The Logic of Proofs as a Foundation for Certifying Mobile Computation

Eduardo Bonelli[12][⋆] and Federico Feller[2]

[1] Depto. de Ciencia y Tecnología, Universidad Nacional de Quilmes and CONICET
[2] LIFIA, Facultad de Informática, Universidad Nacional de La Plata

Abstract. We explore an intuitionistic fragment of Artëmov's *Logic of Proofs* as a type system for a programming language for *mobile units*. Such units consist of both a code and certificate component. Dubbed the *Certifying Mobile Calculus*, our language caters for both code and certificate development in a unified theory. In the same way that mobile code is constructed out of code components and extant type systems track local resource usage to ensure the mobile nature of these components, our system *additionally* ensures correct *certificate construction* out of certificate components. We present proofs of type safety and strong normalistion for a run-time system based on an abstract machine.

1 Introduction

We explore an intuitionistic fragment (ILP) of Artëmov's *Logic of Proofs* (LP) as a type system for a programming language for *mobile units*. This language caters for both code and certficate development in a unified theory. LP may be regarded as refinement of modal logic S4 in which $\Box A$ is replaced by $[s]A$, for s a *proof term* expression, and is read: *"s is a proof of A"*. It is sound and complete w.r.t. provability in PA (see [Art94] for a precise statement) and realizes all theorems of S4. It therefore provides an answer to the (long-standing) problem of associating an exact provability semantics to S4 [Art94]. LP is purported to have important applications not only in logic but also in Computer Science [AB04]. This work may be regarded as a small step in exploring the applications of LP in programming languages and type theory.

Modal necessity $\Box A$ may be read as the type of programs that compute values of type A and that do not depend on local resources [Moo04,VCHP04,VCH05] or resources not available at the current stage of computation [TS97,WLPD98,DP01b]. The former reading refers to *mobile computation* ($\Box A$ as the type of mobile code that computes values of type A) while the latter to *staged computation* ($\Box A$ as the type of code that generates, at run-time, a program for computing a value of type A). See Sec. 7 for further references. We introduce the *Certifying Mobile Calculus* or $\lambda^{\mathsf{Cert}}_{\Box}$ by taking a mobile computation interpretation of ILP. ILP's mechanism for internalizing its own derivations provides a natural setting for code certification. A contribution of our approach is that, in the same way that mobile code is constructed out of code components and extant type systems track local resource usage to ensure the mobile nature of these components, our system *additionally* ensures correct *certificate construction* out of certificate components. Mobile units consist of both a code component and a certificate component. A sample $\lambda^{\mathsf{Cert}}_{\Box}$ expression,

[⋆] Work partially supported by Instituto Tecnológico de Buenos Aires

one encoding a proof of the ILP axiom scheme $[s](A \supset B) \supset [t]A \supset [s \cdot t]B$ where s, t are any proof term expressions and A, B any propositions, is the following:

$$\lambda a.\lambda b.unpack\ a\ to\ \langle u^\bullet, u^\circ\rangle\ in\ (unpack\ b\ to\ \langle v^\bullet, v^\circ\rangle\ in\ (box_{u^\circ \cdot v^\circ}\ u^\bullet v^\bullet))$$

This is read as follows: *"Given a mobile unit a and a mobile unit b, extract code $v^\bullet$ and certificate v° from b and extract code $u^\bullet$ and certificate u° from a. Then create new code $u^\bullet\, v^\bullet$ by applying $u^\bullet$ to $v^\bullet$ and a new certificate for this code $u^\circ \cdot v^\circ$. Finally, wrap both of these up into a new mobile unit."*. The syntax of code and certificates is described in detail in Sec. 3. The new mobile unit is created at the some current (implicit) world w. Moreover, the example assumes that both a and b reside at w. The following variant M illustrates the case where mobile units a and b reside at worlds w_a and w_b which are assumed different from the current world w:

$$unpack\ fetch[w_a]\, a\ to\ \langle u^\bullet, u^\circ\rangle\ in\ (unpack\ fetch[w_b]\, b\ to\ \langle v^\bullet, v^\circ\rangle\ in\ (box_{u^\circ \cdot v^\circ}\ u^\bullet v^\bullet))$$

Here the expression $fetch[w_a]\, a$ is operationally interpreted as a remote call to compute the value of a (a mobile unit) at w_a and then return it to the current world. Note that a and b occur free in this expression. Since b is a non-local resource it cannot be bound straightforwardly by prefixing the above term with λb. Rather, the code first must be moved from the current world w to w_b; similarly for a:

$$\lambda a.fetch[w_b]\, (\lambda b.fetch[w]\, M)$$

$\lambda^{\mathsf{Cert}}_{\Box}$ arises from a Curry-de Bruijn-Howard interpretation of a Natural Deduction presentation of ILP based on a judgemental analysis of the Logic of Proofs given in [AB07]. Propositions and proofs of ILP correspond to types and terms of $\lambda^{\mathsf{Cert}}_{\Box}$. Regarding semantics, we provide an operational reading of expressions encoding proofs in this system in terms of global computation. An abstract machine is introduced that computes over multiple worlds. Apart from the standard lambda calculus expressions new expressions for constructing mobile units and for computing in remote worlds are introduced. We state and prove *type safety* of a type system for $\lambda^{\mathsf{Cert}}_{\Box}$ w.r.t. its operational semantics. Also, we prove strong normalization.

This paper is organized as follows. Sec. 2 briefly recapitulates ILPnd [AB07], a Natural Deduction presentation of ILP. We then introduce a term assignment for ILPnd and discuss differences with the term assignment in [AB07] including the splitting of validity variables [AB07] into code and certificate variables. Sec. 4 introduces the run-time system of $\lambda^{\mathsf{Cert}}_{\Box}$, the abstract machine for execution of $\lambda^{\mathsf{Cert}}_{\Box}$ programs. Sec. 5 analyses type safety and Sec. 6 strong normalization. References to related work follows. Finally, we conclude and suggest further directions for research.

