—again, without changing the term* t*. Assume we knew already that* $\Gamma' \vdash t'_\Gamma : (\forall X.\, X \to X)'$*. Then, we would need to establish* $\Gamma' \vdash t'_\Gamma : ((\forall Y.Y) \to (\forall Y.Y))'$*. Obviously,* $(\forall X.\, X \to X)' = \forall X.\, \sharp(\sharp X \to \sharp X)$ *cannot be instantiated to* $((\forall Y.Y) \to (\forall Y.Y))' = \sharp((\forall Y.\, \sharp Y) \to (\forall Y.\, \sharp Y))$*. Hence, also this alternative approach fails. In section 5.3, it will be mentioned that typing à la Curry also excludes CPS-translations (for second-order systems).*

Before we can show that the translation also respects reduction, we need that the admissible typing rules of weakening and cut are respected by the prospective embedding.

Lemma 14

- *If* $\Gamma \vdash t : B$ *and* x *is not declared in* Γ*, then* $t'_\Gamma = t'_{\Gamma,x:A}$.
- *If* $\Gamma, x : A \vdash t : B$ *and* $\Gamma \vdash s : A$ *then* $(t[x := s])'_\Gamma = t'_{\Gamma,x:A}[x := s'_\Gamma]$.
- *If* $\Gamma \vdash t : B$ *then* $(t[X := A])'_{\Gamma[X:=A]} = t'_\Gamma[X := A^*]$.

Proof Easy (but tedious) induction. □

These rules will guarantee the simulation of the reduction rules of F, for μ-reductions, we need a lemma about the interaction of special substitution with our translation:

Lemma 15 *If* $\Gamma, a : A \vdash r : C$*, the term context* E *has* $\Gamma, z : A \vdash E[z] : B$ *and* b *is a "fresh"* μ*-variable, then*

$$r'_{\Gamma,a:A}\Big[a := \lambda z^{A'}.\, b\,(E[z])'_{\Gamma,z:A}\Big] \longrightarrow^* \Big(r[a\star := b\,E]\Big)'_{\Gamma,b:B}.$$

Proof By induction on r. The only interesting case is with $r = a\,t$, hence with the same μ-variable a. Then, $C = \bot$ and $\Gamma, a : A \vdash t : A$. Call the term to the left L and the term to the right R. Finally, the term substituted into L will be called S. Then,

$$L = q(a\,t'_{\Gamma,a:A})[a := S] = q\Big(S\,t'_{\Gamma,a:A}[a := S]\Big).$$

By induction hypothesis, $t'_{\Gamma,a:A}[a := S] \longrightarrow^* \Big(t[a\star := b\,E]\Big)'_{\Gamma,b:B}$. Therefore,

$$\begin{array}{rcl}
L & \longrightarrow^* & q\Big(S\,(t[a\star := b\,E])'_{\Gamma,b:B}\Big) \\
& = & q\Big((\lambda z^{A'}.\,b\,(E[z])'_{\Gamma,z:A})\,(t[a\star := b\,E])'_{\Gamma,b:B}\Big) \\
& \longrightarrow_\beta & q\Big(b\,(E[z])'_{\Gamma,z:A}\Big[z := (t[a\star := b\,E])'_{\Gamma,b:B}\Big]\Big) \\
& = & q\Big(b\,(E[z])'_{\Gamma,b:B,z:A}\Big[z := (t[a\star := b\,E])'_{\Gamma,b:B}\Big]\Big) \\
& = & q\Big(b\,\Big((E[z])\Big[z := t[a\star := b\,E]\Big]\Big)'_{\Gamma,b:B}\Big) \\
& = & q\Big(b\,\Big(E[t[a\star := b\,E]]\Big)'_{\Gamma,b:B}\Big) \\
R & = & \Big((a\,t)[a\star := b\,E]\Big)'_{\Gamma,b:B} \\
& = & \Big(b\,\Big(E[t[a\star := b\,E]]\Big)\Big)'_{\Gamma,b:B} \\
& = & q\Big(b\,\Big(E[t[a\star := b\,E]]\Big)'_{\Gamma,b:B}\Big)
\end{array}$$

□

Remark 4 *The previous lemma is the reformulation of Lemma 3 in [Mat01] for our slightly more general and more typing-aware situation.*

Lemma 16 (Simulation) *If $\Gamma \vdash r : A$ and $r \longrightarrow s$ then $r'_\Gamma \longrightarrow^+ s'_\Gamma$.*

Proof By induction on $r \longrightarrow s$. Since the defining clauses for r'_Γ never erase subterms—unlike several CPS-translations which therefore fail, see section 5.3—the closure under the term formation rules is straightforward (also use subject reduction for $\lambda\mu$). Hence, we only discuss the axiom schemes.

$\Gamma \vdash (\lambda x^A.t)s : B$ can only be derived from $\Gamma, x : A \vdash t : B$ and $\Gamma \vdash s : A$, and we have to show that $((\lambda x^A.t)s)'_\Gamma \longrightarrow^+ (t[x := s])'_\Gamma$. The left-hand side is

$$\mathsf{It}_\sharp^{B'}\Big(\mathsf{emb}(A' \to B')(\lambda x^{A'}.t'_{\Gamma,x:A}), \lambda z^{A'\to B'}.zs'_\Gamma, \mathsf{stab}\,B^*\Big).$$

One step by help of $\longrightarrow_\sharp$ yields $(\lambda z^{A'\to B'}.zs'_\Gamma)(\lambda x^{A'}.t'_{\Gamma,x:A})$, and two β-reduction steps lead to $t'_{\Gamma,x:A}[x := s'_\Gamma]$ which is the right-hand side by Lemma 14.

