The Journal of Functional and Logic Programming

The MIT Press

Volume 2000, Article 1
31 January 2000

ISSN 1080–5230. MIT Press Journals, Five Cambridge Center, Cambridge, MA 02142-1493, USA; (617)253-2889; *journals-orders@mit.edu, journals-info @mit.edu*. Published one article at a time in LaTeX source form on the Internet. Pagination varies from copy to copy. For more information and other articles see:

- *http://www.cs.tu-berlin.de/journal/jflp/*
- *http://mitpress.mit.edu/JFLP/*
- *gopher.mit.edu*
- *ftp://mitpress.mit.edu/pub/JFLP*

The Journal of Functional and Logic Programming is a peer-reviewed and electronically published scholarly journal that covers a broad scope of topics from functional and logic programming. In particular, it focuses on the integration of the functional and the logic paradigms as well as their common foundations.

Electronic Mail: *jflp.request@ls5.informatik.uni-dortmund.de*

Metatheoretic Results for a Modal λ-Calculus

Joelle Despeyroux
Joelle.Despeyroux@inria.fr

Pierre Leleu
LeleuP@wanadoo.fr

31 January 2000

Abstract

This paper presents the proofs of the strong normalization, subject reduction, and Church-Rosser theorems for a presentation of the intuitionistic modal λ-calculus S4. It is adapted from Healfdene Goguen's thesis, where these properties are shown for the simply typed λ-calculus and for Luo's type theory UTT. Following this method, we introduce the notion of typed operational semantics for our system. We define a notion of typed substitution for our system, which has context stacks instead of the usual contexts. This latter peculiarity leads to the main difficulties and consequently to the main original features in our proofs. The techniques elaborated in this work have already been found useful in recent works [DL98, DL99] and should be further exploited to prove the properties of other systems based on modality.

1 Introduction

We present here proofs of metatheoretic results for the modal λ-calculus IS4 (see, for example, [Che90] for a classification of modal logics), in the presentation by Frank Pfenning and Hao-Chi Wong [PW95]. We have chosen this variant because the terms generated by the syntax are simpler than those of [BdP96], [PW95], or [DP96], since we have no "let" construction.

The proof follows Healfdene Goguen's method [Gog94, Gog95] of "typed operational semantics." It appears to be surprisingly difficult to extend the method to the modal case, because the type system we choose uses context stacks. In particular, we have to extend the notion of typed substitution, which is central to the method, in a nontrivial way.

As a result, we get proofs of the strong normalization, subject reduction, and Church-Rosser theorems for our modal calculus IS4. These results are already well known. They can be proved fairly easily by interpreting modal λ terms into simply typed λ terms. However, we cannot expect to extend this interpretation method to a stronger calculus, since it relies on the existence of a corresponding nonmodal λ-calculus, which does not exist for the calculi we had in mind (refer to [DPS97, DL98]). A simple method of proof of normalisation was developed in [PW95], using "approximate typing." Here again, this method is unlikely to apply smoothly to a system with reduction rules (as opposed to a particular strategy for reduction) and with dependent types.

The main motivation of this work stems from the fact that modality enables the definition of primitive recursive functions over types defined with higher-order abstract syntax (HOAS) (cf. [DPS97, DL98]). The point is that if we are not careful, primitive recursive functions may "pollute" the function space and create "exotic" terms that do not represent any entity at the object level. Fortunately, the modal connector $\Box$ enables us to distinguish the types of functions in general (including the primitive recursive functions), $\Box A \to B$, from the types of functions of the simply typed λ-calculus (the "parametric" functions), $A \to B$.

In this context, the adaptation of Goguen's method to the modal setting seems promising: Since Goguen extends his proof to an inductive setting (namely, UTT), it seems that we should be able to extend this proof to a type system with modality and recursion over HOAS types. Although we do not define a typed operational semantics for the system presented in [DL98], some of the key notions introduced in that paper prove to be useful (mainly, typed substitutions, interpretations of modal types, and interpretations of context stacks).

The structure of the paper follows the plan outlined by Goguen in his method. We first introduce the type system for our calculus, and then we give the reduction rules and the typed operational semantics (TOS). The next section presents metatheoretical results for the TOS. At this point we prove that if a term has a reduction in the TOS, then it verifies the subject reduction, strong normalization, and Church-Rosser properties. Then a soundness result—namely, if a term is well typed, then it reduces to a normal form in the TOS—enables us to transfer these properties onto the original type system. Finally, we give in the appendix a technical development of a tricky subpart of the proof.

2 The type system

We present here Pfenning and Wong's system of modal λ-calculus IS4 [PW95]. Then, we recall briefly its basic properties.

2.1 Syntax

The sets of terms and contexts are generated by the usual syntax, with an additional definition for context stacks, as follows:

$$
\begin{array}{lll}
\text{Types} & A & ::= c \mid A \to A' \mid \Box A \\
\text{Terms} & t & ::= x \mid \lambda x : A.t' \mid (t\ t') \mid \uparrow t \mid \downarrow t \\
\text{Contexts} & \Gamma & ::= . \mid \Gamma, x : t \\
\text{Context stacks} & \Delta & ::= . \mid \Delta; \Gamma
\end{array}
$$

We use c for type constants. Instead of being declared in simple contexts, variables are declared in context stacks, that is, ordered lists of contexts. We sometimes use D to range over contexts.

Notation A *valid* context (resp., context stack) is a context (resp., context stack) where all the variables are distinct. We speak of the *domain* of a context Γ (resp., a stack Δ), denoted by $\mathrm{dom}(\Gamma)$ (resp., $\mathrm{dom}(\Delta)$), to refer to the set of the variables declared in this context (resp., stack). The notation Δ, Γ, where Δ is a stack $.; \Gamma_1; \ldots; \Gamma_n$ and Γ is a context, is the stack $.; \Gamma_1; \ldots; \Gamma_n, \Gamma$. Similarly, the notation Δ, Δ', where Δ is the stack $.; \Gamma_1; \ldots; \Gamma_n$ and Δ' is the stack $.; \Gamma'_1; \ldots; \Gamma'_m$, denotes the stack $.; \Gamma_1; \ldots; \Gamma_n, \Gamma'_1; \ldots; \Gamma'_m$. The notation $\Delta; \Delta'$, where Δ is the stack $.; \Gamma_1; \ldots; \Gamma_n$ and Δ' is the stack $.; \Gamma'_1; \ldots; \Gamma'_m$, denotes the stack $.; \Gamma_1; \ldots; \Gamma_n; \Gamma'_1; \ldots; \Gamma'_m$.

Examples

- $(.; x : A; .); (.; z : C; u : D) = (.; x : A; .; z : C; u : D)$
- $(.; x : A; y : B), (.; z : C; u : D) = (.; x : A; y : B, z : C; u : D)$
- $(.; x : A; .), (.; z : C; u : D) = (.; x : A; z : C; u : D)$

- $(.; x : A; .), (.; .; z : C) = (.; x : A; .; z : C)$

Note Instead of requiring a context stack to be valid in the Var and Pop rules below, in [PW95] it is assumed by default that any variable can only be declared at most once in a context stack.

The $\uparrow$ operator introduces an object of type $\Box A$, while the $\downarrow$ operator marks the elimination of a term of type $\Box A$. The last context Γ_n of a stack $\Delta \equiv .; \Gamma_1; \ldots; \Gamma_n$ is called the *local context* of Δ.

2.2 Typing rules

We have one typing judgment, $\Delta \vdash M : A$, which is taken to mean that the canonical (or normal) form of M is an element of type A in context stack Δ. In short, we say that "M has type A in stack Δ." The idea is to begin a new segment of context each time we encounter a $\uparrow$ operator during type checking (rule $\uparrow$). A context can only be popped (i.e., added to the current stack) when type checking a subterm of type $\Box$ (rule Pop). The complete system is as follows:

$$(\text{Var})\ \frac{x : A \in \Gamma}{\Delta; \Gamma \vdash x : A}\ \Delta; \Gamma \text{ valid}$$

$$(\lambda)\ \frac{\Delta, x : A \vdash M : B}{\Delta \vdash \lambda x : A.M \ : \ A \to B} \qquad (\text{App})\ \frac{\Delta \vdash M : A \to B \qquad \Delta \vdash N : A}{\Delta \vdash (M\ N) : B}$$

$$(\uparrow)\ \frac{\Delta; . \vdash M : A}{\Delta \vdash \uparrow M : \Box A} \qquad (\downarrow)\ \frac{\Delta \vdash M : \Box A}{\Delta \vdash \downarrow M : A}$$

$$(\text{Pop})\ \frac{\Delta \vdash M : \Box A}{\Delta; \Gamma \vdash M : \Box A}\ \Delta; \Gamma \text{ valid}$$

Because of the (Pop) rule, this system is not syntax-driven; that is, given a judgment $\Delta \vdash M : A$, we cannot guess which typing rule was the last one applied. Indeed, if $\Delta \vdash M : \Box B$ and if Δ is made of at least two contexts, the last rule applied can be (Pop) as well as the structural rule corresponding to the form of M.

Examples of derivation If $\Gamma = f : (\Box(A \to B), x : \Box A)$, we have the following:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash f : \Box(A \to B)}{\Gamma; . \vdash f : \Box(A \to B)}}{\Gamma; . \vdash \downarrow f : A \to B} \qquad \dfrac{\dfrac{\Gamma \vdash x : \Box A}{\Gamma; . \vdash x : \Box A}}{\Gamma; . \vdash \downarrow x : A}}{\Gamma; . \vdash (\downarrow f \quad \downarrow x) : B}}{\Gamma \vdash \uparrow(\downarrow f \quad \downarrow x) : \Box B} \qquad \dfrac{\dfrac{x : \Box A \vdash x : \Box A}{x : \Box A; . \vdash x : \Box A}}{x : \Box A \vdash \uparrow x : \Box\Box A}$$

If $\Gamma = (f : A \to \Box B, x : A)$, then the following holds:

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash f : A \to \Box B \qquad \Gamma \vdash x : A}{\Gamma \vdash (f \quad x) : \Box B}}{\Gamma; . \vdash (f \quad x) : \Box B}}{\Gamma \vdash \uparrow(f \quad x) : \Box\Box B}$$

Note that in a naive formulation of modal λ-calculus, where the rule of introduction for connector $\Box$ would be

$$\dfrac{\Box\Gamma \vdash M : A}{\Box\Gamma \vdash \uparrow M : \Box A}$$

the latter derivation would be impossible. Thus we have the need for more complex rules with "let" constructs, for instance (cf. [BdP96]). Following Pfenning and Wong, we instead choose to capture the modal mechanism with context stacks. Each stack can be seen as a world in Kripke's semantics of modal logic.