2 Natural Deduction for ILP

As mentioned, LP [Art95,Art01] is a refinement of modal logic S4 in which $\Box A$ is replaced by $[s]A$. Here s is an expression representing a Hilbert style proof and is called a *proof polynomial*. In the minimal propositional logic fragment of LP without plus, ILP, proof polynomials are constructed from proof variables and constants using two operations: application "·" and proof-checker "!". The usual propositional connectives are augmented by a new one: given a proof polynomial s and a proposition A build $[s]A$. The intended reading is: "*s is a proof of A*". The axioms and inference schemes of LP are:

A0. Axiom schemes of minimal logic in the language of LP
A1. $[s]A \supset A$ *"verification"*
A2. $[s](A \supset B) \supset ([t]A \supset [s \cdot t]B)$ *"application"*
A3. $[s]A \supset [!s][s]A$ *"proof checker"*
R1. $\Gamma \rhd A \supset B$ and $\Gamma \rhd A$ implies $\Gamma \rhd B$ *"modus ponens"*
R2. If $\mathbf{A}$ is an axiom **A0-A3**, and c is a proof constant, then $\rhd [c]\mathbf{A}$ *"necessitation"*

For verification one reads: "*if s is a proof of A, then A holds*". As regards the proof polynomials the standard interpretation is as follows. For application one reads: "*if s is a proof of $A \supset B$ and t is a proof of A, then $s \cdot t$ is a proof of B*". Thus "$\cdot$" represents composition of proofs. For proof checking one reads: "*if s is a proof of A, then $!s$ is a proof of the sentence* 's is a proof of A' ". Thus $!s$ is seen as a computation that verifies $[s]A$.

In previous work [AB07] a Natural Deduction presentation of ILP (ILPnd) is introduced by considering two sets of hypothesis, truth and validity hypothesis, and analysing the meaning of the following Hypothetical Judgement with Explicit Evidence:

$$\Delta; \Gamma \rhd A \mid s$$

Here Δ is a sequence of *validity assumptions*, Γ a sequence of *truth assumptions*, A is a proposition and s is a proof term. A validity assumption is written $v : A$ where v ranges over a given infinite set of *validity variables* and states that A holds at all accessible worlds. Likewise, a truth assumption is written $a : A$ where a ranges over a given infinite set of *truth variables* and states that A holds at the current world. We write x to denote either of these variables. The judgement is read as: "*A is true with evidence s under validity assumptions Δ and truth assumptions Γ*". Note that s is a constituent of this judgement without whose intended reading is not possible. The meaning of this judgement is given by axiom and inference schemes (Fig. 1). We say a judgement is *derivable* if it has a derivation using these schemes.

Proof Terms $s, t ::= x \mid s \cdot t \mid \lambda a : A.s \mid !s \mid \text{LETC}\, s \,\text{BE}\, v : A \,\text{IN}\, t$
Propositions $A, B ::= P \mid A \supset B \mid [s]A$
Truth Contexts $\Gamma ::= \cdot \mid \Gamma, a : A$
Validity Contexts $\Delta ::= \cdot \mid \Delta, v : A$

All free occurrences of a (resp. v) in s are bound in $\lambda a : A.s$ (resp. $\text{LETC}\, t \,\text{BE}\, v : A \,\text{IN}\, s$). A proposition is either a variable P, an implication $A \supset B$ or a validity proposition $[s]A$. We write "$\cdot$" for empty contexts and $s\{x/t\}$ for the result of substituting all free occurrences of x in s by t (bound variables are renamed whenever necessary); likewise for $A\{x/t\}$.

A brief informal explanation of some of these schemes follows. The axiom scheme oVar states that the judgement $\Delta; \Gamma, a : A, \Gamma' \rhd A \mid a$ is evident in itself. Indeed, if we

Minimal Propositional Logic Fragment

$$\frac{}{\Delta; \Gamma, a : A, \Gamma' \rhd A \mid a}\ \mathsf{oVar}$$

$$\frac{\Delta; \Gamma, a : A \rhd B \mid s}{\Delta; \Gamma \rhd A \supset B \mid \lambda a : A.s}\ \supset \mathsf{I} \qquad \frac{\Delta; \Gamma \rhd A \supset B \mid s \quad \Delta; \Gamma \rhd A \mid t}{\Delta; \Gamma \rhd B \mid s \cdot t}\ \supset \mathsf{E}$$

Provability Fragment

$$\frac{}{\Delta, v : A, \Delta'; \Gamma \rhd A \mid v}\ \mathsf{mVar}$$

$$\frac{\Delta; \cdot \rhd A \mid s}{\Delta; \Gamma \rhd [s]A \mid !s}\ \Box\mathsf{I} \qquad \frac{\Delta; \Gamma \rhd [r]A \mid s \quad \Delta, v : A; \Gamma \rhd C \mid t}{\Delta; \Gamma \rhd C\{v/r\} \mid \textsc{letc}\, s\, \textsc{be}\, v : A\, \textsc{in}\, t}\ \Box\mathsf{E}$$

$$\frac{\Delta; \Gamma \rhd A \mid s \quad \Delta; \Gamma \vdash s \equiv t : A}{\Delta; \Gamma \rhd A \mid t}\ \mathsf{EqEvid}$$

Fig. 1. Explanation for Hypothetical Judgements with Explicit Evidence

assume that a is evidence that proposition A is true, then we immediately conclude that A is true with evidence a. The introduction scheme for the $[s]$ modality internalises metalevel evidence into the object logic. It states that if s is unconditional evidence that A is true, then A is in fact valid with witness s (i.e. $[s]A$ is true). Evidence for the truth of $[s]A$ is constructed from the (verified) evidence that A is unconditionally true by prefixing it with a bang constructor. Finally, $\Box\mathsf{E}$ allows the discharging of validity hypothesis. In order to discharge the validity hypothesis $v : A$, a proof of the validity of A is required. In this system, this requires proving that $[r]A$ is true with evidence s, for some evidence of proof r and s. Note that r is evidence that A is unconditionally true (i.e. valid) whereas s is evidence that $[r]A$ is true. The former is then substituted in the place of all free occurrences of v in the proposition C. This construction is recorded with evidence $\textsc{letc}\, s\, \textsc{be}\, v : A\, \textsc{in}\, t$ in the conclusion.

Since ILPnd internalizes its own derivations and normalisation introduces identities on derivations at the meta-level, such identities must be reflected in the object-logic too. This is the aim of EqEvid. The schemes defining the judgement of evidence equality $\Delta; \Gamma \vdash s \equiv t : A$ are the axioms for β equality and β equality on $\Box$ together with appropriate congruence schemes (consult [AB07] for details). It should be noted that soundness of ILPnd with respect to ILP does not require the presence of EqEvid. It is, however, required in order for normalisation to be closed over the set of derivations.