If $\Gamma \vdash (\Lambda X.t)B : A[X := B]$ has been derived from $\Gamma \vdash t : A$ and $X \notin \mathsf{FV}(\Gamma)$, then we have to show $((\Lambda X.t)B)'_\Gamma \longrightarrow^+ (t[X := B])'_\Gamma$. The left-hand side is

$$\mathsf{It}_\sharp^{(A[X:=B])'}\Big(\mathsf{emb}(\forall X.A')(\Lambda X.t'_\Gamma), \lambda z^{\forall X.A'}.zB^*, \mathsf{stab}\,(A[X := B])^*\Big).$$

It reduces by one step of $\longrightarrow_\sharp$ to $(\lambda z^{\forall X.A'}.zB^*)(\Lambda X.t'_\Gamma)$ and two more β-reduction steps to $t'_\Gamma[X := B^*]$. Since $X \notin \mathsf{FV}(\Gamma)$, hence $\Gamma[X := B] = \Gamma$, Lemma 14 ensures that this term is the right-hand side.

$\Gamma \vdash (\mu a^{A\to B}.r)s : B$ is necessarily derived from $\Gamma, a : A \to B \vdash r : \bot$ and $\Gamma \vdash s : A$. We have to show $((\mu a^{A\to B}.r)s)'_\Gamma \longrightarrow^+ (\mu b^B.r[a\star := b\,(\star\,s)])'_\Gamma$. The left-hand side is

$$\mathsf{It}_\sharp^{B'}\Big(\mathsf{stab}(A \to B)^*(\lambda a^{\neg(A\to B)'}.\,p\,r'_{\Gamma,a:A\to B}), \lambda z^{A'\to B'}.zs'_\Gamma, \mathsf{stab}\,B^*\Big).$$

One $\longrightarrow_\sharp$-reduction step yields

$$\mathsf{stab}\,B^*\Big(\lambda b^{\neg B'}.\,\underbrace{(\lambda a^{\neg(A\to B)'}.\,p\,r'_{\Gamma,a:A\to B})(\lambda z^{(A\to B)'}.b\,(zs)'_{\Gamma,z:A\to B})}_{\longrightarrow_\beta p\,r'_{\Gamma,a:A\to B}[a:=\lambda z^{(A\to B)'}.b\,(zs)'_{\Gamma,z:A\to B}]}\Big)$$

Using the previous lemma with $A \to B$ in place of A, $C := \bot$ and $E := \star s$, we get

$$r'_{\Gamma,a:A\to B}[a := \lambda z^{(A\to B)'}.b\,(zs)'_{\Gamma,z:A\to B}] \longrightarrow^* (r[a\star := b\,(\star s)])'_{\Gamma,b:B}$$

To conclude,

$$((\mu a^{A\to B}.r)s)'_\Gamma \longrightarrow^+ \mathsf{stab}\, B^* \Big(\lambda b^{\neg B'}.p\,(r[a\star := b\,(\star s)])'_{\Gamma,b:B}\Big).$$

The right-hand side is the one required above, by definition.

$\Gamma \vdash (\mu a^{\forall X.A}.r)\,B : A[X := B]$ is derived from $\Gamma, a : \forall X.A \vdash r : \bot$, and we have to show that $((\mu a^{\forall X.A}.r)\,B)'_\Gamma \longrightarrow^+ (\mu b^{A[X:=B]}.r[a\star := b\,(\star B)])'_\Gamma$. This is done analogously to the previous case, again by help of the previous lemma, this time with A replaced by $\forall X.A$ and B replaced by $A[X := B]$, and $E := \star B$. □

This concludes the proof of theorem 1 and hence the alternative proof of strong normalization of $\lambda\mu$.

5 Other Approaches and Extensions

In the first part, the embedding of $\mathsf{F}^\neg$ into F by Joly is presented. It is based on a very different idea than the one of using stabilization through system $\mathsf{I}\sharp$. The second part challenges Joly's method by the addition of positive fixed-points. The systems with these fixed-points are defined, the probable failure of Joly's method for them is discussed, the success of our method demonstrated, and a further extension by *least* fixed-points is dealt with. In the third part, we comment on the recent article by Nakazawa and Tatsuta [NT03] about failing attempts to prove strong normalization for second-order classical natural deduction by translations in continuation-passing style, and also on their proposed solution.

5.1 The Embedding by Joly

In this section, an easier embedding—even of $\mathsf{F}^\neg$ instead of $\lambda\mu$—into F by Joly [Jol97] is described in some detail, in order to show a further variation on embeddings, and to see its limitations, notably the fact that fixed-points cannot be treated (see the next section).

From the discussion in section 2.2, we know that a term of type $\neg\neg(\forall X.A) \to \forall X.A$ can be defined from a term witnessing stability of A. If X occurs free in A, this term possibly involves the constant stab^X. The idea to resolve this problem is the relativization of universal quantifiers $\forall X.A$ to stable X, i.e., $\forall X.A$ will be interpreted by $\forall X.\,(\neg\neg X \to X) \to A$ (and A will be transformed recursively). Then, a term witnessing stability for the interpretation of a type A can be defined from only the stability constants stab^X with X free in A. But there are no stability constants in F! They are even not needed because they can now be replaced by variables of type $\neg\neg X \to X$ for $X \in \mathsf{FV}(A)$. The freedom thus obtained allows Joly to get the substitution property we were lacking in section 2.2.

Definition 9 *By iteration on type A, define the type A′:*

$$\begin{aligned} X' &:= X \\ \bot' &:= \bot \\ (A \to B)' &:= A' \to B' \\ (\forall X.A)' &:= \forall X.\,(\neg\neg X \to X) \to A' \end{aligned}$$

Clearly, $\mathsf{FV}(A') = \mathsf{FV}(A)$ and $(A[X := B])' = A'[X := B']$.

Definition 10 (Environment) *Let an environment by any injective function ξ from the set* TV *of type variables into the set of all term variables such that countably many term variables are not in the range of ξ.*

Let always denote ξ, ξ_1, ξ_2 environments. Environments exist if we assume, e. g., that the type variable set and term variable set are countable.