The temporal interpretation of modal logic IS4 is useful for grasping the intuition behind the connector $\Box$. Intuitively, "$\Box A$" means that A is *always* true. Context stacks are stacks of hypotheses ordered from the past to the present (local context). Thus, the rule ($\uparrow$) says that if we can prove A with no hypotheses from the present, then we always can prove A (i.e., $\Box A$). The rule ($\downarrow$) is very simple: If A is always true, then A is true. Finally, rule (Pop) states that if A is proved to be always true at a certain moment in time (with hypotheses Δ), then it is always true in the future of this moment (hypotheses $\Delta; \Gamma$).

Note It is easy to see that, because of their structure, terms of the form $((\uparrow M)\ N)$ and $\downarrow \lambda x : A.M$ are not well typed in any valid stack. Indeed, if $((\uparrow M)\ N)$ were well typed in a stack Δ, $\uparrow M$ could only have a type of the form $\Box A$ but at the same time $\uparrow M$ would have a type of the form $B \to C$ since it is on the left side of an application. This is a contradiction. A similar argument works for $\downarrow \lambda x : A.M$.

2.3 Basic properties

The structure of a context stack is more complex than the structure of a simple context. The following lemmas express the basic stack manipulations that preserve well-typedness. Namely, if $\Delta; \Gamma \vdash M : A$, then M is still well typed in a context stack where we have permuted declarations in a context, tranformed some semicolons into commas, removed unnecessary variables, and so on. These lemmas are all easily proved by induction on the structure of the proof of the hypothesis.

Lemma 1 (Swapping) *Two variables of any context in a stack can be swapped arbitrarily:*

$$(Swap)\ \frac{.; D_1; \ldots ; \Gamma, x : B, \Gamma', y : C, \Gamma''; \ldots ; D_n \vdash M : A}{.; D_1; \ldots ; \Gamma, y : C, \Gamma', x : B, \Gamma''; \ldots ; D_n \vdash M : A}$$

Lemma 2 (Thinning) *A fresh variable can be added anywhere in a stack:*

$$(Thin)\ \frac{.; D_1; \ldots ; D_i \ldots ; D_n \vdash M : A}{.; D_1; \ldots ; (D_i, x : B); \ldots ; D_n \vdash M : A}$$

provided that, for $1 \leq i \leq n, x \notin dom(.; D_1; \ldots ; D_n)$.

Lemma 3 (Modal weakening) *A "fresh" context (i.e., a context whose domain only contains fresh variables) can be added anywhere in a stack, except behind the local context:*

$$(Weak)\ \frac{.; D_1; \ldots ; D_i; D_{i+1}; \ldots ; D_n \vdash M : A}{.; D_1; \ldots ; D_i; D; D_{i+1}; \ldots ; D_n \vdash M : A}$$

provided that $.; D_1; \ldots ; D_i; D; D_{i+1}; \ldots ; D_n$ *is valid* $(1 \leq i < n)$.

The declaration of a variable that does not appear as a fresh variable in the term to type is somewhat superfluous. The following lemma tells us that we are allowed to remove it from the stack, regardless of its location in the stack.

Lemma 4 (Strengthening) *The following rule is admissible:*

$$(Strength)\ \frac{.; D_1; \ldots ; \Gamma, x : B, \Gamma'; \ldots ; D_n \vdash M : A}{.; D_1; \ldots ; \Gamma, \Gamma'; \ldots ; D_n \vdash M : A} \quad x \notin FV(M).$$

Lemma 5 (Fusion) *Two successive contexts of a stack can be merged:*

$$(Fus)\ \frac{.; D_1; \ldots ; D_i; D_{i+1}; \ldots ; D_n \vdash M : A}{.; D_1; \ldots ; (D_i, D_{i+1}); \ldots ; D_n \vdash M : A} \quad \textit{for } 1 \leq i < n.$$

In particular, if $\Delta; \Gamma; . \vdash M : A$, then $\Delta; \Gamma \vdash M : A$. Conversely, we have $.; x : A \vdash x : A$ but not $.; x : A; . \vdash x : A$ in general (actually $.; x : A; . \vdash x : A$ if and only if A is of the form $\Box B$).

Note that, in general, splitting a context of a stack into two separate contexts does not preserve typing. For example, it is true that $.; f : c \to \Box c, x : c; . \vdash (f\ x) : \Box c$. However, we do not have $.; f : c \to \Box c; x : c; . \vdash (f\ x) : \Box c$.

2.4 Substitution

We denote the substitution of N for the free variable x in M by $M[N/x]$. It is defined as usual to avoid the capture of free variables. The rules for the modal operators are as expected:

- $(\downarrow M)[P/x] \equiv \downarrow(M[P/x])$,
- $(\uparrow M)[P/x] \equiv \uparrow(M[P/x])$.

Lemma 6 (Admissibility of Subst) *The following rule is admissible:*

$$(Subst)\ \frac{.; D_1; \ldots ; \Gamma, x : B, \Gamma'; \ldots ; D_n \vdash M : A \qquad .; D_1; \ldots ; \Gamma \vdash N : B}{.; D_1; \ldots ; \Gamma, \Gamma'; \ldots ; D_n \vdash M[N/x] : A}$$

Proof The proof is by induction on the derivation of the first premise. □

2.5 Inversion lemmas

We end this section by giving the inversion lemmas for our typing rules. The inversion lemmas tell us how to type the immediate subterms of a well-typed term. Because of the nondeterminism of the typing system, we cannot always find with certainty which typing rule was the last one applied. First we give a definition for truncated context stacks.

Definition 7 *For a context stack* Δ *with* $n+1$ *contexts, we define the context stacks* Δ^i *and the contexts* $\delta^i\Delta$ $(i \in I\!N\,,\ 0 \leq i \leq n)$ *as follows:* Δ^0 *is the stack* Δ *itself and if* $\Delta^i = \Psi; \Gamma$*, then* $\Delta^{i+1} = \Psi$ *and* $\delta^i = \Gamma$*, so that*

$$\Delta = (\Delta^1; \delta^0\Delta) = (\Delta^2; \delta^1\Delta; \delta^0\Delta) = \cdots = (.; \delta^n\Delta; \ldots; \delta^1\Delta; \delta^0\Delta).$$

Lemma 8 (Inversion lemmas or generation lemmas) *We have the following:*

1. $\Delta \vdash x : \Box A \Rightarrow x : \Box A \in \Delta$.
2. $\Delta; \Gamma \vdash x : A$ *and* $A \not\equiv \Box A' \Rightarrow x : A \in \Gamma$.
3. $(\Delta \vdash \lambda x : A.M \ : \ A \to B) \Rightarrow (\Delta, x : A \vdash M : B)$*, where the variable* x *has possibly been renamed so that* $\Delta, x : A$ *is valid.*
4. $(\Delta \vdash \uparrow M : \Box A) \Rightarrow (\Delta; . \vdash M : A)$.
5. $(\Delta \vdash \downarrow N : A) \Rightarrow (\Delta \vdash N : \Box A)$.
6. $(\Delta \vdash (M\ \ N) : B) \Rightarrow (\exists n \in N.\Delta^n \vdash M : A \to B$ *and* $\Delta^n \vdash N : A)$.

Proof The proof is done by induction on the derivation of the hypothesis. We use the basic lemmas of Section 2.3 to simplify the result in the fourth and fifth cases. $\Box$

Note The nondeterministic nature of the typing system slightly affects the application rule (only this rule is affected): If B is not of the form $\Box C$, the rule yielding $\Delta \vdash (M\ \ N) : B$ is (App), and n is equal to 0 in the sixth inversion rule. Otherwise, rule (App) is eventually reached after a certain number of applications of the (Pop) rule.

3 Untyped reduction

After describing the syntax of the system, we turn to its semantics. We begin our study by introducing the untyped reduction rules.

Definition 9 (Untyped reduction) *We introduce the following one-step reduction relations:*

$$
\begin{array}{ll}
(\beta) & (\lambda x : A.M\ \ N)\ \ \beta\ \ M[N/x] \\
(\eta) & \lambda x : A.(M\ \ x)\ \ \eta\ \ M\ \ \textit{if } x \notin FV(M) \\
(\beta_\Box) & \downarrow\uparrow M\ \ \beta_\Box\ \ M \\
(\eta_\Box) & \uparrow\downarrow M\ \ \eta_\Box\ \ M
\end{array}
$$

Note The $(\beta_\Box)$ and $(\eta_\Box)$ rules correspond to the elimination of the following patterns in the derivation tree:

$$
\dfrac{\dfrac{\Delta; . \vdash M : A}{\Delta \vdash \uparrow M : \Box A}}{\Delta \vdash \downarrow\uparrow M : A}
\qquad\qquad
\dfrac{\dfrac{\Delta; . \vdash M : \Box A}{\Delta; . \vdash \downarrow M : A}}{\Delta \vdash \uparrow\downarrow M : \Box A}
$$

Notice that if $\Delta; . \vdash M : A$, then $\Delta \vdash M : A$ (by Lemma 5).

The rules $(\beta_\Box)$ and $(\eta_\Box)$ are named after the rules (β) and (η). Indeed, just as the abstraction and the application are the constructor and the destructor of the arrow, $\uparrow$ and $\downarrow$ are the constructor and the destructor of $\Box$.

Definition 10 (Compatible closure) *Let R be a relation on terms. Then the* compatible closure *of R, notation $M \hookrightarrow_R N$, is the least relation satisfying the following rules:*

$$
(R_i)\ \dfrac{M\ R\ N}{M \hookrightarrow_R N}
\qquad\qquad
(\xi)\ \dfrac{M \hookrightarrow_R N}{\lambda x : A.M \hookrightarrow_R \lambda x : A.N}
$$

$$
(App_l)\ \dfrac{M \hookrightarrow_R P}{(M\ \ N) \hookrightarrow_R (P\ \ N)}
\qquad\qquad
(App_r)\ \dfrac{N \hookrightarrow_R P}{(M\ \ N) \hookrightarrow_R (M\ \ P)}
$$

$$
(\downarrow)\ \dfrac{M \hookrightarrow_R N}{\downarrow M \hookrightarrow_R \downarrow N}
\qquad\qquad
(\uparrow)\ \dfrac{M \hookrightarrow_R N}{\uparrow M \hookrightarrow_R \uparrow N}
$$

Let the untyped relation $M \hookrightarrow N$ be the compatible closure of the reduction relations defined above. The reflexive and transitive closure (resp., reflexive, symmetrical, and transitive closure) are denoted by $\hookrightarrow_*$ (resp., by $\cong$).