A sample derivation in ILPnd of $[s](A \supset B) \supset [t]A \supset [s \cdot t]B$ follows, where $\Gamma = a : [s](A \supset B), b : [t]A$ and $\Delta = u : A \supset B, v : A$:

$$\dfrac{\dfrac{\cdot;\Gamma \rhd [s](A\supset B)\mid a \qquad \dfrac{u:(A\supset B);\Gamma\rhd [t]A\mid b \qquad \dfrac{\dfrac{\Delta;\cdot\rhd A\supset B\mid u \quad \Delta;\cdot\rhd A\mid v}{\Delta;\cdot\rhd B\mid u\cdot v}}{\Delta;\Gamma\rhd [u\cdot v]B\mid !(u\cdot v)}\;\Box\mathsf{I}}{u:(A\supset B);\Gamma\rhd [u\cdot t]B\mid \mathrm{LETC}\,b\,\mathrm{BE}\,v:A\,\mathrm{IN}\,!(u\cdot v)}\;\Box\mathsf{E}}{\dfrac{\cdot;\Gamma\rhd [s\cdot t]B\mid \mathrm{LETC}\,a\,\mathrm{BE}\,u:A\supset B\,\mathrm{IN}\,\mathrm{LETC}\,b\,\mathrm{BE}\,v:A\,\mathrm{IN}\,!u\cdot v}{\cdot;a:[s](A\supset B)\rhd [t]A\supset [s\cdot t]B\mid \lambda b:[s]A.\mathrm{LETC}\,a\,\mathrm{BE}\,u:A\supset B\,\mathrm{IN}\,\mathrm{LETC}\,b\,\mathrm{BE}\,v:A\,\mathrm{IN}\,!(u\cdot v)}\;\supset\mathsf{I}}\;\Box\mathsf{E}}{\cdot;\cdot\rhd [s](A\supset B)\supset [t]A\supset [s\cdot t]B\mid \lambda a:[s](A\supset B).\lambda b:[t]A.\mathrm{LETC}\,a\,\mathrm{BE}\,u:A\supset B\,\mathrm{IN}\,\mathrm{LETC}\,b\,\mathrm{BE}\,v:A\,\mathrm{IN}\,!(u\cdot v)}\;\supset\mathsf{I}$$

3 Term assignment

We assume a set $\{w_1, w_2, \ldots\}$ of worlds, a set $\{v_1^\bullet, v_2^\bullet, \ldots\}$ of code variables and a set $\{v_1^\circ, v_2^\circ, \ldots\}$ of certificate variables. We use Σ for a (finite) set of worlds. Δ and Γ are as before. The syntactic categories of *certificates*, *values* and *terms* are defined as follows:

$$\begin{aligned} s,t &::= a \,|\, v^\circ \,|\, s\cdot t \,|\, \lambda a:A.s \,|\, !s \,|\, \mathit{letc}\ s\ \mathit{be}\ v^\circ : A\ \mathit{in}\ t \,|\, \mathit{fetch}(s) \\ V &::= \mathit{box}_s\, M \,|\, \lambda a.M \\ M,N &::= a \,|\, v^\bullet \,|\, V \,|\, M\,N \\ &\;\;|\;\; \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N \,|\, \mathit{fetch}[w]\, M \end{aligned}$$

Certificates have two kinds of variables. Local variables a are used for abstracting over local assumptions when constructing certificates. Certificate variables v° represent unknown certificates. $s\cdot t$ is certificate composition. $!s$ is certificate endorsement. $\mathit{letc}\ s\ \mathit{be}\ v^\circ : A\ \mathit{in}\ t$ is certificate validation, the inverse operation to endorsement. Finally, $\mathit{fetch}(s)$ certifies the *fetch* code movement operation to be described shortly. An example of a certificate is the following, which encodes a derivation of the first example presented in the introduction:

$$\lambda a:[s](A\supset B).\lambda b:[t]A.\mathit{letc}\ a\ \mathit{be}\ u^\circ : A\supset B\ \mathit{in}\ (\mathit{letc}\ b\ \mathit{be}\ v^\circ : A\ \mathit{in}\ !(u^\circ\cdot v^\circ))$$

Values are a subset of terms that represent the result of computations of well-typed, closed terms. A value of the form $\lambda a.M$ is an abstraction (free occurrences of a in M are bound as usual) and one of the form $\mathit{box}_s\, M$ is a *mobile unit* (composed of mobile code M and certificate s). A *term* is either a term variable for local code a, a term variable for mobile code $v^\bullet$, a value V, an application term $M\,N$, an unpacking term for extraction of code-certificate pairs from mobile units $\mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N$ (free occurrences of v° and $v^\bullet$ in N are bound by this construct) or a fetch term $\mathit{fetch}[w]\, M$. In an unpacking term, M is the argument and N is the body; in a fetch term we refer to w as the target of the *fetch* and M as its body. The operational semantics of these constructs is discussed in Sec. 4.

The term assignment results essentially (the differences are explained below) from the schemes of Fig. 1 with terms encoding derivations and localizing the hypothesis in Δ, Γ at specific worlds. Also, a reference to the current world is added. Typing judgements take the form

$$\Sigma;\Delta;\Gamma \rhd M : A@w \,|\, s \tag{1}$$

Validity and truth contexts are now sequences of expressions of the form $v : A@w$ and $a : A@w$, respectively. The former indicates that mobile unit v computing a value of type A may be assumed to exist and to be located at world w. The latter indicates that a local value a of type A may be assumed to exist at world w. The truth of a proposition at w shall rely, on the one hand, on truth hypothesis in Γ that are located at w, and on the other, on validity hypothesis in Δ that have been fetched, from their appropriate hosts, to the current location w. Logical connectives bind tighter than @, therefore an expression such as $A \supset B@w$ should be read as $(A \supset B)@w$.

It should be mentioned that ILP is not a hybrid logic [AtC06]. In other words, $A@w$ is not a proposition of our object logic. For example, expressions of the form $A@w \supset B@w'$ are not valid propositions.

Typing schemes defining (1) are presented in Fig. 2 and discussed below. A first difference with ILPnd is that the scheme EqEvid has been dropped. Although the latter is required for normalisation of derivations to be a closed operation (as already mentioned), our operational interpretation of terms does not rely on normalisation of Natural Deduction proofs. For a computational interpretation of ILP based on normalisation the reader may consult [AB07]. A further difference is that □I has been refined into two schemes, namely $\Box I$ and Fetch. The first introduces a modal formula and states it to be true at the current world w. The second states that all worlds accesible to w may also assume this formula to be true.