Definition 11 (Modified Environment) *Let ξ be an environment, X be a type variable and u be a term variable not occurring in the range of ξ. Then, $\xi[X \mapsto u]$ is defined to be the environment that is equal to ξ for all variables except X, and such that $\xi[X \mapsto u](X) = u$.*

Let always denote $\xi[X \mapsto x]$ a modified environment, that is, the injectivity condition is tacitly assumed.

Definition 12 (Stability Witnesses) *By iteration on the type A, define the term stab^A_ξ:*

$$\begin{array}{ll}
\mathsf{stab}^X_\xi & := \xi(X) \\
\mathsf{stab}^\bot_\xi & := \lambda v^{\neg\neg\bot}.\, v\ \mathsf{id}^\bot \\
\mathsf{stab}^{A\to B}_\xi & := \lambda v^{\neg\neg(A\to B)'} \lambda w^{A'}.\ \mathsf{stab}^B_\xi \left(\lambda y^{\neg B'}.\, v\,(\lambda z^{(A\to B)'}.\, y\,(z\,w))\right) \\
\mathsf{stab}^{\forall X.A}_\xi & := \lambda v^{\neg\neg(\forall X.A)'} \Lambda X \lambda u^{\neg\neg X\to X}.\ \mathsf{stab}^A_{\xi[X\mapsto u]} \left(\lambda y^{\neg A'}.\, v\,(\lambda z^{(\forall X.A)'}.\, y\,(z\,X\,u))\right)
\end{array}$$

The definition of $\mathsf{stab}^{A\to B}_\xi$ is similar to that in section 2.2, the one for the universal quantifier is essentially different in its use of the variable u which also modifies the environment.

Lemma 17 (Coincidence) *Let $\xi_1(X) = \xi_2(X)$ for all $X \in \mathsf{FV}(A)$. Then $\mathsf{stab}^A_{\xi_1} = \mathsf{stab}^A_{\xi_2}$.* □

Corollary 2 $\mathsf{stab}^A_\xi = \mathsf{stab}^A_{\xi[X\mapsto u]}$ *for $X \notin \mathsf{FV}(A)$.* □

The following lemma states in which sense we defined "stability witnesses".

Lemma 18 (Correct Types) *Let $\mathsf{FV}(A) \subseteq \{X_1, \ldots, X_n\}$. Then*

$$\xi(X_1) : \neg\neg X_1 \to X_1, \ldots, \xi(X_n) : \neg\neg X_n \to X_n \vdash \mathsf{stab}^A_\xi : \neg\neg A' \to A'.$$

Proof Induction on A. □

Lemma 19 (Substitution) $\mathsf{stab}^{A[X:=B]}_\xi = \mathsf{stab}^A_{\xi[X\mapsto u]}[X := B'][u := \mathsf{stab}^B_\xi]$.

Proof Induction on A, using injectivity of environments in the case $A = Y \neq X$, and the Coincidence Lemma in the case of universal quantification. □

The definition of the translation of terms of $\mathsf{F}^\neg$ into those of F is always with respect to some environment.

Definition 13 *By iteration on the term r of $\mathsf{F}^\neg$, define the term r'_ξ of F:*

$$\begin{array}{ll}
x'_\xi & := x \\
(\lambda x^A.r)'_\xi & := \lambda x^{A'}.\, r'_\xi \\
(r\,s)'_\xi & := r'_\xi\, s'_\xi \\
(\Lambda X.t)'_\xi & := \Lambda X \lambda u^{\neg\neg X\to X}.\, r'_{\xi[X\mapsto u]} \\
(r\,B)'_\xi & := r'_\xi\, B'\ \mathsf{stab}^B_\xi \\
(\mu x^{\neg A}.r)'_\xi & := \mathsf{stab}^A_\xi\, (\lambda x^{\neg A'}.\, r'_\xi)
\end{array}$$

Definition 14 (Critical Variables) *By iteration on the term r of $\mathsf{F}^\neg$, define its (finite) set $\mathsf{CV}(r)$ of critical type variables:*

$$\begin{array}{rl}
\mathsf{CV}(x) & := \emptyset \\
\mathsf{CV}(\lambda x^A.r) & := \mathsf{CV}(r) \\
\mathsf{CV}(r\,s) & := \mathsf{CV}(r) \cup \mathsf{CV}(s) \\
\mathsf{CV}(\Lambda X.t) & := \mathsf{CV}(r) \setminus \{X\} \\
\mathsf{CV}(r\,B) & := \mathsf{CV}(r) \cup \mathsf{FV}(B) \\
\mathsf{CV}(\mu x^{\neg A}.r) & := \mathsf{FV}(A) \cup \mathsf{CV}(r)
\end{array}$$

This means, we record all free type variables that occur in type applications and uses of RAA.

Lemma 20 (Coincidence) *Let $\xi_1(X) = \xi_2(X)$ for all $X \in \mathsf{CV}(r)$. Then $r'_{\xi_1} = r'_{\xi_2}$.* $\square$

Our translation of $\mathsf{F}^{\neg}$ into F respects types in the following sense.

Lemma 21 (Correct Types) *Let $x_1 : A_1, \dots, x_m : A_m \vdash r : A$ in system $\mathsf{F}^{\neg}$, and assume $\mathsf{CV}(r) \subseteq \{X_1, \dots, X_n\}$. Then, we get in F that*

$$\xi(X_1) : \neg\neg X_1 \to X_1, \dots, \xi(X_n) : \neg\neg X_n \to X_n, x_1 : A'_1, \dots, x_m : A'_m \vdash r'_\xi : A'.$$

Proof Induction on r. $\square$

Lemma 22 (Substitution) *For x not in the range of ξ, one has $(r[x := s])'_\xi = r'_\xi[x := s'_\xi]$. Moreover, for u not free in r, one has*

$$(r[X := B])'_\xi = r'_{\xi[X \mapsto u]}[X := B'][u := \mathsf{stab}^B_\xi].$$

Proof Induction on r. $\square$

Lemma 23 (Simulation) *If $r \longrightarrow s$ in $\mathsf{F}^{\neg}$ then $r'_\xi \longrightarrow^+ s'_\xi$ in F.*