Definition 11 (Normal form) *A term is normal if and only if it has no reduction for $\hookrightarrow$.*

Lemma 12 (Forms of normal terms) *The normal forms can be characterized by induction:*

- *variables are normal,*
- $\lambda x : A.M$ *is normal if* M *is normal and not of the form* $(N\ x)$ *with* $x \notin FV(N)$*,*
- $(M\ N)$ *is normal if* M *and* N *are normal and* M *is not of the form* $\lambda x : A.P$*,*
- $\uparrow M$ *is normal if* M *is normal and not of the form* $\downarrow N$*,*
- $\downarrow M$ *is normal if* M *is normal and not of the form* $\uparrow N$*.*

Note As usual, if M is normal, then all of its subterms are normal, too. (This is obvious, since reduction is compatible with all of the operations.)

Definition 13 (Strongly normalizing) *A term is* strongly normalizing *if all of the reduction sequences starting from that term terminate.*

Definition 14 (Diamond property) *We say that a term satisfies the* diamond property *if whenever* $M \hookrightarrow_* N$ *and* $M \hookrightarrow_* P$ *then there exists a term* Q *such that* $N \hookrightarrow_* Q$ *and* $P \hookrightarrow_* Q$*:*

$$\begin{tikzcd}
 & M \arrow[dl, "*"'] \arrow[dr, "*"] & \\
N \arrow[dr, "*"'] & & P \arrow[dl, "*"] \\
 & Q &
\end{tikzcd}$$

Lemma 15 (Substitution and reduction) *We have the following:*

1. $M \hookrightarrow_* M' \Rightarrow M[N/x] \hookrightarrow_* M'[N/x]$.
2. $N \hookrightarrow_* N' \Rightarrow M[N/x] \hookrightarrow_* M[N'/x]$.
3. $(M \hookrightarrow_* M'$ *and* $N \hookrightarrow_* N') \Rightarrow M[N/x] \hookrightarrow_* M'[N'/x]$.

Proof The first case is proved by induction, first on the hypothesis $M \hookrightarrow_* M'$, and then on M. The second case is proved by induction on M. The third one follows from the previous two results. $\square$

Now we define a typed judgment $\Delta \vdash M = N : A$, the intended meaning of which is that M and N are equal objects of type A in stack Δ. The inference rules that define this judgment stem from the reduction rules introduced above.

Definition 16 (Equality) *The judgment $\Delta \vdash M = N : A$ is defined as follows:*

$$(\mathit{Refl})\ \frac{\Delta \vdash M : A}{\Delta \vdash M = M : A}$$

$$(\mathit{Sym})\ \frac{\Delta \vdash M = N : A}{\Delta \vdash N = M : A}$$

$$(\mathit{Trans})\ \frac{\Delta \vdash M = N : A \qquad \Delta \vdash N = P : A}{\Delta \vdash M = P : A}$$

$$(\beta)\ \frac{\Delta, x : A \vdash M : B \qquad \Delta \vdash N : A}{\Delta \vdash (\lambda x : A.M \quad N) \ = \ M[N/x] \ : \ B}$$

$$(\eta)\ \frac{\Delta \vdash M : A \to B}{\Delta \vdash \lambda x : A.(M \ x) = M \ : \ A \to B}$$

$$(\mathit{Eq}\lambda)\ \frac{\Delta, x : A \vdash M = N : B}{\Delta \vdash \lambda x : A.M \ = \ \lambda x : A.N \ : \ A \to B}$$

$$(\mathit{EqApp})\ \frac{\Delta \vdash M = P : A \to B \qquad \Delta \vdash N = Q : A}{\Delta \vdash (M \ N) \ = \ (P \ Q) : B}$$

$$(\mathit{Eq}\beta_\Box)\ \frac{\Delta; . \vdash M : A}{\Delta \vdash {\downarrow}{\uparrow} M = M : A}$$

$$(\mathit{Eq}\eta_\Box)\ \frac{\Delta; . \vdash M : \Box A}{\Delta \vdash {\uparrow}{\downarrow} M = M : \Box A}$$

$$(\mathit{Eq}{\downarrow})\ \frac{\Delta \vdash M = N : \Box A}{\Delta \vdash {\downarrow} M = {\downarrow} N : A}$$

$$(\mathit{Eq}{\uparrow})\ \frac{\Delta; . \vdash M = N : A}{\Delta \vdash {\uparrow} M = {\uparrow} N : \Box A}$$

$$(\mathit{EqPop})\ \frac{\Delta \vdash M = N : \Box A}{\Delta; \Gamma \vdash M = N : \Box A}\ \Delta; \quad \Gamma \ \mathit{valid}$$

As expected, this equality expresses conversion between well-typed terms.

Lemma 17 *If $\Delta \vdash M = N : A$, then ($\Delta \vdash M : A$, $\Delta \vdash N : A$ and $M \cong N$).*

Proof The proof is by induction on the derivation of $\Delta \vdash M = N : A$. $\square$

4 Typed operational semantics

Following Goguen [Gog94], we define a typed operational semantics based on standard reduction (or left-most reduction). Like him, we not only give a reduction path from any term M to a normal form, but also require that all the subterms of M have a normal form as well. Our operational system has the same form as Goguen's with additional rules for modal operators.

4.1 Weak-head normal forms

Before introducing the system, we need some preliminary definitions in order to state properly the side conditions of some inference rules. These definitions are quite similar to the ones found in [Gog94]. We use the similarities between application and $\downarrow$ and between abstraction and $\uparrow$ to extend the definitions to our modal setting.

Definition 18 (Base term) *A term is a* base term *if it is a variable or if it is an application $(M\ N)$ and M is a base term or if it is a term of the form $\downarrow M$ and M is a base term.*

Definition 19 (Weak-head normal) *We say that a term is* weak-head normal (whn) *or in weak-head normal form if it is any one of the following:*

- *a variable,*
- *an abstraction,*
- *a term of the form $\uparrow M$,*
- *an application $(M\ N)$ where M is whn and not an abstraction,*
- *a term $\downarrow M$ where M is whn and not of the form $\uparrow M'$.*

A base term is always whn. The notion of whn terms is used later, in the side conditions of the typed operational semantics (see Section 4.2). The following results are useful when proving metatheoretical results about the typed operational semantics.

Lemma 20 *We have the following:*

- *If M is normal, then M is whn.*
- *If $(M\ \ N)$ is whn and M and N are normal, then $(M\ \ N)$ is normal.*
- *If $\downarrow M$ is whn and M is normal, then $\downarrow M$ is normal.*
- *If M is whn and $M \hookrightarrow_{\beta\beta_{\Box}*} N$, then N is whn.*

Proof The proof is straightforward. $\Box$

Note The proposition "If M is whn and $M \hookrightarrow_* N$, then N is whn" is false here, because of the presence of η-contraction. For instance, $\lambda x : A.(\lambda y : B.y\ \ (M\ x))$ is whn and reduces to M even if M is not whn. Nevertheless, if we refine the hypotheses, we get similar results, as seen in the next lemma.

Lemma 21 *We have the following:*

- *If $(M\ \ N)$ is well typed, $(M\ \ N) \hookrightarrow_* R$, and $(M\ \ N)$ is whn, then R is whn.*
- *If $\downarrow M$ is well typed, $\downarrow M \hookrightarrow_* R$, and $\downarrow M$ is whn, then R is whn.*

Proof The proof is done by induction on M. $\Box$

4.2 The inference rules

Now we come to the actual inference rules of the typed operational semantics. They are strongly inspired by the original system [Gog94]. The main changes are the new rules for the modal operators, which are inspired by the analogies between $\downarrow$ and the application and between $\uparrow$ and λ.

Note that $\Delta \vdash M \rightarrow_{nf} N : A$ means that M has canonical form N that is a canonical term of type A in stack Δ. $\Delta \vdash M \rightarrow_{wh} N : A$ means that M weak-head reduces to N of type A in stack Δ.

Normal forms

$$(S\mathrm{Var})\ \frac{x : A \in \Gamma}{\Delta; \Gamma \vdash x \to_{nf} x : A}\ \Delta;\quad \Gamma \text{ valid}$$

$$(S\lambda)\ \frac{\Delta, x : A \vdash M \to_{nf} P : B}{\Delta \vdash \lambda x : A.M \to_{nf} \lambda x : A.P : A \to B}\quad \text{if } P \equiv (Q\ x)\ \Rightarrow\ x \in FV(Q)$$

$$(S\eta)\ \frac{\Delta, x : A \vdash M \to_{nf} (P\ x) : B \qquad \Delta \vdash P \to_{nf} Q : A \to B}{\Delta \vdash \lambda x : A.M \to_{nf} Q\ :\ A \to B}$$

$$(S\mathrm{App})\ \frac{\Delta \vdash M \to_{nf} P : A \to B \qquad \Delta \vdash N \to_{nf} Q : A}{\Delta \vdash (M\ N) \to_{nf} (P\ Q) : B}\quad \text{if } (M\ N) \text{ is whn}$$

$$(S{\downarrow})\ \frac{\Delta \vdash M \to_{nf} N : \Box A}{\Delta \vdash {\downarrow} M \to_{nf} {\downarrow} N : A}\quad \text{if } {\downarrow} M \text{ is whn}$$

$$(S{\uparrow})\ \frac{\Delta; . \vdash M \to_{nf} N : A}{\Delta \vdash {\uparrow} M \to_{nf} {\uparrow} N : \Box A}\quad \text{if } N \neq {\downarrow} P$$

$$(S\eta_{\Box})\ \frac{\Delta; . \vdash M \to_{nf} {\downarrow} N : A}{\Delta \vdash {\uparrow} M \to_{nf} N : \Box A}$$

$$(S\mathrm{Pop})\ \frac{\Delta \vdash M \to_{nf} N : \Box A}{\Delta; \Gamma \vdash M \to_{nf} N : \Box A}\ \Delta;\quad \Gamma \text{ valid}$$

$$(SW)\ \frac{\Delta \vdash M \to_{wh} N : A \qquad \Delta \vdash N \to_{nf} P : A}{\Delta \vdash M \to_{nf} P : A}$$

weak-head reduction

$$(W\beta)\ \frac{\Delta \vdash \lambda x : A.M \to_{nf} M' : A \to B \qquad \Delta \vdash N \to_{nf} N' : A}{\Delta \vdash (\lambda x : A.M\ \ N) \to_{wh} M[N/x]\ :\ B}$$

$$(W\mathrm{App})\ \frac{\Delta \vdash M \to_{wh} P : A \to B \qquad \Delta \vdash N \to_{nf} N' : A}{\Delta \vdash (M\ N) \to_{wh} (P\ N) : B}$$

$$(W\beta_{\Box})\ \frac{\Delta \vdash {\uparrow} M \to_{nf} M' : \Box A}{\Delta \vdash {\downarrow}{\uparrow} M \to_{wh} M : A}$$

$$(W{\downarrow})\ \frac{\Delta \vdash M \to_{wh} N : \Box A}{\Delta \vdash {\downarrow} M \to_{wh} {\downarrow} N : A}$$

$$(W\mathrm{Pop})\ \frac{\Delta \vdash M \to_{wh} N : \Box A}{\Delta; \Gamma \vdash M \to_{wh} N : \Box A}\ \Delta;\quad \Gamma \text{ valid}$$