In this work mobile code is accompanied by a certificate. We speak of *mobile units* rather than mobile code to emphasize this. Since mobile units are expressions of modal types and validity variables v represent holes for values of modal types, validity variables v may actually be seen as pairs $\langle v^\bullet, v^\circ \rangle$. Here $v^\bullet$ is the mobile code component and v° is the certificate component of the mobile unit[3]. As a consequence, the modality axiom mVar of ILPnd now takes the following form, where judgement $\Sigma \vdash w$ ensures w is a world in Σ (it is defined by requiring $w \in \Sigma$):

$$\frac{\Sigma \vdash w}{\Sigma; \Delta, v : A@w, \Delta'; \Gamma \rhd v^\bullet : A@w \mid v^\circ} \text{VarV}$$

Substitution of code variables for terms in terms ($M\{v^\bullet/N\}$) and substitution of certificate variables for certificates in certificates ($t\{v^\circ/s\}$) and in terms ($M\{v^\circ/s\}$) is defined as expected. We illustrate the definition of the first of these notions.

$$\begin{aligned}
a\{v^\bullet/N\} &=_{def} a \\
v^\bullet\{v^\bullet/N\} &=_{def} N \\
u^\bullet\{v^\bullet/N\} &=_{def} u^\bullet \\
(PQ)\{v^\bullet/N\} &=_{def} P\{v^\bullet/N\}Q\{v^\bullet/N\} \\
(\lambda a : A.P)\{v^\bullet/N\} &=_{def} \lambda a : A.P\{v^\bullet/N\} \\
(box_t\, P)\{v^\bullet/N\} &=_{def} box_t\, P\{v^\bullet/N\} \\
(fetch[w]\, P)\{v^\bullet/N\} &=_{def} fetch[w]\, P\{v^\bullet/N\} \\
(unpack\ P\ to\ \langle u^\bullet, u^\circ\rangle\ in\ Q)\{v^\bullet/N\} &=_{def} unpack\ P\{v^\bullet/N\}\ to\ \langle u^\bullet, u^\circ\rangle\ in\ Q\{v^\bullet/N\}
\end{aligned}$$

[3] The "∘" is reminiscent of a wrapping with which the interior "•" is protected. Hence our use of the former for certificates and the latter for code.

$$\dfrac{\Sigma \vdash w}{\Sigma; \Delta; \Gamma, a : A@w, \Gamma' \rhd a : A@w \,|\, a}\ \text{VarT}$$

$$\dfrac{\Sigma; \Delta; \Gamma, a : A@w \rhd M : B@w \,|\, s}{\Sigma; \Delta; \Gamma \rhd \lambda a.M : A \supset B@w \,|\, \lambda a : A.s}\supset I \qquad \dfrac{\Sigma; \Delta; \Gamma \rhd M : A \supset B@w \,|\, s \quad \Sigma; \Delta; \Gamma \rhd N : A@w \,|\, t}{\Sigma; \Delta; \Gamma \rhd M\, N : B@w \,|\, s \cdot t}\supset E$$

$$\dfrac{\Sigma \vdash w}{\Sigma; \Delta, v : A@w, \Delta'; \Gamma \rhd v^{\bullet} : A@w \,|\, v^{\circ}}\ \text{VarV}$$

$$\dfrac{\Sigma; \Delta; \cdot \rhd M : A@w \,|\, s}{\Sigma; \Delta; \Gamma \rhd box_s\, M : [s]A@w \,|\, !s}\Box I \qquad \dfrac{\Sigma; \Delta; \Gamma \rhd M : [s]A@w' \,|\, t \quad \Sigma \vdash w}{\Sigma; \Delta; \Gamma \rhd fetch[w']\, M : [s]A@w \,|\, fetch(t)}\ \text{Fetch}$$

$$\dfrac{\Sigma; \Delta; \Gamma \rhd M : [r]A@w \,|\, s \quad \Sigma; \Delta, v : A@w; \Gamma \rhd N : C@w \,|\, t}{\Sigma; \Delta; \Gamma \rhd unpack\ M\ to\ \langle v^{\bullet}, v^{\circ}\rangle\ in\ N : C\{v^{\circ}/r\}@w \,|\, letc\ s\ be\ v : A\ in\ t}\Box E$$

Fig. 2. Term assignment for ILPnd

The schemes $\supset I$ and $\supset E$ form abstractions and applications at the current world w. Applications of these schemes are reflected in their corresponding certificates. Scheme $\Box I$ states that if we have a typing derivation of M that does not depend on local assumptions (although it may depend on assumptions universally true) and s is a witness to this fact, then M is in fact executable at an arbitrary location. Thus a mobile unit $box_s\, M$ is introduced. The Fetch scheme types the *fetch* instruction. A term of the form $fetch[w']\, M$ at world w is typed by considering M at world w'. We are in fact assuming that w sees w' (or that w' is accesible from w) at run-time. Moreover, since the result of this instruction is to compute M at w' and then *return* the result to w (cf. Sec. 4), worlds w' and w are assumed interaccesible[4]. The *unpack* instruction is typed using the scheme $\Box E$. Suppose we are given a term N that computes some value of type C at world w and depends on a validity hypothesis $v : A@w$. Suppose we also have a term M that computes a mobile unit of type $[r]A@w$ at the same world w. Then $unpack\ M\ to\ \langle v^{\bullet}, v^{\circ}\rangle\ in\ N$ is well-typed at w and computes a value of type $C\{v^{\circ}/r\}$. The certificate $letc\ s\ be\ v : A\ in\ t$ encodes the application of this scheme.

The following substitution principles reveal the true hypothetical nature of hypothesis, both for truth and for validity. Both are proved by induction on the derivation of the second judgement.

Lemma 1 (Substitution principle for truth hypothesis). *If $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd M : A@w \,|\, s$ and $\Sigma; \Delta; \Gamma_1, a : A@w, \Gamma_2 \rhd N : B@w' \,|\, t$ are derivable, then so is $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd N\{a/M\} : B@w' \,|\, t\{a/s\}$.*

[4] We are considering a term assignment for a Natural Deduction presentation of a refinement of S4 (and not S5; see Lem. 3). This reading, which suggests symmetry of the accesibility relation in a Kripke style model (and hence S5), is part of the run-time interpretation of terms (cf. 7).