Proof Induction on $\longrightarrow$. Only the axiom schemes of reduction need to be studied since the term closure trivially works by using term closure rules again. The β-reduction rules of F are immediately dealt with by the previous lemma. The first μ-reduction rule—for implication—goes just as in the motivation in section 2.2, decorated with ξ and $-'$. The failed calculation for the μ-reduction rule for the universal quantifier is replaced by the following correct behaviour:

$$\begin{aligned}
&\Big((\mu x^{\neg\forall X.A}.r)\,B\Big)'_\xi \quad = \quad \mathsf{stab}^{\forall X.A}_\xi\,(\lambda x^{\neg(\forall X.A)'}.\,r'_\xi)\,B'\ \mathsf{stab}^B_\xi\\
&\longrightarrow^3 \underbrace{\mathsf{stab}^A_{\xi[X\mapsto u]}[X := B'][u := \mathsf{stab}^B_\xi]}_{\mathsf{stab}^{A[X:=B]}_\xi}\Big(\lambda y^{\neg A'[X:=B']}.\,(\lambda x^{\neg(\forall X.A)'}.\,r'_\xi)\,(\lambda z^{(\forall X.A)'}.\,y\,(z\,B'\ \mathsf{stab}^B_\xi))\Big)\\
&\longrightarrow \mathsf{stab}^{A[X:=B]}_\xi\Big(\lambda y^{\neg A[X:=B]'}.\,r'_\xi[x := \lambda z^{(\forall X.A)'}.\,y\,\underbrace{(z\,B'\ \mathsf{stab}^B_\xi)}_{(z\,B)'_\xi}]\Big)\\
&= \Big(\mu y^{\neg A[X:=B]}.\,r[x := \lambda z^{\forall X.A}.\,y\,(z\,B)]\Big)'_\xi, \text{ by the previous lemma.}
\end{aligned}$$

$\square$

By Lemma 21 and the previous lemma, $\mathsf{F}^{\neg}$ immediately inherits strong normalization from system F.

Note that these results do not constitute an embedding in the sense of any of the definitions 1, 2 or 5/6. The passage from the first to the second definition is driven by the need to refer to the typing when defining the translation of a term. In our present definition 13, this is clearly not called for. The definitions 5 and 6 are tailor-made for $\lambda\mu$-calculus. For the present embedding, we would have to allow a change in the typing context—even according to the critical variables. It does not look worthwhile introducing a general definition for so specific a case.

5.2 Extension by Fixed-Points

We extend $\lambda\mu$ by positive fixed-points, and get the system $\lambda\mu^{\mathsf{fix}}$: The inductive definition of the set of types is extended by the clause $\mathsf{fix}X.A$, with the proviso that X occurs only positively in A. More precisely, we define the set TP of types and for every type $A \in \mathsf{TP}$ the sets $\mathsf{TV}_+(A)$ and $\mathsf{TV}_-(A)$ of type variables that occur only positively in A or only negatively in A, respectively. Let always range p (polarity) over $\{-, +\}$ and set $-- := +$ and $-+ := -$.

- $X \in \mathsf{TP}$, $\mathsf{TV}_-(X) := \mathsf{TV} \setminus \{X\}$, $\mathsf{TV}_+(X) := \mathsf{TV}$.
- $\perp \in \mathsf{TP}$, $\mathsf{TV}_p(\perp) := \mathsf{TV}$.

- If $A, B \in \mathsf{TP}$, then $A \to B \in \mathsf{TP}$ and $\mathsf{TV}_p(A \to B) := \mathsf{TV}_{-p}(A) \cap \mathsf{TV}_p(B)$.
- If $A \in \mathsf{TP}$, then $\forall X.A \in \mathsf{TP}$ and $\mathsf{TV}_p(\forall X.A) := \mathsf{TV}_p(A) \cup \{X\}$.
- If $A \in \mathsf{TP}$ and $X \in \mathsf{TV}_+(A)$ then $\mathsf{fix}X.A \in \mathsf{TP}$ (only here is a positivity condition) and $\mathsf{TV}_p(\mathsf{fix}X.A) := \mathsf{TV}_p(A) \cup \{X\}$.

An important example is the impredicative definition of disjoint sums:

$$A + B := \forall X.(A \to X) \to (B \to X) \to X$$

for X not free in A or B. Trivially, if $A, B \in \mathsf{TP}$ then $A + B \in \mathsf{TP}$. Moreover, $\mathsf{TV}_p(A + B) = \mathsf{TV}_p(A) \cap \mathsf{TV}_p(B)$.

Our definition allows interleaving of fixed-points, e. g., $\mathsf{fix}X.\mathsf{fix}Y.X + Y \in \mathsf{TP}$ since $X \in \mathsf{TV}_+(\mathsf{fix}Y.X + Y)$. Note that this is more than just nesting of fixed-points: The outer fix binds the parameter X of the inner fixed-point. Here, we already use non-strict positivity, since $Y \in \mathsf{TV}_+(X + Y)$ is only derivable by help of TV_-. Had we not encoded sums but taken them as primitives, we would have nevertheless obtained genuine examples of non-strict positivity, e. g., $\mathsf{fix}\,X.A + \neg\neg X$ for $X \notin \mathsf{FV}(A)$ since $X \in \mathsf{TV}_+(\neg\neg X)$ which comes from $X \in \mathsf{TV}_-(\neg X)$ (which in turn rests on $X \in \mathsf{TV}_+(X)$). Note that $X \notin \mathsf{FV}(A)$ implies $X \in \mathsf{TV}_-(A) \cap \mathsf{TV}_+(A)$. In general, and loosely speaking, a non-strictly positive occurrence is to the left of an even number of $\to$. A strictly-positive occurrence is never to the left of $\to$.