Note In rule $(S\eta)$, the purpose of the second premise is to ensure that the variable x is not free in P. In Proposition 49, we show the admissibility of the more natural η-rule:

$$(S\eta')\ \frac{\Delta, x : A \vdash M \rightarrow_{nf} (P\ x) : B \qquad x \notin FV(P)}{\Delta \vdash \lambda x : A.M \rightarrow_{nf} P\ :\ A \rightarrow B}$$

5 Metatheoretical results for the typed operational semantics

Now we study metatheoretical results for the typed operational semantics (TOS). We first introduce the notion of typed substitutions between two stacks. The definition stems from Goguen's one, but it is modified to take modality into account. Then we present basic results about the TOS (stack manipulations, inversion lemmas). Finally, we prove that the reductions in the TOS correspond to actual untyped reductions (adequacy of reduction), and we show subject reduction, strong normalization, and the diamond property for all the terms that have an $\rightarrow_{nf}$ reduction in the TOS. These latter results are interesting in themselves if our aim is to study the TOS, but above all, they are intermediate results in our proof of the subject reduction, strong normalization, and Church-Rosser properties for the untyped reduction.

5.1 Typed substitutions

Definition 22 (Presubstitution) *A* presubstitution *for a finite set of variables S is a function from S to terms.*

Definition 23 (Prerenaming) *A* prerenaming δ *for a finite set of variables S is a presubstitution for S such that for each x in S, $(\delta\ x)$ is a variable.*

Notation Suppose Δ and Φ are context stacks, D_i and Γ are contexts, δ is a presubstitution for dom(Δ), ϕ is a presubstitution for dom(Φ), and ρ is a presubstitution for dom(Γ). Then the following hold:

- We write $(\overline{\delta}\ M)$ for the result of simultaneously substituting the values for the variables in the domain of Δ:

$$(\overline{\delta}\ M) =_{def} M[(\delta\ x_1), \ldots, (\delta\ x_n)/x_1, \ldots, x_n].$$

- We write $\delta[x := M]$ for the extended presubstitution for $\mathrm{dom}(\Delta, x : A)$ such that
  - $(\delta[x := M]\ y) = (\delta\ y)$ if $y \in \mathrm{dom}(\Delta)$,
  - $(\delta[x := M]\ x) = M$.
- We write $\delta; \rho$ for the extended presubstitution for $\mathrm{dom}(\Delta; \Gamma)$ such that
  - $(\delta; \rho\ x) = (\delta\ x)$ if $x \in \Delta$,
  - $(\delta; \rho\ x) = (\rho\ x)$ if $x \in \Gamma$.
- The composition of δ and ϕ, $\delta \circ \phi$, is $(\delta \circ \phi\ \ x) = (\overline{\delta}(\phi\ x))$.
- If Δ' is a stack such that $\Delta' \subseteq \Delta$ (i.e., all of the declarations of Δ' appear in Δ), then $\delta|\Delta'$ is the presubstitution for $\mathrm{dom}(\Delta')$ such that $\forall x \in \mathrm{dom}(\Delta'), (\delta|\Delta'\ x) = (\delta\ x)$.

Defining a proper notion of typed substitution for our modal system is not obvious, because we manipulate context stacks instead of contexts. A first attempt, too restrictive, would be to define a substitution ρ from Φ to Δ as a presubstitution for $\mathrm{dom}(\Phi)$ such that, for each $x : A \in \delta^i\Phi$, we have $\Delta^i \vdash (\rho\ x) : A$. (Besides, it would implicitly mean that Δ is necessarily a stack with more contexts than Φ.) The following definition is more flexible.

Definition 24 (Substitution) *A* substitution *ρ from Φ to Δ, where Δ and Φ are context stacks, is a presubstitution for dom(Φ) such that there exists a nondecreasing function f such that for each $x : A \in \delta^i\Phi$ we have $\Delta^{f(i)} \vdash (\rho\ x) : A$.*

Definition 25 (Renaming) *A* renaming *δ from Φ to Δ is a substitution from Φ to Δ such that $(\delta\ x) = y$, where $y \in$ dom(Δ), for each $x : A \in \Phi$.*

Definition 26 (Compatible context stacks) *A context stack Δ is said to be* compatible *with a context stack Φ if, for each context $\delta^i\Phi$, there exists a context $\delta^j\Delta$ such that $\delta^j\Delta$ has all declarations of $\delta^i\Phi$, and the function $i \mapsto j$ is a nondecreasing function f such that $f(0) = 0$.*

Examples We have the following:

- $.; D_1; \dots; (D_i, D_{i+1}); \dots; D_n$ is compatible with $.; D_1; \dots; D_n$ ($f(n-i-1) = f(n-i) = n-i$):

$$\begin{tikzcd}
. \,; & D_1 \arrow[d] & ; \dots ; & D_i \arrow[dr] & ; & D_{i+1} \arrow[dl] & ; \dots ; & D_n \arrow[d, "f"] \\
. \,; & D_1 & ; \dots ; & & (D_i, D_{i+1}) & & ; \dots ; & D_n
\end{tikzcd}$$

- $.; D_1; \dots; (D_i, x : B); \dots; D_n$ is compatible with $.; D_1; \dots; D_n$ ($f(k) = k, \forall k \in \{1, \dots, n\}$):

$$\begin{tikzcd}
. \,; & D_1 \arrow[d] & ; \dots ; & D_i \arrow[d] & ; \dots ; & D_n \arrow[d, "f"] \\
. \,; & D_1 & ; \dots ; & D_i, x : B & ; \dots ; & D_n
\end{tikzcd}$$

- $.; D_1; \dots; D_i; D; D_{i+1}; \dots; D_n$ is compatible with $.; D_1; \dots; D_n$ ($f(n-i-1) = n-i-1, f(n-i) = n-i+1$):

$$\begin{tikzcd}
 & . \,; & D_1 \arrow[dl] & ; \dots ; & D_i \arrow[dl] & ; D_{i+1} \arrow[d] & ; \dots ; & D_n \arrow[d, "f"] \\
. \,; & D_1 & ; \dots ; & D_i \,; & D & ; D_{i+1} & ; \dots ; & D_n
\end{tikzcd}$$

Notation

- If Δ is compatible with Φ, then we write $weak_{\Phi}^{\Delta}$ for the presubstitution $(weak_{\Phi}^{\Delta}\ x) = x$ for $dom(\Phi)$. Let us present some examples:
  - $\mathrm{weak}_{.;D_1;\dots;D_n}^{.;D_1;\dots;(D_i,D_{i+1});\dots;D_n}$ for $i \in [1..n[$
  - $\mathrm{weak}_{.;D_1;\dots;D_n}^{.;D_1;\dots;(D_i,x:B);\dots;D_n}$ for $i \in [1..n]$
  - $\mathrm{weak}_{.;D_1;\dots;D_n}^{.;D_1;\dots;D_i;D;D_{i+1};\dots;D_n}$ for $i \in [1..n[$
  - $\mathrm{weak}_{\Delta;.}^{\Delta}$
- $id_{\Delta} =_{def} \mathrm{weak}_{\Delta}^{\Delta}$

It appears that some ideas underlying our definition of typed substitution can be found in the work by Jean Goubault-Larrecq [GL96]. In particular, the above definition of compatible contexts stacks can be expressed with two operations that are atomic in Goubault's work: fusion of two contexts in a stack and insertion of a context into a stack. These operations are defined there as term operators.

Lemma 27 *If Δ is a context stack compatible with Φ, then $weak^{\Delta}_{\Phi}$ is a renaming from Φ to Δ.*

Proof The proof is by definition of renamings. $\square$

Lemma 28 *Suppose δ is a substitution from Φ to Δ^n (for $n \geq 0$) and ρ is a substitution from $.;\Gamma$ to Δ. Then $\delta;\rho$ is a substitution from $\Phi;\Gamma$ to Δ.*

Proof The proof is by definition of substitutions and $\delta;\rho$. $\square$

In particular, $\delta; \mathrm{weak}^{\Delta;.}_{.;.}$ is a substitution from $\Phi;.$ to $\Delta;.$ that we use in several places in our proofs. Note that for any $x \in \Phi$ we have $(\delta; \mathrm{weak}^{\Delta;.}_{.;.}\ x) = (\delta\ x)$.

Lemma 29 *For prerenamings and presubstitutions, we have the following:*

- $\delta \circ (\phi[x := N]) \equiv (\delta \circ \phi)[x := (\overline{\delta}\ N)]$.
- $(\overline{\delta \circ \phi}\ M) \equiv (\overline{\delta}(\overline{\phi}\ M))$.
- $\delta \circ (\phi_1; \phi_2) \equiv (\delta \circ \phi_1); (\delta \circ \phi_2)$.
- $(\overline{\rho} \downarrow M) \equiv \downarrow(\overline{\rho}\ M)$ *and* $(\overline{\rho} \uparrow M) \equiv \uparrow(\overline{\rho}\ M)$.
- *Let δ be a prerenaming from Φ to Δ, and let us assume that $x \notin dom(\Phi)$, $y \notin dom(\Delta)$, and $y \notin FV(M)$. Then $(\overline{\delta}\ \lambda x : A.M) \equiv \lambda y : A.(\overline{\delta[x := y]}\ M)$.*

Moreover, for renamings and substitutions, we have that the following hold:

- *If ϕ is a substitution from Φ to Δ and δ is a substitution from Δ to Θ, then $\delta \circ \phi$ is a substitution from Φ to Θ.*
- *If δ is a substitution from Φ to Δ and f is the corresponding nondecreasing function, then $\delta|\Phi^i$ is a substitution from Φ^i to $\Delta^{f(i)}$.*
- *If δ is a substitution from $\Phi;\Gamma;\Phi'$ to Δ, then $\delta|\Phi;\Phi'$ is a substitution from $\Phi;\Phi'$ to Δ.*
- *If δ is a renaming from $\Phi;\Gamma$ to Δ, then δ is of the form $\delta_1;\delta_2$, where δ_1 is a renaming from Φ to Δ^n ($n \in I\!N$) and δ_2 is a renaming from $.;\Gamma$ to Δ.*

Proof The proof is straightforward. $\square$

5.2 Basic lemmas

Now we state basic results about the typed operational semantics.