Lemma 2 (Substitution principle for validity hypothesis). *If $\Sigma; \Delta_1, \Delta_2; \cdot \rhd M : A@w \,|\, s$ and $\Sigma; \Delta_1, v : A@w, \Delta_2; \Gamma \rhd N : B@w' \,|\, t$ are derivable, then so is $\Sigma; \Delta_1, \Delta_2; \Gamma \rhd N\{v^\circ/s\}\{v^\bullet/M\} : B\{v^\circ/s\}@w \,|\, t\{v^\circ/s\}$.*

Regarding the relation of this type system for $\lambda^{\mathsf{Cert}}_{\Box}$ with ILPnd we have the following result, which may be verified by structural induction on the derivation of the first judgement. Applications of the Fetch scheme become instances of the scheme $\frac{\mathcal{J}}{\mathcal{J}}$ with copies of identical judgements in ILPnd.

Lemma 3. *If $\Sigma; \Delta; \Gamma \rhd A@w \,|\, s$ is derivable, then so is $\Delta'; \Gamma' \rhd A' \mid s'$ in ILPnd, where Δ' and Γ' result from Δ and Γ, respectively, by dropping all location qualifiers and A' and s' result from A and s, respectively, by replacing all occurrences of $v^\bullet$ and v° by v and replacing all certificates of the form $fetch(s)$ with s.*

4 Operational Semantics

The operational semantics of $\lambda^{\mathsf{Cert}}_{\Box}$ follows ideas from [VCHP04]. We introduce an abstract machine over a network of nodes. Nodes are named using worlds. Computation takes place sequentially, at some designated world. We are, in effect, modelling sequential programs that are aware of other worlds (other than their local host), rather than concurrent computation. An *abstract machine state* is an expression of the form $\mathbb{W}; w : [k, M]$ (top of Fig. 3). The world w indicates the node where computation is currently taking place. M is the code that is being executed under local context k (M is the current *focus* of computation). The context k is a stack of terms with holes (written "$\circ$") that represent the layers of terms that are peeled out in order to access the redex. This representation ensures a reduction relation that always operates at the root of an expression and thus allows us to speak of an abstract machine. An alternative presentation based on a small or big-step semantics on terms, rather than machine states, is also possible. Continuing our explanation of the context k, it is a sequence of terms with holes ending in either return w or finish. return w indicates that once the term currently in focus is computed to a value, this value is to be returned to world w. The type system ensures that this value is, in effect, a mobile unit. If k takes the form finish, then the value of the term currently in focus is the end result of the computation. Finally, $k \lhd l$ states that the outermost peeled term layer is l. This latter expression may be of one of the following forms: $\circ\ N$ indicates a pending argument, $V\ \circ$ a pending abstraction (that V is an abstraction rather than a mobile unit is enforced by the type system) and *unpack* $\circ$ *to* $\langle v^\bullet, v^\circ \rangle$ *in* N a pending unpack body.

Finally, $\mathbb{W}$ is called a *network environment* and encodes the current state of execution at the remaining nodes of the network. The *domain of* $\mathbb{W}$ is the set of worlds to which it refers. Also, we sometimes refer to $\mathbb{W}; k$ as the network environment.

The *initial* machine state (over $\Sigma = \{w_1, \ldots, w_n\}$) is $\mathbb{W}; w : [\text{finish}, M]$, where $\mathbb{W} = \{w_1 : \epsilon, \ldots, w_n : \epsilon\}$ and w and M are any world and term, respectively. Similarly, the *terminal* machine state is one of the form $\mathbb{W}; w : [\text{finish}, V]$. Note that in a terminal state the focus of computation is a fully evaluated term (i.e. a value).

Run – time system syntax

$$\begin{array}{rcl}
\mathbb{N} & ::= & \mathbb{W}; w : [k, M] \\
\mathbb{W} & ::= & \{w_1 : C_1, \ldots w_n : C_n\} \\
k & ::= & \text{return } w \,|\, \text{finish} \,|\, k \lhd l \\
l & ::= & \circ\ N \,|\, V\ \circ \,|\, \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^{\bullet}, v^{\circ}\rangle\ \mathit{in}\ N \\
C & ::= & \epsilon \,|\, C\!::k
\end{array}$$

Run – time system reduction schemes

$$\begin{array}{lrcl}
(1) & \mathbb{W}; w : [k, MN] & \longrightarrow & \mathbb{W}; w : [k \lhd \circ\ N, M] \\
(2) & \mathbb{W}; w : [k \lhd \circ\ N, V] & \longrightarrow & \mathbb{W}; w : [k \lhd V\ \circ, N] \\
(3) & \mathbb{W}; w : [k \lhd (\lambda a.M)\ \circ, V] & \longrightarrow & \mathbb{W}; w : [k, M\{a/V\}] \\
(4) & \mathbb{W}; w : [k, \mathit{unpack}\ M\ \mathit{to}\ \langle v^{\bullet}, v^{\circ}\rangle\ \mathit{in}\ N] & \longrightarrow & \mathbb{W}; w : [k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^{\bullet}, v^{\circ}\rangle\ \mathit{in}\ N, M] \\
(5) & \mathbb{W}; w : [k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^{\bullet}, v^{\circ}\rangle\ \mathit{in}\ N, \mathit{box}_s\ M] & \longrightarrow & \mathbb{W}; w : [k, N\{v^{\circ}/s\}\{v^{\bullet}/M\}] \\
(6) & \{w : C; w_s\}; w : [k, \mathit{fetch}[w']\ M] & \longrightarrow & \{w : C\!::k; w_s\}; w' : [\text{return } w, M] \\
(7) & \{w : C\!::k; w_s\}; w' : [\text{return } w, V] & \longrightarrow & \{w : C; w_s\}; w : [k, V\{w'/w\}]
\end{array}$$

Fig. 3. Operational semantics of $\lambda_{\Box}^{\mathsf{Cert}}$

The operational semantics is presented by means of a small-step call-by-value reduction relation whose definition is given by the *reduction schemes* depicted in Fig. 3. The first scheme selects the leftmost term in an application for reduction and pushes the pending part of the term (in this case the argument of the application) into the context. Once a value is attained (which the type system, described below, will ensure to be an abstraction) the pending argument is popped off the context for reduction and the value V is pushed onto the context. Finally, when the argument has been reduced to a value, the pending abstraction is popped off the context and the beta reduct placed into focus for the next computation step. In the case that reduction encounters an *unpack* term, the argument M is placed into focus whilst the rest of the the term is pushed onto the context. When reduction of the argument of an *unpack* computes a value, more precisely a mobile unit, the code and certificate components are extracted from it and substituted in the body of the *unpack* term. Note that the schemes presented upto this point all compute locally, we now address those that operate non-locally. If computation's focus is on a *fetch* instruction, then the execution context k is pushed onto the network environment for the current world w' and control transfers to world w. Moreover, focus of computation is now placed on the term M. Finally, the context of computation at w is set to return w thus ensuring that, once a value is computed, control transfers back to the caller. The latter is the rôle of the final reduction scheme.