The term system of $\lambda\mu$ is extended by fixed-point folding and unfolding, which yields the following term grammar for $\lambda\mu^{\mathsf{fix}}$:

$$r, s, t \quad ::= \quad \dots \mid \mathsf{in}_{X.A}\, t \mid \mathsf{out}_{X.A}\, r.$$

Variable binding for the indices $X.A$ is assumed as if they were $\lambda X.A$. The λ is just left out for stylistic reasons.

The definition of $\Gamma \vdash t : A$ is extended by the two clauses

$$\frac{\Gamma \vdash t : A[X := \mathsf{fix}X.A]}{\Gamma \vdash \mathsf{in}_{X.A}\, t : \mathsf{fix}X.A} \qquad \frac{\Gamma \vdash r : \mathsf{fix}X.A}{\Gamma \vdash \mathsf{out}_{X.A}\, r : A[X := \mathsf{fix}X.A]}$$

and there is a new axiom scheme of reduction

$$\mathsf{out}_{X.A}(\mathsf{in}_{X.A}\, t) \quad \longrightarrow_{\mathsf{fix}} \quad t$$

Moreover, there is a new μ-rule:

$$\mathsf{out}_{X.A}(\mu a^{\mathsf{fix}X.A}.\, r) \quad \longrightarrow_{\mu} \quad \mu b^{A[X:=\mathsf{fix}X.A]}.\, r[a \star := b\,(\mathsf{out}_{X.A} \star)]$$

It follows the pattern of the other two μ-reduction rules and clearly also enjoys subject reduction.

The new μ-reduction rule corresponds to a "reduction" of stability of $\mathsf{fix}X.A$ to that of $A[X := \mathsf{fix}X.A]$. The latter type can be more complex than the former which rules out the method by Joly, discussed in the previous section: The only candidate for $(\mathsf{fix}\,X.A)'$ is $\mathsf{fix}\,X.A'$. (A relativization to stable X as for the universal quantifier would not yield a positive type.) With Lemma 18 on the proper types for stab^A_ξ in mind, we would like to define $\mathsf{stab}^{\mathsf{fix}\,X.A}_\xi$ as

$$\lambda v^{\neg\neg(\mathsf{fix}\,X.A)'}.\, \mathsf{in}_{X.A'} \Big(\mathsf{stab}^{A[X:=\mathsf{fix}\,X.A]}_\xi \Big(\lambda y^{\neg A'[X:=\mathsf{fix}\,X.A']}.\, v\,(\lambda z^{(\mathsf{fix}\,X.A)'}.\, y\,(\mathsf{out}_{X.A'}\, z))\Big)\Big).$$

Clearly, this cannot be a definition: For $A = X$, this would mean that $\mathsf{stab}^{\mathsf{fix}\,X.X}_\xi$ is defined by an expression that involves $\mathsf{stab}^{\mathsf{fix}\,X.X}_\xi$ again. There is nothing like the solution to such fixed-point equations for *terms* in our systems. For more complex A, the situation would even be worse in that $A[X := \mathsf{fix}X.A]$ would be more complex than $\mathsf{fix}X.A$. The author does not envision a solution to this problem, which therefore limits the applicability of Joly's method.

System F with *non-interleaving* positive fixed-points essentially has been studied by Geuvers [Geu92] under the name F_{ret}, and strong normalization has been shown by him through an embedding into Mendler's system [Men87]. A direct proof of strong normalization by saturated sets has been given by the author [Mat99] under the name NPF. No embedding into system F exists [SU99]. One expects that this negative result also holds for any reasonable extension of the two systems by some η-rules.

Here, we show that strong normalization also holds with the reductions for classical logic and arbitrary positive fixed-points. Using a model construction involving saturated sets, this has been performed already in the respective extension of the Curry-style formulation of $\mathsf{F}^{\neg}$ by positive fixed-points (even in the presence of sums with permutative conversions) in [Mat04].

The point here is to give a straightforward extension of the embedding of $\lambda\mu$ into F in section 4.3 to an embedding of $\lambda\mu^{\mathsf{fix}}$ into $\mathsf{F}^{\mathsf{fix}}$. As expected, $\mathsf{F}^{\mathsf{fix}}$ shall denote the extension of F to the set TP of positive types, and with the additional axiom for $\longrightarrow_{\mathsf{fix}}$. System $\mathsf{F}^{\mathsf{fix}}$ is strongly normalizing, to be proven by an easy adaptation of the proof in [Mat99] or by omitting everything on RAA and sums in the proof in [Mat04].

To begin with, we also need the extension $\mathsf{I}\sharp^{\mathsf{fix}}$ of system $\mathsf{I}\sharp$ by positive fixed-points: For this, we have to stipulate that if A is a positive type, then so is $\sharp A$, and set $\mathsf{TV}_p(\sharp A) := \mathsf{TV}_p(A)$. The reduction axioms are those of $\mathsf{I}\sharp$, plus the above axiom for $\longrightarrow_{\mathsf{fix}}$.

It is obvious that the embedding in section 3.1 of $\mathsf{I}\sharp$ into F immediately extends to an embedding of $\mathsf{I}\sharp^{\mathsf{fix}}$ into $\mathsf{F}^{\mathsf{fix}}$: the fixed-point rules are translated homomorphically. (For the type translation, this is legitimate since positive and negative occurrences are maintained by it, especially by the definition of $(\sharp A)'$.) Hence, we are left with the task to extend Theorem 1 to $\lambda\mu^{\mathsf{fix}}$ and $\mathsf{I}\sharp^{\mathsf{fix}}$.

Definition 7 is extended by

$$(\mathsf{fix}\, X.A)^* := \mathsf{fix}\, X.\, \sharp A^*$$

In order to show that A^* is also a positive type, one simultaneously has to show the obvious fact that $\mathsf{TV}_p(A^*) = \mathsf{TV}_p(A)$. By induction on A, one gets $\mathsf{FV}(A^*) = \mathsf{FV}(A)$ and $(A[X := B])^* = A^*[X := B^*]$.