Lemma 30 (Free variables) *The following rule is valid for reductions $\rightarrow_{nf}$ and $\rightarrow_{wh}$: If $\Delta \vdash M \rightarrow N : A$, then $FV(M) \subseteq dom(\Delta)$ and $FV(N) \subseteq dom(\Delta)$.*

Lemma 31 (Contexts) *If $\Delta \vdash M \rightarrow_{nf} N : A$ or $\Delta \vdash M \rightarrow_{wh} N : B$, then Δ is a valid context stack.*

Proof The proof is by induction on derivations. □

Lemma 32 (Substitution preserves typing) *If $\Delta \vdash M : A$ and δ is a renaming from Δ to Φ, then $\Phi \vdash (\overline{\delta}\ M) : A$.*

Proof The proof is by induction on derivations of $\Delta \vdash M : A$. □

Lemma 33 (Renaming) *The following rule is valid for reductions $\rightarrow_{nf}$ and $\rightarrow_{wh}$: If δ is a renaming from Φ to Δ and if $\Phi \vdash M \rightarrow N : A$, then $\Delta \vdash (\overline{\delta}\ M) \rightarrow (\overline{\delta}\ N) : A$.*

Proof The proof is by induction on derivations, using the basic properties and definitions of Section 5.1. In particular, the definition of the renaming $\mathrm{weak}_{.;.}^{\Delta;.}$ is a useful tool here, as in many proofs in the rest of the paper. □

Thanks to our flexible definition of renaming, the renaming lemma can be seen as a generalization of several lemmas that deal with the preservation of $\rightarrow_{nf}$ and $\rightarrow_{wh}$ judgments by usual manipulations of stacks.

Corollary 34 (Thinning) *The following rule is valid for reductions $\rightarrow_{nf}$ and $\rightarrow_{wh}$:*

$$(\mathit{Thinning})\ \frac{.; D_1; \ldots ; D_i \ldots ; D_n \vdash M \rightarrow N : A}{.; D_1; \ldots ; (D_i, x : B); \ldots ; D_n \vdash M \rightarrow N : A}$$

where $x \notin dom(.; D_1; \ldots ; D_n)$ $(1 \leq i \leq n)$

Proof Apply Lemma 33 with $\delta = \mathrm{weak}_{.;D_1;\ldots;D_i\ldots;D_n}^{.;D_1;\ldots;(D_i,x:B);\ldots;D_n}$. □

Corollary 35 (Fusion) *The following rule is valid for reductions $\rightarrow_{nf}$ and $\rightarrow_{wh}$:*

$$(Fusion)\ \frac{.; D_1; \ldots ; D_i; D_{i+1}; \ldots ; D_n \vdash M \rightarrow N : A}{.; D_1; \ldots ; (D_i, D_{i+1}); \ldots ; D_n \vdash M \rightarrow N : A} \quad for\ 1 \leq i < n.$$

Proof Apply Lemma 33 with $\delta = \mathrm{weak}^{.;D_1;\ldots;(D_i,D_{i+1});\ldots;D_n}_{.;D_1;\ldots;D_i;D_{i+1};\ldots;D_n}$. □

Corollary 36 (Weakening) *The following rule is valid for reductions $\rightarrow_{nf}$ and $\rightarrow_{wh}$:*

$$(Weakening)\ \frac{.; D_1; \ldots ; D_i; D_{i+1}; \ldots ; D_n \vdash M \rightarrow N : A}{.; D_1; \ldots ; D_i; D; D_{i+1}; \ldots ; D_n \vdash M \rightarrow N : A} \quad for\ 1 \leq i < n.$$

Proof Apply Lemma 33 with $\delta = \mathrm{weak}^{.;D_1;\ldots;D_i;D;D_{i+1};\ldots;D_n}_{.;D_1;\ldots;D_i;D_{i+1};\ldots;D_n}$. □

Lemma 37 (Strengthening) *The following rule is valid for reductions $\rightarrow_{nf}$ and $\rightarrow_{wh}$:*

$$(Strengthening)\ \frac{.; D_1; \ldots ; \Gamma, x : C, \Gamma'; \ldots ; D_n \vdash M \rightarrow N : A}{.; D_1; \ldots ; \Gamma, \Gamma'; \ldots ; D_n \vdash M \rightarrow N : A} \quad x \notin FV(M).$$

Proof The proof is done by induction on derivations. □

The following results analyse how a judgment $\Delta \vdash M \rightarrow_{nf} P : A$ can be obtained, according to the form of term M. As before, the inversion lemmas are affected by the nondeterminism introduced by rule (Pop).

Lemma 38 (Inversion lemmas for $\rightarrow_{nf}$) *We have the following:*

1. $\Delta \vdash x \rightarrow_{nf} x : \Box A \Rightarrow x : \Box A \in \Delta$.

2. $\Delta; \Gamma \vdash x \rightarrow_{nf} x : A$ *and* $A \not\equiv \Box A' \Rightarrow x : A \in \Gamma$.

3. $\Delta \vdash \lambda x : A.M \rightarrow_{nf} Q : A \rightarrow B \Rightarrow (\Delta, x : A \vdash M \rightarrow_{nf} P : B$ *with either* $P \equiv (R\ x)$ *with* $x \notin FV(R)$, $\Delta \vdash R \rightarrow_{nf} S : A \rightarrow B$ *and* $Q \equiv S$ *or* $Q \equiv \lambda x : A.P)$.

4. $\Delta \vdash (M\ N) \rightarrow_{nf} R : B$ *and* $(M\ N)$ *is whn* $\Rightarrow \exists P, Q.R \equiv (P\ Q)$ *and* $\exists n \in I\!N .\Delta^n \vdash M \rightarrow_{nf} P : A \rightarrow B$ *and* $\Delta^n \vdash N \rightarrow_{nf} Q : A$ *where* $n = 0$ *if* $B \not\equiv \Box B'$.

5. $\Delta \vdash (M\ N) \rightarrow_{nf} P : A$ *and* $(M\ N)$ *is not whn* $\Rightarrow \Delta \vdash (M\ N) \rightarrow_{wh} M' : A$ *and* $\Delta \vdash M' \rightarrow_{nf} P : A$.

6. $\Delta \vdash \downarrow M \rightarrow_{nf} R : A$ *and* $\downarrow M$ *is whn* $\Rightarrow \exists N. R \equiv \downarrow N$ *and* $\Delta \vdash M \rightarrow_{nf} N : \Box A$

7. $\Delta \vdash \downarrow M \rightarrow_{nf} R : A$ *and* $\downarrow M$ *is not whn* $\Rightarrow \Delta \vdash \downarrow M \rightarrow_{wh} M' : A$ *and* $\Delta \vdash M' \rightarrow_{nf} R : A$.

8. $\Delta \vdash \uparrow M \rightarrow_{nf} Q : \Box A \Rightarrow \Delta; . \vdash M \rightarrow_{nf} P : A$ *with either* $Q \equiv \uparrow P$ *and* $P \not\equiv \downarrow P'$, *or* $P \equiv \downarrow Q$.

Proof The proof is by induction on derivations. □

We also have inversion lemmas for judgment $\rightarrow_{wh}$.

Lemma 39 (Inversion lemmas for $\rightarrow_{wh}$) *We have the following:*

1. $\Delta \vdash (M\ N) \rightarrow_{wh} Q : B \Rightarrow \exists n \in I\!N . \Delta^n \vdash N \rightarrow_{nf} N' : A$ *with either* $M \equiv \lambda x : A.P$, $Q \equiv P[N/x]$, $\Delta^n \vdash \lambda x : A.P \rightarrow_{nf} M' : A \rightarrow B$, *or* $\Delta^n \vdash M \rightarrow_{wh} P : A \rightarrow B$ *and* $Q \equiv (P\ N)$.

2. $\Delta \vdash \downarrow M \rightarrow_{wh} N : A \Rightarrow$ *either* $M \equiv \uparrow P$, $N \equiv P$, $\Delta; . \vdash P \rightarrow_{nf} P' : A$, *or* $N \equiv \downarrow P$, $\Delta \vdash M \rightarrow_{wh} P : \Box A$.

Proof The proof is by induction on derivations. □

Lemma 40 (Uniqueness of normal forms) *The following result is valid for reductions* $\rightarrow_{nf}$ *and* $\rightarrow_{wh}$: *If* $\Delta \vdash M \rightarrow P : A$ *and* $\Delta \vdash M \rightarrow Q : B$, *then* $P \equiv Q$ *and* $A \equiv B$.

Proof The proof is done by induction on the proof of the first hypothesis, using the inversion lemmas on the second hypothesis. □

Lemma 41 (Completeness) *We have the following:*

- *If* $\Delta \vdash M \rightarrow_{nf} N : A$, *then* $\Delta \vdash M = N : A$.
- *If* $\Delta \vdash M \rightarrow_{wh} N : A$, *then* $\Delta \vdash M = N : A$.

Proof The proof is by induction on derivations. □

5.3 Typed operational system and untyped reduction

In this section we establish the links between the typed operational semantics and the untyped reduction. First we prove that the judgment $\Delta \vdash M \rightarrow_{nf} P : A$ actually computes a reduct of M that is normal. Similarly, the $\rightarrow_{wh}$ judgment corresponds to β and $\beta_\Box$ reductions.

Lemma 42 (Adequacy for reduction) *We have the following:*

- *If $\Delta \vdash M \rightarrow_{nf} P : A$, then $M \hookrightarrow_* P$ and P is normal, and, furthermore, there is an N such that $M \hookrightarrow_{\beta\beta_\Box *} N \hookrightarrow_{\eta\eta_\Box *} P$.*
- *If $\Delta \vdash M \rightarrow_{wh} P : A$, then $M \hookrightarrow_{\beta\beta_\Box} P$.*

Proof The proof is by induction on derivations. $\Box$

Lemma 43 *If $\Delta \vdash M \rightarrow_{nf} N : A$ and M is normal, then $M \equiv N$.*

Proof The proof is done by adequacy for reduction $M \hookrightarrow_* N$. Furthermore, M has no reduction. Thus $M \equiv N$. $\Box$

In the rest of the section, we prove important properties about the judgment $\rightarrow_{nf}$, namely, if $\Delta \vdash M \rightarrow_{nf} P : A$, then M is strongly normalizing, and if $\Delta \vdash M \rightarrow_{nf} P : A$ and $M \hookrightarrow N$, then $\Delta \vdash N \rightarrow_{nf} P : A$.