5 Type Soundness

This section addresses both *progress* (well-typed, non-terminal machine states are not stuck) and *subject reduction* (well-typed machine states are closed under the reduction). Recall from above that a machine state $\mathbb{N}$ is *terminal* if it is of the form $\mathbb{W}; w : [\text{finish}, V]$. It is *stuck* if it is not terminal and there is no $\mathbb{N}'$ such

$$\frac{}{\Sigma \vdash \mathbb{W}\,;\mathrm{finish} : A@w}\;C.\mathit{Finish}$$

$$\frac{\Sigma \vdash \mathbb{W}\,; k : B@w \quad \Sigma; \cdot ; \cdot \rhd N : A@w \mid s}{\Sigma \vdash \mathbb{W}\,; k \lhd \circ\; N : A \supset B@w}\;C.\mathit{Abs} \qquad \frac{\Sigma \vdash \mathbb{W}\,; k : B@w \quad \Sigma; \cdot ; \cdot \rhd V : A \supset B@w \mid s}{\Sigma \vdash \mathbb{W}\,; k \lhd V\; \circ : A@w}\;C.\mathit{App}$$

$$\frac{\Sigma \vdash \mathbb{W}\,; k : B\{v^{\circ}/t\}@w \quad \Sigma; v : A; \cdot \rhd N : B@w \mid s}{\Sigma \vdash \mathbb{W}\,; k \lhd \mathit{unpack}\;\circ\; \mathit{to}\; \langle v^{\bullet}, v^{\circ} \rangle\; \mathit{in}\; N : [t]A@w}\;C.\mathit{Box}$$

$$\frac{\Sigma \vdash \{w' : C; w_s\}\,; k : A@w'}{\Sigma \vdash \{w' : C{::}\,k; w_s\}\,; \mathrm{return}\; w' : A@w}\;C.\mathit{Return}$$

$$\frac{\begin{array}{cc} \Sigma = \{w_1, \ldots, w_n\} & \mathbb{W} = \{w_1 : C_1, \ldots w_n : C_n\} \\ \Sigma; \cdot ; \cdot \rhd M : A@w_j \mid s & \Sigma \vdash \mathbb{W}\,; k : A@w_j \end{array}}{\Sigma \vdash \mathbb{W}\,; w_j : [k, M]}\;\mathit{MState}$$

Fig. 4. Typing schemes for machine states

that $\mathbb{N} \longrightarrow \mathbb{N}'$. Two new judgements are introduced, machine state judgements and network environment judgements:

- $\Sigma \vdash \mathbb{W}\,; w_j : [k, M]$
- $\Sigma \vdash \mathbb{W}\,; k : A@w_j$

The first states that $\mathbb{W}; w_j : [k, M]$ is a well-typed machine state under the set of worlds Σ. The second states that the network environment together with the local context is well-typed under the set of worlds Σ.

A machine state is well-typed (Fig. 4) if the following three requirements hold. First $\mathbb{W}$ is a network environment with domain Σ. Second, M is closed, well-typed code at world w_j with certificate s that produces a value of type A, if at all. Finally, the network environment should be well-typed. The type of $\mathbb{W}$; finish has to be the type of the term currently in focus and located at the same world as indicated in the machine state. A network environment $\mathbb{W}; k \lhd \circ\; N$ is well-typed with type $A \supset B$ at world w under Σ, if the argument is well-typed with type A at w, and the network environment $\mathbb{W}; k$ is well-typed with type B at the same world and under the same set of worlds. Note that $A \supset B$ is the type of the hole in the next term layer in k, and shall be completed by applying the term in focus to N. This is reminiscent of the left introduction scheme for implication in the Sequent Calculus presentation of Intuitionistic Propositional Logic. This connection is explored in detail in [Her94,CH00]. The $C.App$ and $C.Box$ schemes may be described in similar terms. Regarding the judgement $\Sigma \vdash \{w' : C{::}\,k; w_s\}\,; \mathrm{return}\; w' : A@w$, in order to verify that the type A at w of the value to be returned to world w' is correct, the context at w' must be checked, at w', to see if its outermost hole is indeed expecting a value of this type.

We now state the promised results. Both are proved by structural induction on the derivation of the judgement $\Sigma \vdash \mathbb{N}$. Together these results imply soundess of the reduction relation w.r.t. the type system: if a machine state is typable under Σ and is not terminal, then a well-typed value shall be attained.

Proposition 1 (Progress). *If $\Sigma \vdash \mathbb{N}$ is derivable and $\mathbb{N}$ is not terminal, then there exists $\mathbb{N}'$ such that $\mathbb{N} \longrightarrow \mathbb{N}'$.*

Proposition 2 (Subject Reduction). *If $\Sigma \vdash \mathbb{N}$ is derivable and $\mathbb{N} \longrightarrow \mathbb{N}'$, then $\Sigma \vdash \mathbb{N}'$ is derivable.*

6 Strong normalization

We prove strong normalization (SN) of machine reduction by translating machine states to terms of the simply typed lambda calculus with unit type ($\lambda^{\mathbf{1},\rightarrow}$). For technical reasons (which we comment on shortly) we shall consider the following modification of the machine reduction semantics of $\lambda^{\mathsf{Cert}}_{\Box}$ obtained by replacing the reduction scheme:

$$(2)\ \mathbb{W}; w : [k \triangleleft \circ\ N, V] \longrightarrow \mathbb{W}; w : [k \triangleleft V\ \circ, N]$$

by the following two new reduction schemes:

$$\begin{array}{ll}(2.1) & \mathbb{W}; w : [k \triangleleft \circ\ N, V] \longrightarrow \mathbb{W}; w : [k \triangleleft V\ \circ, N],\ N \text{ is not a value} \\ (2.2) & \mathbb{W}; w : [k \triangleleft \circ\ V, \lambda a.M] \longrightarrow \mathbb{W}; w : [k, M\{a/V\}]\end{array}$$

These schemes result from refining (2) by inspecting its behavior in any non-terminating reduction sequence. If N happens to be a value, then each (2) step is followed by a (3) step. The juxtaposition of these two steps gives precisely (2.2). The reduction scheme (2.1) is just (2) when N is not a value. It is clear that every non-terminating reduction sequence in the original formulation can be mimicked by a non-terminating reduction sequence in the modified semantics in such a way that each (2) step

- either it is not followed by a (3) step and thus becomes a (2.1) step or
- it is followed by a (3) step and hence (2) followed by (3) become one (2.2) step.