As before, the translation of types is defined by $A' := \sharp A^*$, hence

$$(\mathsf{fix}\, X.A)' = \sharp\, \mathsf{fix}\, X.\, A'$$

We also get the substitution property

$$(A[X := B])' = A'[X := B^*],$$

which instantiates to the crucial equation

$$(A[X := \mathsf{fix}\, X.A])' = A'[X := \mathsf{fix}\, X.A'].$$

Definition 8 is extended by

- If $\Gamma \vdash t : A[X := \mathsf{fix}\, X.A]$ then $(\mathsf{in}_{X.A}\, t)'_\Gamma := \mathsf{emb}(\mathsf{fix}\, X.A')(\mathsf{in}_{X.A'}\, t'_\Gamma)$.
- If $\Gamma \vdash r : \mathsf{fix}\, X.A$ then

$$(\mathsf{out}_{X.A}\, r)'_\Gamma := \mathsf{It}_\sharp^{(A[X:=\mathsf{fix}\, X.A])'}(r'_\Gamma, \lambda z^{\mathsf{fix}\, X.A'}.\, \mathsf{out}_{X.A'}\, z, \mathsf{stab}(A[X := \mathsf{fix}\, X.A])^*).$$

In a straightforward manner, the proofs of Lemma 13, Lemma 14 and Lemma 15 can be extended to the present systems. For simulation (Lemma 16), we again remark that the term translation never erases subterms. Consequently, only the simulation of the two new reduction rules within $\mathsf{I}\sharp^{\mathsf{fix}}$ has to be verified. For the rule for $\longrightarrow_{\mathsf{fix}}$, this is a trivial sequence of three reduction steps; the μ-reduction rule for the fixed-points has to be treated analogously to that for implication in the proof of Lemma 16.

Composing the two embeddings, we get the following:

Theorem 2 *There is a refined embedding of $\lambda\mu^{\mathsf{fix}}$ into $\mathsf{F}^{\mathsf{fix}}$. Hence, $\lambda\mu^{\mathsf{lfp}}$ is strongly normalizing.* $\square$

There is an objection to the usefulness of $\lambda\mu^{\mathsf{fix}}$, though: The fixed-point $\mathsf{fix}\,X.A$ is just meant to be an arbitrary solution to the informal fixed-point equation $F \simeq A[X := F]$. Recursion principles would need more information, namely, that we had the least or the greatest solution of that equation. As an example, we consider primitive recursion à la Mendler [Men87], which we extend

- to arbitrary least fixed-points—not just positive ones—following the observation made in [UV97] (journal version: [UV02]), and
- by the appropriate μ-reduction rule.

For our extension $\lambda\mu^{\mathsf{lfp}}$, we extend $\lambda\mu$ by types of the form $\mathsf{lfp}\,X.A$ for arbitrary X and A. That they model least fixed-points, comes from the new term and typing rules: Extend the term grammar by

$$r, s, t \quad ::= \quad \ldots \mid \mathsf{in}_{X.A}\, t \mid \mathsf{MRec}^{B}_{X.A}(r, s),$$

and the typing rules by

$$\frac{\Gamma \vdash t : A[X := \mathsf{lfp}X.A]}{\Gamma \vdash \mathsf{in}_{X.A}\, t : \mathsf{lfp}X.A} \qquad \frac{\Gamma \vdash r : \mathsf{lfp}X.A \quad s : \forall X.\,(X \to \mathsf{lfp}\,X.A) \to (X \to B) \to A \to B}{\Gamma \vdash \mathsf{MRec}^{B}_{X.A}(r, s) : B}$$

Clearly, the introduction rule is just taken from $\lambda\mu^{\mathsf{fix}}$, but the rule for the Mendler recursor MRec is new (to our discourse). The reduction axiom for primitive recursion is

$$\mathsf{MRec}^{B}_{X.A}(\mathsf{in}_{X.A}\, t, s) \quad \longrightarrow_{\mathsf{lfp}} \quad s\,(\mathsf{lfp}\,X.A)\ \mathsf{id}^{\mathsf{lfp}\,X.A} \left(\lambda x^{\mathsf{lfp}\,X.A}.\,\mathsf{MRec}^{B}_{X.A}(x, s)\right) t$$

For explanations, consult [UV02, p. 326] or [Mat98, chapter 6]. (Very simple instances of the schema are the primitive recursive functionals of Gödel's system T.)

The new μ-reduction rule is as expected:

$$\mathsf{MRec}^{B}_{X.A}(\mu a^{\mathsf{lfp}X.A}.\, r, s) \quad \longrightarrow_{\mu} \quad \mu b^{B}.\, r[a\,\star := b\,(\mathsf{MRec}^{B}_{X.A}(\star, s))]$$

Note, however, that there is no relation whatsoever between the type B and $\mathsf{lfp}X.A$, hence no "reduction" of stability in any sense. Nevertheless, we get the following theorem:

Theorem 3 *There is a refined embedding of $\lambda\mu^{\mathsf{lfp}}$ into $\mathsf{F}^{\mathsf{fix}}$. Hence, $\lambda\mu^{\mathsf{lfp}}$ is strongly normalizing.*

Proof By the previous theorem, it suffices to find an embedding of $\lambda\mu^{\mathsf{lfp}}$ into $\lambda\mu^{\mathsf{fix}}$. All type formation rules are treated homomorphically, with exception of $\mathsf{lfp}\,X.A$. If already A has been translated to A', then $\mathsf{lfp}\,X.A$ is translated to $(\mathsf{lfp}\,X.A)' := \mathsf{fix}\,Y.\,\hat{A}$, with

$$\hat{A} := \forall Z.\,(\forall X.\,(X \to Y) \to (X \to Z) \to A' \to Z) \to Z.$$

This definition can be obtained by applying the propositions 1, 5 and 9 in [UV02] and some simple isomorphisms in order to get rid of existential quantification and products.