Unlike Goguen, we do not use an intermediate predicate (the logical meaning of which was unclear to us) but instead directly prove the properties. Also we do not need to define the stack in which a subterm of a well-typed term is itself typable. (This would not have been as easy as in the simply typed λ-calculus.)

We need some preliminary lemmas because of (η) and $(\eta\Box)$ reductions. They are proved using the inversion lemmas. The first one was already proved in Goguen's thesis [Gog94].

Lemma 44 (Subject reduction for η) *If $\Delta \vdash \lambda x : A.(M\ x) \rightarrow_{nf} P : A \rightarrow B$ and $x \notin FV(M)$, then $\Delta \vdash M \rightarrow_{nf} P : A \rightarrow B$.*

Lemma 45 (Subject reduction for $\eta_\Box$) *If $\Delta \vdash \uparrow\downarrow M \rightarrow_{nf} P : A$, then $\Delta \vdash M \rightarrow_{nf} P : A$.*

Note that in the previous lemma, A is necessarily of the form $\Box B$. Now we prove the important results of this section, using the technical developments of the appendix. It could seem worrying that the proofs of these results are not stand-alone. In fact, extra lemmas are also needed to complete the proofs of Goguen's thesis. Unfortunately, they are missing in [Gog94]. This defect has since been fixed by replacing the two judgments of the TOS by one: $\Gamma \vdash M \to N \to P : A$ means that M,N and P are well formed of type A under assumption Γ, and M has whn form N and normal form P (cf. [CG97, Gog99]).

Lemma 46 (Strong normalization for $\to_{nf}$) *We have the following:*

- *If $\Delta \vdash M \to_{nf} P : A$, then M is strongly normalizing.*
- *If $\Delta \vdash M \to_{wh} N : A$ and N is strongly normalizing, then M is strongly normalizing.*

Proof By induction on the length of the derivations of the $\to_{nf}$ and $\to_{wh}$ judgments, we examine the possible sequences of reductions starting from M. In the case (SApp), we use Lemma 21. In the case (WApp), we have to use the results of the appendix:

$$(W\text{App})\ \frac{\Delta \vdash M \to_{wh} P : A \to B \qquad \Delta \vdash N \to_{nf} N' : A}{\Delta \vdash (M\ N) \to_{wh} (P\ N) : B}$$

We assume that $(P\ N)$ is strongly normalizing. Thus P and N are strongly normalizing, and by induction hypothesis M enjoys this property, too. We examine the possible sequences of reductions starting from $(M\ N)$:

- If $(M\ N) \hookrightarrow_* (M'\ N')$, we cannot have an infinite sequence, because M and N are strongly normalizing.
- Otherwise, the left argument of the application is weak-head reduced at some point: $(M\ N) \hookrightarrow_* (M'\ N') \hookrightarrow_{wh} (M''\ N') \hookrightarrow_* R$. In this case we use the results of the appendix: Lemma 73 tells us that $(P\ N) \hookrightarrow_* (M''\ N')$. Thus $(M''\ N')$ is strongly normalizing and we cannot have any infinite sequence.

□

Lemma 47 (Subject Reduction for $\to_{nf}$) *The following hold:*

- *If* $\Delta \vdash M \rightarrow_{nf} P : A$ *and* $M \hookrightarrow_{*} N$*, then* $\Delta \vdash N \rightarrow_{nf} P : A$.
- *If* $\Delta \vdash M \rightarrow_{wh} N : A$, $\Delta \vdash N \rightarrow_{nf} P : A$*, and* $(N \hookrightarrow_{*} N' \Rightarrow \Delta \vdash N' \rightarrow_{nf} P : A)$*, then* $(M \hookrightarrow_{*} M' \Rightarrow \Delta \vdash M' \rightarrow_{nf} P : A)$.

Proof By induction on the length of the derivations of the $\rightarrow_{nf}$ and $\rightarrow_{wh}$ judgments, we examine the possible sequences of reductions starting from M. In the cases $(S\lambda)$ and $(S\eta)$, we use Lemma 44 (subject reduction for η). In the cases $(S\uparrow)$ and $(S\eta_{\Box})$, we use Lemma 45 (subject reduction for $\eta_{\Box}$). As in the proof of the previous lemma, we have to use the results of the appendix in the case $(W\mathrm{App})$:

$$(W\mathrm{App})\ \frac{\Delta \vdash M \rightarrow_{wh} P : A \rightarrow B \qquad \Delta \vdash N \rightarrow_{nf} N'' : A}{\Delta \vdash (M\ N) \rightarrow_{wh} (P\ N) : B}$$

We assume that $\Delta \vdash (P\ N) \rightarrow_{nf} R : B$ and that $(P\ N) \hookrightarrow_{*} P' \Rightarrow \Delta \vdash P' \rightarrow_{nf} R : B$. We examine the possible sequences of reductions starting from $(M\ N)$:

- If $(M\ N) \hookrightarrow_{*} (M'\ N')$ without any weak-head reduction step, then $M \hookrightarrow_{*} M'$ without any weak-head reduction step. Once again, we use the results of the appendix. By Lemma 73, there is a term Q such that $M' \hookrightarrow_{wh} Q$ and $P \hookrightarrow_{*} Q$. By Lemma 69, all the elements of M have a derivation $\rightarrow_{nf}$. By Lemma 72 and induction hypothesis, it is also true for M'. Thus, by Lemma 74, $\Delta \vdash M' \rightarrow_{wh} Q : A \rightarrow B$. Now, since $(P\ N) \hookrightarrow_{*} (Q\ N')$, we have $\Delta \vdash (Q\ N') \rightarrow_{nf} R : B$. By $(W\mathrm{App})$ and induction hypothesis for N, we have $\Delta \vdash (M'\ N') \rightarrow_{wh} (Q\ N') : B$. Finally, by (SW), $\Delta \vdash (M'\ N') \rightarrow_{nf} R : B$.
- Otherwise, the left argument of the application is weak-head reduced at some point:

  $$(M\ N) \hookrightarrow_{*} (M'\ N') \hookrightarrow_{wh} (M''\ N') \hookrightarrow_{*} T.$$

  By Lemma 73, $P \hookrightarrow_{*} M''$. Thus we have $(P\ N) \hookrightarrow_{*} (M''\ N') \hookrightarrow_{*} T$, which implies by hypothesis $\Delta \vdash T \rightarrow_{nf} R : B$.

$\Box$

Corollary 48 (Diamond Property for $\rightarrow_{nf}$) *If* $\Delta \vdash M \rightarrow_{nf} Q : A$, $M \hookrightarrow_{*} N$*, and* $M \hookrightarrow_{*} P$*, then* $N \hookrightarrow_{*} Q$ *and* $P \hookrightarrow_{*} Q$.

The following property is useful when showing the soundness lemma (Lemma 57).

Proposition 49 (Admissibility of $S\eta'$) *The following rule is admissible:*

$$(S\eta')\ \frac{\Delta, x : A \vdash M \rightarrow_{nf} (P\ x) : B \qquad x \notin FV(P)}{\Delta \vdash \lambda x : A.M \rightarrow_{nf} P\ :\ A \rightarrow B}$$

Proof We assume that $\Delta, x : A \vdash M \rightarrow_{nf} (P\ x) : B$. By the adequacy for reduction lemma (Lemma 42), $(P\ x)$ is normal. Thus P is normal as well. We can show by an easy induction that for any normal term Q, well-typed of type A in the stack Φ, we have $\Phi \vdash Q \rightarrow_{nf} Q : A$. Thus $\Delta \vdash P \rightarrow_{nf} P : B$. By applying $(S\ \eta)$, we get $\Delta \vdash \lambda x : A.M \rightarrow_{nf} P\ :\ A \rightarrow B$. ☐

6 Soundness

In the previous section, we proved that if a term has a reduction in the TOS, then it verifies the subject reduction, strong normalization, and Church-Rosser properties for this system. Now it is time to transfer these properties to the original type system by proving the soundness result, namely, that for any well-typed term M of type A in stack Δ, there is a normal form P such that $\Delta \vdash M \rightarrow_{nf} P : A$. The proof is carried out quite classically, in the manner of Tait. Finally, the results of last section enable us to deduce the classic properties of subject reduction, strong normalization, and Church-Rosser for our system.

Definition 50 (Semantic object) *A* semantic object *for Δ and A is a term M such that $\Delta \vdash M \rightarrow_{nf} P : A$ for some term P.*

Definition 51 (Interpretation of types) *Let Δ be a context stack. The* interpretation *of a type A in Δ, denoted by $[\![A]\!]_\Delta$, is given by induction on the structure of the type A:*

- *$[\![c]\!]_\Delta$ is the set of semantic objects for Δ and c.*
- *$[\![A \rightarrow B]\!]_\Delta$ is the set of semantic objects M for Δ and $A \rightarrow B$ such that, for any context Γ such that Δ, Γ is valid and for any $N \in [\![A]\!]_{\Delta,\Gamma}$, we know that $(M\ N) \in [\![B]\!]_{\Delta,\Gamma}$.*

- $[\![\Box A]\!]_\Delta$ *is the set of semantic objects* M *for* Δ *and* $\Box A$ *such that, for any stack* Δ' *such that* Δ, Δ' *is valid, we know that* $(\downarrow M) \in [\![A]\!]_{\Delta,\Delta'}$.

In his thesis, Goguen uses renamings to define $[\![A \to B]\!]_\Delta$. Namely, $[\![A \to B]\!]_\Delta$ is the set of semantic objects M for Δ and $A \to B$ such that, for any renaming δ from Δ' to Δ and any $N \in [\![A]\!]_{\Delta'}$, $((\overline{\delta}\ M)\ N) \in [\![B]\!]_{\Delta'}$. Here we are implicitly using weakenings instead of renamings. Actually, what we want is to be able to extend the contexts with fresh variables, and this is precisely what weakening does.

Lemma 52 (Weakening for $[\![A]\!]_\Delta$) *We can make the following statements:*

1. *If* $M \in [\![A]\!]_\Delta$ *and if* Γ *is a context such that* Δ, Γ *is valid, then* $M \in [\![A]\!]_{\Delta,\Gamma}$.
2. *If* $M \in [\![\Box A]\!]_\Delta$ *and if* Δ' *is a stack such that* $\Delta; \Delta'$ *is valid, then* $M \in [\![\Box A]\!]_{\Delta;\Delta'}$. *In particular, if* Γ *is a context such that* $\Delta; \Gamma$ *is valid, then* $M \in [\![\Box A]\!]_{\Delta;\Gamma}$.