Therefore, it suffices to prove SN of the modified system in order to deduce the same property for our original formulation.

The proof of SN proceeds in two phases (Fig. 6). First we relate machine reduction with a notion of reduction that operates directly on lambda terms via a mapping $F(\cdot)$. Then we relate the latter with reduction in $\lambda^{\mathbf{1},\rightarrow}$ via a mapping $T(\cdot)$. We consider the first phase. The map $F(\cdot)$ flattens out the local context of a machine state in order to produce a term of $\lambda^{\mathsf{Cert}}_{\Box}$ and replaces all worlds by some distinguished world "$\bullet$" whose name is irrelevant. We write $\overline{M}$ for $M\{w_1/\bullet\} \ldots \{w_n/\bullet\}$, the result of replacing all worlds in M with $\bullet$. This function is type preserving (assuming $\bullet$ belongs to Σ), a result which is proved by induction on the pair $\langle |\mathbb{W}|, k\rangle$, where $|\mathbb{W}|$ is the size of $\mathbb{W}$ (i.e. the sum of the length of the context stacks of all worlds in its domain).

Lemma 4. *Let $\mathbb{N}$ be $\mathbb{W}; w : [k, M]$. If $\Sigma \vdash \mathbb{N}$ is derivable and $\Sigma \vdash \bullet$, then there exist A and s such that $\Sigma; \cdot; \cdot \rhd F(\mathbb{N}) : A@\bullet \mid s$ is derivable.*

$$\begin{tikzcd}\begin{array}{c}\text{Machine reduction}\\(\lambda^{\mathsf{Cert}}_{\Box})\end{array} \arrow[r, "F(\cdot)"'] & \begin{array}{c}\text{Lambda reduction}\\(\lambda^{\mathsf{Cert}}_{\Box})\end{array} \arrow[r, "T(\cdot)"'] & \begin{array}{c}\text{Simply typed lambda calculus}\\(\lambda^{\mathbf{1},\rightarrow})\end{array}\end{tikzcd}$$

$$\begin{aligned}
F(\mathbb{W}; w : [\text{finish}, M]) &=_{def} \overline{M}\\
F(\mathbb{W}; w : [k \lhd \circ\ N, M]) &=_{def} F(\mathbb{W}; w : [k, M\,N])\\
F(\mathbb{W}; w : [k \lhd V\ \circ, N]) &=_{def} F(\mathbb{W}; w : [k, V\,N])\\
F(\mathbb{W}; w : [k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N, M]) &=_{def} F(\mathbb{W}; w : [k, \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N])\\
F(\{w : C :: k; w_s\}; w' : [\text{return } w, M]) &=_{def} F(\{w : C; w_s\}; w : [k, M])
\end{aligned}$$

$$\begin{aligned}
T(P) &=_{def} P\\
T(A \supset B) &=_{def} T(A) \supset T(B)\\
T([s]A) &=_{def} \mathbf{1} \supset T(A)
\end{aligned}
\qquad
\begin{aligned}
T(a) &=_{def} a\\
T(v^\bullet) &=_{def} v\ \mathit{unit}\\
T(\lambda a.M) &=_{def} \lambda a.T(M)\\
T(M\,N) &=_{def} T(M)\ T(N)\\
T(\mathit{box}_s\,M) &=_{def} \lambda a.T(M), a \text{ fresh of type } \mathbf{1}\\
T(\mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N) &=_{def} (\lambda v.T(N))\ T(M)\\
T(\mathit{fetch}[w]\,M) &=_{def} (\lambda a.a)\ T(M)
\end{aligned}$$

In order to relate machine reduction in $\lambda^{\mathsf{Cert}}_{\Box}$ with reduction in $\lambda^{\mathbf{1},\rightarrow}$ we introduce *lambda reduction*. These schemes are standard except for the last one which states that *fetch* terms have no computational effect at the level of lambda terms. It should be mentioned that strong lambda reduction reduction is considered (i.e. reduction under all term constructors).

Definition 1 (Lambda reduction for $\lambda^{\mathsf{Cert}}_{\Box}$).

$$\begin{aligned}
(\lambda a.M)\,N &\longrightarrow_{\beta} \quad M\{a/N\}\\
\mathit{unpack}\ \mathit{box}_s\,M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N &\longrightarrow_{\beta_\Box} \quad N\{v^\bullet/M\}\{v^\circ/s\}\\
\mathit{fetch}[w]\,M &\longrightarrow_{\mathit{ftch}} M
\end{aligned}$$

We can now establish that the flattening map is also reduction preserving:

Lemma 5. *If $\mathbb{N} \longrightarrow_{1,2.1,4,7} \mathbb{N}'$, then $F(\mathbb{N}) = F(\mathbb{N}')$.*
If $\mathbb{N} \longrightarrow_{2.2,3,5,6} \mathbb{N}'$, then $F(\mathbb{N}) \longrightarrow_{\beta,\beta_\Box,\mathit{ftch}} F(\mathbb{N}')$.

The second part of the proof consists in relating lambda reduction in $\lambda^{\mathsf{Cert}}_{\Box}$ with reduction in $\lambda^{\mathbf{1},\rightarrow}$. For that we introduce a mapping $T(\cdot)$ (Fig. 6) that associates types and terms in $\lambda^{\mathsf{Cert}}_{\Box}$ with types and terms in $\lambda^{\mathbf{1},\rightarrow}$. Function types are translated to function types and the modal type $[s]A$ is translated to functional types whose domain is the unit type $\mathbf{1}$ and whose codomain is the translation of A. Translations of terms is straightforward given the translation on types; the case for *fetch* guarantees that each $\longrightarrow_{\mathit{ftch}}$ step is mapped to a non-empty step in $\lambda^{\mathbf{1},\rightarrow}$. $T(\cdot)$ over terms is both type preserving and reduction preserving. The first of these is proved by induction over the derivation of $\Sigma; \Delta; \Gamma \rhd M : A@w \mid s$.