The interesting clauses for the term translation are:

- $(\mathsf{MRec}^{B}_{X.A}(r, s))' := \mathsf{out}_{Y.\hat{A}}\, r'\, B'\, s'$
- $(\mathsf{in}_{X.A}\, t)' := \mathsf{in}_{Y.\hat{A}}\Big(\Lambda Z \lambda z^{\forall X.\,(X \to (\mathsf{lfp}\,X.A)') \to (X \to Z) \to A' \to Z}.$
  $$z\,(\mathsf{lfp}\,X.A)'\ \mathsf{id}^{(\mathsf{lfp}\,X.A)'} \left(\lambda x^{(\mathsf{lfp}\,X.A)'}.\,(\mathsf{MRec}^{Z}_{X.A}(x, z))'\right) t'\Big)$$

It is fairly easy to check the statement on types in Definition 1 for this setting. Also, the compatibility of the term translation with the notions of term substitution, type substitution and special substitution is easily established. Given these compatibilities, simulation for the two reduction rules concerning MRec is just a matter of calculation. In both cases, three reduction steps lead from the translation of the left-hand side to that of the right-hand side; for the μ-reduction rule, it is just three μ-reductions, pertaining to the application of $\mathsf{out}_{Y.\hat{A}}$, of B' and of s'. $\square$

Note that it seems impossible to extend the translation by stabilization from section 4.3 by these least fixed-points (also in the target system $\mathsf{I}\sharp$). So, we made essential use of impredicativity, which cannot be too inconvenient since Mendler's style anyway rests on impredicativity. For formulations of primitive recursion in the style of universal algebra (see, e. g., [Geu92]), see the treatment and discussion in [Mat01, section 7].

5.3 Embeddings in Continuation-Passing Style?

Nakazawa and Tatsuta [NT03] show that a number of published proofs of strong normalization of classical natural deduction by translations in continuation-passing style (CPS) fail. They explicity mention [Par97, dG95, Fuj00].

It should be recalled that CPS translations do not model the μ-reductions but only allow to deduce strong normalization from that of F and separately of the μ-reductions (which, according to [Par97], strongly normalize even in the untyped case).

In the cited articles, the term closure does not preserve simulation of reduction steps. This is due to "erasing-continuations" [NT03]: Vacuous μ-abstractions may devour an argument term, hence ordinary β-reductions in that argument are not simulated but erased.

In more recent work [dG01], de Groote distinguishes the vacuous abstraction from the other ones and gives different translations, involving β-expansions in the vacuous case. Unfortunately, the case distinction is not invariant under β-reduction: μ-variables may get lost. Simulation fails, e. g., for $\mu a^A.\, (\lambda z^\perp.\, x)\,(a\, y) \longrightarrow \mu a^A.\, x$ (with x, y, z different) in that the translated terms only have a common reduct.

One would have hoped to see the promised proof by "optimized CPS-translation" of strong normalization of domain-free classical pure type systems [BHS97, section 6.1], but there does not seem to be a full version of that article.

Nakazawa and Tatsuta give a new CPS-translation that simulates β-reductions and erases μ-reductions, as expected from CPS-translations. They use the notion of augmentation, which gives rise to a quite complicated form of embedding. Unfortunately, the whole article treats a formulation of second-order $\lambda\mu$-calculus with typing à la Curry. As for the embedding of this article (see remark 3) and for the CPS-translation in [dG94, Proposition 5.2] (see the remarks between the statement and the proof), this does not work. In [NT03], Proposition 4.6 fails. A counterexample (in their notation) is provided by the typing $\lambda x.x : \emptyset \vdash \forall X.\, X \to X, \emptyset$. The lemma claims that $(\lambda x.x)^* : (f : \neg(\forall X.\, X \to X)^*) \vdash \perp$. In our notation, this amounts to the typing $f : \neg\forall X.\, \neg\neg(\neg\neg X \to \neg\neg X) \vdash f\,(\lambda x \lambda g.\, xg) : \perp$, which is certainly impossible in Curry-style system F.

It seems highly plausible that the Church-style formulation can be treated by their method (a draft has been provided by Nakazawa and Tatsuta after the present author informed them of this counterexample). But their term translation is more demanding than ours: Simulation is only proven for a clever choice among an infinite set of "augmentations" for every term. In effect, a hypothetical reduction sequence in $\lambda\mu$ which contains infinitely many β-reduction steps is translated into one in F, where the choices depend on the earlier reduction steps. This works well for the sake of inheriting strong normalization, but does not seem to be very explicative.

The author conjectures that the method by Nakazawa and Tatsuta will smoothly extend to fixed-points, as introduced in the previous section.

6 Conclusion

A new translation of classical second-order logic into intuitionistic second-order logic has been described: Stabilization. It has been put forward as an alternative to double-negation translations (translations in continuation-passing style). It is also faithful to the operational behaviour of those systems, and therefore allows to infer strong normalization of the classical systems from the intuitionistic systems.

For the pure system, also the embedding by Joly is available, but does not seem to extend to fixed-points, which is hence a true application of our stabilization method.

Proofs of strong normalization of classical second-order logic in the literature by double-negation translations seem to fail altogether, but there is the recent work by Nakazawa and Tatsuta which does perhaps not fully qualify as an "embedding".

References

[BB85] Corrado Böhm and Alessandro Berarducci. Automatic synthesis of typed λ-programs on term algebras. *Theoretical Computer Science*, 39:135–154, 1985.

[BHS97] Gilles Barthe, John Hatcliff, and Morten Heine Sørensen. A notion of classical pure type system (preliminary version). In S. Brookes and M. Mislove, editors, *Proceedings of the Thirteenth Conference on the Mathematical Foundations of Programming Semantics*, volume 6 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 1997. 56 pp.

[dG94] Philippe de Groote. A CPS-translation of the $\lambda\mu$-calculus. In Sophie Tison, editor, *Trees in Algebra and Programming - CAAP'94, 19th International Colloquium*, volume 787 of *Lecture Notes in Computer Science*, pages 85–99, Edinburgh, 1994. Springer Verlag.