Proof The proof is done by case analysis on type A for the first part and by using the definitions of the interpretations of types for the second part. □

Roughly speaking, the interpretation of a context stack Φ in Δ is the set of the substitutions that replace variables $x : A$ declared in Φ by terms belonging to $[\![A]\!]_\Delta$. The exact definition is a bit more complex, because we deal with context stacks instead of contexts.

Definition 53 (Interpretation of context stacks) *The* interpretation *of a context stack* Φ *in* Δ, $[\![\Phi]\!]_\Delta$, *is the set of substitutions from* Φ *to* Δ *defined by induction on the structure of* Φ*:*

- $[\![.]\!]_\Delta =_{def} \{weak^{\Delta}_{.}\}$.
- $[\![.;.]\!]_\Delta =_{def} \{weak^{\Delta}_{.;.}\}$.
- $[\![.;\Gamma, x : A]\!]_\Delta =_{def} \{\rho[x := M] \mid \rho \in [\![.;\Gamma]\!]_\Delta$ *and* $M \in [\![A]\!]_\Delta\}$.
- $[\![\Phi; \Gamma]\!]_\Delta =_{def} \{\delta; \rho \mid \exists n \in I\!N\ .\delta \in [\![\Phi]\!]_{\Delta^n}$ *and* $\rho \in [\![.;\Gamma]\!]_\Delta\}$.

As a consequence of the last item, $[\![\Phi; .]\!]_{\Delta;.}$ is equal to $\{\delta; \rho \mid \exists n \in \mathbb{N} . \delta \in [\![\Phi]\!]_{(\Delta;.)^n}$ and $\rho \in [\![.;.]\!]_{\Delta;.}\}$, that is, $\{\delta; \mathrm{weak}_{.;.}^{\Delta;.} \mid \delta \in [\![\Phi]\!]_{(\Delta;.)^n}\}$.

The difficulty of the definition lies in the expression of $[\![\Phi; \Gamma]\!]_\Delta$. Simpler choices in the definitions of $[\![\Box A]\!]_\Delta$ and $[\![\Phi; \Gamma]\!]_\Delta$ such as the following would not be flexible enough to meet our needs:

- $[\![\Box A]\!]_\Delta$ is the set of semantic objects M for Δ and $\Box A$ such that $(\downarrow M) \in [\![A]\!]_\Delta$.
- $[\![\Phi; \Gamma]\!]_\Delta =_{def} \{\delta; \rho \mid \delta \in [\![\Phi]\!]_{\Delta^1}$ and $\rho \in [\![.; \Gamma]\!]_\Delta\}$.

In particular, in the proof of the soundness Lemma 57, the case $(\uparrow)$ does not work.

Example Let us assume that $\Phi = (.; x : A; y : B)$ and $\Delta = (.; f : B \to A, z : B; .; u : B)$. Then the substitution δ from Φ to Δ defined by $\delta(x) = (f\ z)$ and $\delta(y) = u$ belongs to $[\![\Phi]\!]_\Delta$ if $u \in [\![B]\!]_\Delta$ and $(f\ z) \in [\![A]\!]_{.;f:B\to A,z:B}$.

Lemma 54 *If $\rho \in [\![\Phi]\!]_\Delta$ and if Γ is a context such that Δ, Γ is valid, then $\rho \in [\![\Phi]\!]_{\Delta,\Gamma}$.*

Proof The proof is done by induction on the proof of $\rho \in [\![\Phi]\!]_\Delta$ using Lemma 52. □

Definition 55 (Saturated set) *A set $\mathcal{S}$ of semantic objects for Δ and A is a* saturated set *for Δ and A if the following hold:*

(S1) If M is a base term and a semantic object for Δ and A, then $M \in \mathcal{S}$.

(S2) If $N \in \mathcal{S}$ and $\Delta \vdash M \to_{wh} N : A$, then $M \in \mathcal{S}$.

Lemma 56 ($[\![A]\!]_\Delta$ is a saturated set) *$[\![A]\!]_\Delta$ is a saturated set for any valid stack Δ and type A.*

Proof The proof is by induction on the structure of A, using the definition of a semantic object and the thinning lemma (Lemma 34). □

Now we state the key lemma of this section: soundness of the typed operational semantics for our calculus.

Lemma 57 (Soundness) *If $\Phi \vdash M : A$ and $\rho \in [\![\Phi]\!]_\Delta$, then $(\overline{\rho}\ M) \in [\![A]\!]_\Delta$.*

Proof We proceed by induction on derivations of $\Delta \vdash M : A$. We use Lemma 49, Lemma 56, and property (S1) of Definition 55 in the case (λ). We show here the case ($\uparrow$), which is similar to the case (λ) and which deals with modality.

We have $\Phi \vdash \uparrow M : \Box A$ by the rule ($\uparrow$). Suppose $\rho \in [\![\Phi]\!]_\Delta$. We have to prove that $(\overline{\rho}\ \ \uparrow M) \in [\![\Box A]\!]_\Delta$. We first need to show that there exists a Q such that $\Delta \vdash (\overline{\rho} \uparrow M) \rightarrow_{nf} Q : \Box A$. By induction hypothesis, we know that there exists a P such that $\Delta; . \vdash (\overline{\rho'}\ M) \rightarrow_{nf} P : A$, for any $\rho' \in [\![\Phi; .]\!]_{\Delta;.}$. Hence if $P \not\equiv \downarrow N$, then $\Delta \vdash (\overline{\rho'} \uparrow M) \rightarrow_{nf} \uparrow P : \Box A$ by rule $(S\uparrow)$, and otherwise $P \equiv \downarrow N$ and $\Delta \vdash (\overline{\rho'} \uparrow M) \rightarrow_{nf} N : \Box A$ by rule $(S\eta_\Box)$. Let Q be the normal form in either case. We take $\rho' = \rho; \text{weak}^{\Delta;.}_{.;.}$.

Then we need to show that $(\overline{\rho} \downarrow\uparrow M) \in [\![A]\!]_{\Delta,\Delta'}$ for any stack Δ' such that Δ, Δ' is valid. By induction hypothesis, we know that $(\overline{\rho'}\ M) \in [\![A]\!]_{\Delta,\Delta'}$ for any $\rho' \in [\![\Phi; .]\!]_{\Delta,\Delta'}$. So by definition of semantic objects, we know that $\Delta, \Delta' \vdash (\overline{\rho'}\ M) \rightarrow_{nf} M' : A$ for some M'. So $\Delta, \Delta' \vdash (\overline{\rho'} \downarrow\uparrow M) = \downarrow\uparrow(\overline{\rho'}\ M) \rightarrow_{wh} (\overline{\rho'}\ M) : A$ by the rule $(W\beta_\Box)$. So, by (S2), $(\overline{\rho'} \downarrow\uparrow M) \in [\![A]\!]_{\Delta,\Delta'}$. To conclude, we take $\rho' = \rho; \text{weak}^{\Delta,\Delta'}_{.;.}$. Then $(\overline{\rho} \downarrow\uparrow M) = (\overline{\rho'} \downarrow\uparrow M) \in [\![A]\!]_{\Delta,\Delta'}$, and so $(\overline{\rho}\ \ \uparrow M) \in [\![\Box A]\!]_\Delta$ by definition. $\Box$

Since the identity substitution from Δ to Δ belongs to $[\![\Delta]\!]_\Delta$, we are able to link typing judgments with $\rightarrow_{nf}$ judgments.

Lemma 58 *If Δ is a context stack, then $id_\Delta \in [\![\Delta]\!]_\Delta$.*

Proof The proof is straightforward, by induction on the structure of Δ, using Lemma 56 and property (S1) of Definition 55. $\Box$

Corollary 59 *If $\Delta \vdash M : A$, then there is a term P such that $\Delta \vdash M \rightarrow_{nf} P : A$.*

Proof The proof is done by soundness, Lemma 58, and the definition of $[\![A]\!]_\Delta$. $\Box$

Lemma 60 (Soundness for $\Delta \vdash M = N : A$) *If $\Delta \vdash M = N : A$, then there is a P such that $\Delta \vdash M \rightarrow_{nf} P : A$ and $\Delta \vdash N \rightarrow_{nf} P : A$.*

Proof We proceed by induction on derivations that $\Delta \vdash M = N : A$. In the case of transitivity, we have

$$\Delta \vdash M = P : A \quad \text{and} \quad \Delta \vdash P = N : A.$$

Thus, by induction hypothesis, we know that there exist terms Q and Q' such that

$$\Delta \vdash M \rightarrow_{nf} Q : A,$$
$$\Delta \vdash P \rightarrow_{nf} Q : A,$$
$$\Delta \vdash P \rightarrow_{nf} Q' : A,$$
$$\Delta \vdash N \rightarrow_{nf} Q' : A.$$

It seems we cannot conclude, because we do not know that $Q \equiv Q'$. This can be proved by using previous results. First, by adequacy of $\rightarrow_{nf}$ judgments for reduction (Lemma 42),

$$P \hookrightarrow_* Q, \quad \text{with } Q \text{ normal},$$
$$P \hookrightarrow_* Q', \quad \text{with } Q' \text{ normal}.$$

Then, by Lemma 48 (diamond property for $\rightarrow_{nf}$), $Q \hookrightarrow_* Q'$, which means that $Q \equiv Q'$ since both terms are normal. □

Soundness for $\Delta \vdash M = N : A$ gives us the three expected results of strong normalization, subject reduction, and Church-Rosser.

Corollary 61 (Strong Normalization) *If $\Delta \vdash M : A$, then M is strongly normalizing.*

Corollary 62 (Subject Reduction) *If $\Delta \vdash M : A$ and $M \hookrightarrow N$, then $\Delta \vdash N : A$.*

Corollary 63 (Church-Rosser) *If $\Delta \vdash M = N : A$, then there is a P such that $M \hookrightarrow_* P$ and $N \hookrightarrow_* P$.*

7 Conclusion

We have presented in this paper the proofs of strong normalization, subject reduction, and Church-Rosser theorems for a modal λ-calculus S4 [PW95]. The proof followed the typed operational semantics method introduced by Goguen [Gog94, Gog95] for the simply typed λ-calculus. We have succeeded in adapting his method despite the fact that our typing rules are not syntax-driven and we had to deal with context stacks instead of simple contexts. In the course of the proof, we have extracted an interesting notion of typed substitution for our modal system. Moreover, some definitions we used are different, maybe clearer and more general than the original ones. For instance, in Section 5.3, we avoided the use of the predicate $\mathcal{S}_{\Delta}^{P,A}(M)$, its logical meaning being unclear to us. Later, in the definitions of the interpretations of types, we used weakenings instead of renamings. Furthermore, the appendix fixes an omission in the original paper.