[dG95] Philippe de Groote. A simple calculus of exception handling. In Mariangiola Dezani-Ciancaglini and Gordon Plotkin, editors, *Proceedings of the Second International Conference on Typed Lambda Calculi and Applications (TLCA '95), Edinburgh, United Kingdom, April 1995*, volume 902 of *Lecture Notes in Computer Science*, pages 201–215. Springer Verlag, 1995.

[dG01] Philippe de Groote. Strong normalization of classical natural deduction with disjunction. In Samson Abramsky, editor, *Proceedings of TLCA 2001*, volume 2044 of *Lecture Notes in Computer Science*, pages 182–196. Springer Verlag, 2001.

[Fuj00] Ken-Etsu Fujita. Domain-free $\lambda\mu$-calculus. *RAIRO - Theoretical Informatics and Applications*, 34:433–466, 2000.

[Geu92] Herman Geuvers. Inductive and coinductive types with iteration and recursion. In Bengt Nordström, Kent Pettersson, and Gordon Plotkin, editors, *Proceedings of the Workshop on Types for Proofs and Programs, Båstad, Sweden*, pages 193–217, 1992. Only published via `ftp://ftp.cs.chalmers.se/pub/cs-reports/baastad.92/proc.dvi.Z`.

[Gir72] Jean-Yves Girard. *Interprétation fonctionnelle et élimination des coupures dans l'arithmétique d'ordre supérieur*. Thèse de Doctorat d'État, Université de Paris VII, 1972.

[Jol97] Thierry Joly. An embedding of 2nd order classical logic into functional arithmetic FA2. *C. R. Acad. Sci. Paris, Série I*, 325:1–4, 1997.

[Mat98] Ralph Matthes. *Extensions of System* F *by Iteration and Primitive Recursion on Monotone Inductive Types*. Doktorarbeit (PhD thesis), University of Munich, 1998. Available via the homepage `http://www.tcs.informatik.uni-muenchen.de/~matthes/`.

[Mat99] Ralph Matthes. Monotone fixed-point types and strong normalization. In Georg Gottlob, Etienne Grandjean, and Katrin Seyr, editors, *Computer Science Logic, 12th International Workshop, Brno, Czech Republic, August 24–28, 1998, Proceedings*, volume 1584 of *Lecture Notes in Computer Science*, pages 298–312. Springer Verlag, 1999.

[Mat01] Ralph Matthes. Parigot's second order $\lambda\mu$-calculus and inductive types. In Samson Abramsky, editor, *Proceedings of TLCA 2001*, volume 2044 of *Lecture Notes in Computer Science*, pages 329–343. Springer Verlag, 2001.

[Mat04] Ralph Matthes. Non-strictly positive fixed-points for classical natural deduction. *Annals of Pure and Applied Logic*, 2004. Accepted for publication.

[Men87] Nax P. Mendler. Recursive types and type constraints in second-order lambda calculus. In *Proceedings of the Second Annual IEEE Symposium on Logic in Computer Science, Ithaca, N.Y.*, pages 30–36. IEEE Computer Society Press, 1987.

[NT03] Koji Nakazawa and Makoto Tatsuta. Strong normalization proof with CPS-translation for second order classical natural deduction. *The Journal of Symbolic Logic*, 68(3):851–859, 2003.

[Par92] Michel Parigot. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In Andrei Voronkov, editor, *Logic Programming and Automated Reasoning, International Conference LPAR'92, St. Petersburg, Russia*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201. Springer Verlag, 1992.

[Par93] Michel Parigot. Strong normalization for second order classical natural deduction. In *Proceedings, Eighth Annual IEEE Symposium on Logic in Computer Science*, pages 39–46, Montreal, Canada, 1993. IEEE Computer Society Press.

[Par97] Michel Parigot. Proofs of strong normalisation for second order classical natural deduction. *The Journal of Symbolic Logic*, 62(4):1461–1479, 1997.

[PM96] Christine Paulin-Mohring. *Définitions Inductives en Théorie des Types d'Ordre Supérieur.* Habilitation à diriger les recherches, Université Claude Bernard Lyon I, 1996.

[Pra65] Dag Prawitz. *Natural Deduction. A Proof-Theoretical Study.* Almquist and Wiksell, 1965.

[Pra71] Dag Prawitz. Ideas and results in proof theory. In Jens E. Fenstad, editor, *Proceedings of the Second Scandianvian Logic Symposium*, pages 235–307. North–Holland, Amsterdam, 1971.

[RS94] Jakob Rehof and Morten Heine Sørensen. The λ_Δ-calculus. In Masami Hagiya and John C. Mitchell, editors, *Theoretical Aspects of Computer Software, International Conference TACS '94, Sendai, Japan, Proceedings*, volume 789 of *Lecture Notes in Computer Science*, pages 516–542. Springer Verlag, 1994.

[SU99] Zdzisław Spławski and Paweł Urzyczyn. Type Fixpoints: Iteration vs. Recursion. *SIGPLAN Notices*, 34(9):102–113, 1999. Proceedings of the 1999 International Conference on Functional Programming (ICFP), Paris, France.

[Tai75] William W. Tait. A realizability interpretation of the theory of species. In R. Parikh, editor, *Logic Colloquium Boston 1971/72*, volume 453 of *Lecture Notes in Mathematics*, pages 240–251. Springer Verlag, 1975.

[Tak95] Masako Takahashi. Parallel reduction in λ-calculus. *Information and Computation*, 118(1):120–127, 1995.

[UV97] Tarmo Uustalu and Varmo Vene. A cube of proof systems for the intuitionistic predicate μ-, ν-logic. In M. Haveraaen and O. Owe, editors, *Selected Papers of the 8th Nordic Workshop on Programming Theory (NWPT '96), Oslo, Norway, December 1996*, volume 248 of *Research Reports, Department of Informatics, University of Oslo*, pages 237–246, May 1997.

[UV02] Tarmo Uustalu and Varmo Vene. Least and greatest fixed points in intuitionistic natural deduction. *Theoretical Computer Science*, 272:315–339, 2002.