It has been shown recently how modality IS4 can be used to construct a simple type system that allows primitive recursion on terms of a higher-order abstract syntax [DPS97]. Using the developments presented in this paper (in particular, the definitions and lemmas of Section 6), we were able to propose a variant of this type system, with simpler reduction rules and a much shorter proof [DL98]. We hope that the results presented there will serve as a basis for the proofs for an extension of the type system to a richer calculus, including polymorphic and dependent types along the lines of [DL99].

Modal λ-calculus IS4 has other important applications. For instance, it can be used to express different stages of computation in a program [DP96]. The study of any such system, based on a modal λ-calculus IS4, should greatly benefit from the proof techniques presented in this work.

8 Appendix: Technical development

The results proved in this appendix are rather technical. They are used in the proofs of Lemmas 46 and 47.

Definition 64 (weak-head reducible) *Terms that are not whn are called* weak-head reducible *(whr).*

Lemma 65 *The whr terms are of the following form:*

$$whr := (\lambda x : A.P\ N) \mid \downarrow\uparrow M \mid (whr\ N) \mid \downarrow whr.$$

Definition 66 (elements) *We recursively define the function elem, which maps a term whr to a list of terms (its* elements*) by the following rules:*

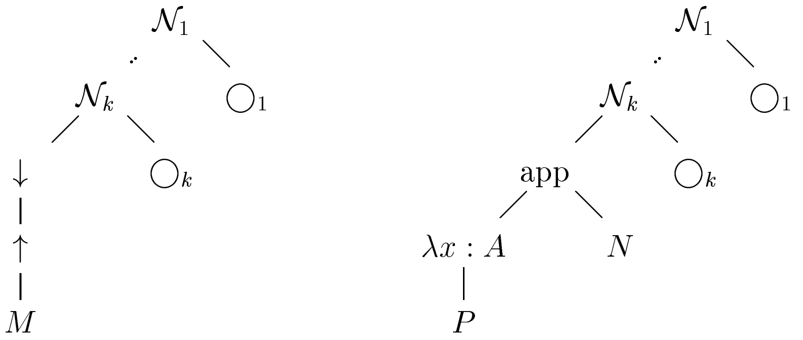
- $elem(\lambda x : A.P\ N) = (N, \lambda x : A.P)$,
- $elem(\downarrow\uparrow M) = \uparrow M$,
- $elem(whr\ N) = (N, elem(whr))$,

- $elem(\downarrow whr) = elem(whr)$.

We also define the function *head*, which maps a whr term to a term.

Definition 67 (head) *If M is whr and $elem(M) = (M_1, \ldots, M_k)$, we define $head(M) := M_k$.*

To summarize, the possible forms of weak-head reducible terms are the following ones:



where $\bigcirc_1, \ldots, \bigcirc_k$ are subtrees, all of the operators $\mathcal{N}_1, \ldots, \mathcal{N}_k$ are either app (i.e., application) or $\downarrow$ (in which case the corresponding subtree $\bigcirc_j$ is empty).

For the first term, the *elem* function returns the list $(\bigcirc_1, \ldots, \bigcirc_k, \uparrow M)$ and the *head* function returns $\uparrow M$. For the second term, the *elem* function returns the list $(\bigcirc_1, \ldots, \bigcirc_k, N, \lambda x : A.P)$ and the *head* function returns $\lambda x : A.P$.

By definition, the *structure* of a whr term is the list $(\mathcal{N}_1, \ldots, \mathcal{N}_k, \downarrow)$ (or $(\mathcal{N}_1, \ldots, \mathcal{N}_k, \text{app})$). A whr term is entirely defined by its structure and the list of its elements.

Lemma 68 *We have that*

$$(\Delta \vdash M \to_{wh} P : B) \Rightarrow (M \text{ is whr}).$$

Proof The proof is easy, by induction on the proof of the hypothesis. $\square$

Lemma 69 *We have*

$$(\Delta \vdash M \to_{wh} P : B) \Rightarrow \left(\begin{array}{l} \forall N \in elem(M), \\ \exists n \in I\!N\ , N' \text{ and } C \text{ s.t. } \Delta^n \vdash N \to_{nf} N' : C \end{array} \right).$$

Proof The proof is easy, by induction on the proof of the hypothesis. □

The following lemma allows us to solve some difficulties associated with truncated stacks. It says that if a term is well typed in a stack Δ^n of type A and has a $\to_{nf}$ evaluation in another stack Δ^m of type B, then it has the same $\to_{nf}$ evaluation in Δ^n and $A = B$.

Lemma 70 *If $\Delta^n \vdash M : A$ ($n \in \mathbb{N}$), then*

- *if $\Delta^m \vdash M \to_{nf} P : B$ ($m \in \mathbb{N}$), then $\Delta^n \vdash M \to_{nf} P : A$,*
- *if $\Delta^p \vdash M \to_{wh} N : C$ ($p \in \mathbb{N}$), then $\Delta^n \vdash M \to_{wh} N : A$.*

Proof The proof is by induction on the proofs of the hypotheses $\Delta^m \vdash M \to_{nf} P : B$ and $\Delta^p \vdash M \to_{wh} N : C$, using the inversion lemmas on the typing hypothesis. □

Definition 71 *For M a whr term such that each N belonging to $elem(M)$ is strongly normalizing, we define $\mu(M)$ as $\mu(M) := \sum_{N \in elem(M)} \nu(N)$, where $\nu(N)$ is the maximum number of reduction steps starting from N.*

Lemma 72 *If $\Delta \vdash M : A$, M whr and $M \hookrightarrow_+ N$ (without weak-head reduction steps), then N is whr, it has the same structure as M, and its elements are obtained from those of M by reductions. Moreover, if all of the elements of M are strongly normalizing, then $\mu(M)$ and $\mu(N)$ are defined and $\mu(N) < \mu(M)$.*

Proof The proof is by induction on the proof of M whr. □

Note Terms M such that $M \hookrightarrow_* N$ both with and without weak-head reduction steps are seen as reducing with a weak-head reduction step.

Lemma 73 *If $\Delta \vdash M : A$, $M \hookrightarrow_{wh} N$, $M \hookrightarrow_* P$ (without weak-head reduction steps), then there is a term Q such that $P \hookrightarrow_{wh} Q$ and $N \hookrightarrow_* Q$.*

Proof The proof is done by induction on the proof of $M \hookrightarrow_{wh} N$. □

Lemma 74 *If $\Delta \vdash M : A$, $M \hookrightarrow_{wh} N$, and, for each element M_k of M, $\Delta^{n_k} \vdash M_k \to_{nf} P_k : B_k$, then $\Delta \vdash M \to_{wh} N : A$.*

Proof The proof is by induction on the proof of $M \hookrightarrow_{wh} N$, using Lemma 70. □

Acknowledgements

We gratefully acknowledge discussions with Healfdene Goguen and André Hirschowitz.

A full version of this paper with complete technical developments is available in [Lel98].

References

[BdP96] Gavin Bierman and Valeria de Paiva. Intuitionistic necessity revisited. Technical Report CSRP-96-10, School of Computer Science, University of Birmingham, June 1996.

[CG97] Adriana Compagnoni and Healfdene Goguen. Typed operational semantics for higher order subtyping. Technical Report ECS-LFCS-97-361, University of Edinburgh, July 1997. Submitted for publication.

[Che90] Brian F. Chellas. *Modal Logic: An Introduction.* Cambridge University Press, 1990.

[DL98] Joëlle Despeyroux and Pierre Leleu. A modal λ-calcul with iteration and case constructs. In *Proceedings of the Annual Types for Proofs and Programs Seminar*, volume 1657 of *Lecture Notes in Computer Science*. Springer-Verlag, 27–31 March 1998.

[DL99] Joëlle Despeyroux and Pierre Leleu. Primitive recursion for higher-order abstract syntax with dependant types. In *Proceedings of the FLoC'99 IMLA Workshop on Intuitionistic Modal Logics and Applications*, June 1999.

[DP96] Rowan Davies and Frank Pfenning. A modal analysis of staged computation. In Guy Steele Jr., editor, *Proceedings of the 23rd Annual Symposium on Principles of Programming Languages*, pages 258–270, St. Petersburg Beach, Florida, January 1996. ACM Press.

[DPS97] Joëlle Despeyroux, Frank Pfenning, and Carsten Schürmann. Primitive recursion for higher-order abstract syntax. In Philippe de Groote and J. Roger Hindley, editors, *Proceedings of the TLCA*

97 International Conference on Typed Lambda Calculi and Applications, Nancy, France, volume 1210 of *Lecture Notes in Computer Science*, pages 147–163. Springer-Verlag, April 2–4 1997. An extended version is available as CMU Technical Report CMU-CS-96-172.

[GL96] Jean Goubault-Larrecq. On computational interpretations of the modal logic S4, part I. Cut elimination. Technical Report 1996-35, Institut für Logik, Komplexität und Deduktionssysteme, Universität Karlsruhe, Karlsruhe, Germany, 1996.

[Gog94] Healfdene Goguen. *A Typed Operational Semantics for Type Theory*. PhD thesis, University of Edinburgh, August 1994.

[Gog95] Healfdene Goguen. Typed operational semantics. In *Proceedings of the International Conference on Typed Lambda Calculi and Applications*, volume 902 of *Lecture Notes in Computer Science*, pages 186–200. Springer-Verlag, 1995.

[Gog99] Healfdene Goguen. Soundness of typed operational semantics for the logical framework. In Jean-Yves Girvard, editor, *Proceedings of the International Conference on Typed Lambda Calculi and Applications, TLCA*, pages 177–197. Springer-Verlag, 1999.

[Lel98] Pierre Leleu. Metatheoretic results for a modal λ-calcul. Research Report RR-3361, INRIA, February 1998.

[PW95] Frank Pfenning and Hao-Chi Wong. On a modal λ-calculus for S4. In S. Brookes and M. Main, editors, *Proceedings of the Eleventh Conference on Mathematical Foundations of Programming Sematics*, New Orleans, Louisiana, March 1995. To appear in *Electronic Notes in Theoretical Computer Science*, Volume 1, Elsevier.