Lemma 6. *If $\Sigma; \Delta; \Gamma \rhd M : A@w \mid s$ is derivable in $\lambda^{\mathsf{Cert}}_{\Box}$, then $\Delta', \Gamma' \rhd T(M) : T(A)$ is derivable in $\lambda^{\mathbf{1},\rightarrow}$, where*

1. *Γ' results from replacing each hypothesis $a : A@w$ by $a : T(A)$ and*
2. *Δ' results from replacing each hypothesis $v : A@w$ by $v : \mathbf{1} \supset T(A)$.*

The second is proved by induction on M making use of the fact (Lem. 18) that T commutes with substitution of (the translation of) local variables (i.e. $T(M)\{a/T(N)\} = T(M\{a/N\})$). T does not commute with substitution of (the translation of) validity variables (i.e. $T(M)\{v/T(N)\} \neq T(M\{v/N\})$; take $M = v^\bullet$). However, the following (Lem. 19) does hold and suffices for our purposes: $T(M)\{v/\lambda a.T(N)\} \longrightarrow_\beta^* T(M\{v^\bullet/N\}\{v^\circ s/\})$. The arrow $\longrightarrow_\beta^*$ denotes the reflexive, transitive closure of $\longrightarrow_\beta$ while $\longrightarrow_\beta^+$ (below) denotes its transitive closure.

Lemma 7. *If $M \longrightarrow_{\beta,\beta_\Box,ftch} N$, then $T(M) \longrightarrow_\beta^+ T(N)$.*

Our desired result may be proved by contradiction as follows. Let us assume, for the time being, that $\longrightarrow_{1,2.1,4,7}$ reduction is SN. Suppose, also, that there is an infinite reduction sequence starting from a machine state $\mathbb{N}_1$. From our assumption this sequence must have an infinite number of interspersed $\longrightarrow_{2.2,3,5}$ reduction steps:

$$\mathbb{N}_1 \longrightarrow_{1,2.1,4,7}^* \mathbb{N}_2 \longrightarrow_{2.2,3,5} \mathbb{N}_3 \longrightarrow_{1,2.1,4,7}^* \mathbb{N}_4 \longrightarrow_{2.2,3,5} \mathbb{N}_5 \longrightarrow_{1,2.1,4,7}^* \mathbb{N}_6 \longrightarrow_{2.2,3,5} \cdots$$

Then (Lem. 5) we have the following lambda reduction sequence over typable terms (Lem. 4):

$$F(\mathbb{N}_1) = F(\mathbb{N}_2) \longrightarrow_{\beta,\beta_\Box,ftch} F(\mathbb{N}_3) = F(\mathbb{N}_4) \longrightarrow_{\beta,\beta_\Box,ftch} F(\mathbb{N}_5) = \\ F(\mathbb{N}_6) \longrightarrow_{\beta,\beta_\Box,ftch} \cdots$$

Finally, we arrive at the following infinite reduction sequence (Lem. 7) of typable terms (Lem. 6) in $\lambda^{\mathbf{1},\to}$, thus contradicting SN of $\lambda^{\mathbf{1},\to}$:

$$T(F(\mathbb{N}_1)) = T(F(\mathbb{N}_2)) \longrightarrow_\beta^+ T(F(\mathbb{N}_3)) = T(F(\mathbb{N}_4)) \longrightarrow_\beta^+ T(F(\mathbb{N}_5)) = \\ T(F(\mathbb{N}_6)) \longrightarrow_\beta^+ \cdots$$

In order to complete our proof we now address our claim, namely that $\longrightarrow_{1,2.1,4,7}$ reduction is SN. It is the proof of this result that has motivated the modified reduction semantics presented at the beginning of this section. First a simple yet useful result for proving SN of combinations of binary relations that we have implicitly made use of above.

Lemma 8. *Let $\longrightarrow_1$ and $\longrightarrow_2$ be binary relations over some set X. Suppose*

1. $\longrightarrow_1$ is SN and
2. $\mathcal{M}$ is a mapping from X to some well-founded set such that
   (a) $x \longrightarrow_1 y$ implies $\mathcal{M}(x) = \mathcal{M}(y)$
   (b) $x \longrightarrow_2 y$ implies $\mathcal{M}(x) > \mathcal{M}(y)$

Then $\longrightarrow_1 \cup \longrightarrow_2$ is SN.

Before we use this lemma for our proof of SN of $\longrightarrow_{1,2.1,4,7}$, some definitions are required. The size of a term M, written $|M|$, is defined as the number of variables and constructors in M:

$$\begin{aligned}
|a| &=_{def} 1\\
|v^\bullet| &=_{def} 1\\
|\mathit{box}_s\, M| &=_{def} |M|+1\\
|\lambda a.M| &=_{def} |M|+1\\
|M\, N| &=_{def} |M|+|N|+1\\
|\mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N| &=_{def} |M|+|N|+1\\
|\mathit{fetch}[w]\, M| &=_{def} |M|+1
\end{aligned}$$

Note that $|M\{w'/w\}| = |M|$. The size of a context k, written $|k|$, is defined by taking the sum of the sizes of the terms with holes, where each hole counts as 1:

$$\begin{aligned}
|\text{return } w| &=_{def} 1\\
|\text{finish}| &=_{def} 1\\
|k \lhd l| &=_{def} |k|+|l|\\
\\
|\circ\ N| &=_{def} |N|+1\\
|V\ \circ| &=_{def} |V|+1\\
|\mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N| &=_{def} |N|+1
\end{aligned}$$

We will write $|k, M|$ to abbreviate $|k| + |M|$.

Lemma 9. $\longrightarrow_{1,2.1,4,7}$ *reduction is SN.*

Proof. First we prove SN of schemes (1) and (4). Then we conclude by resorting to Lem. 8, introducing a measure $\mathcal{M}_2$ such that:

1. $\mathbb{N} \longrightarrow_{1,4} \mathbb{N}'$ implies $\mathcal{M}_2(\mathbb{N}) = \mathcal{M}_2(\mathbb{N}')$ and
2. $\mathbb{N} \longrightarrow_{2.1,7} \mathbb{N}'$ implies $\mathcal{M}_2(\mathbb{N}) > \mathcal{M}_2(\mathbb{N}')$.

SN of schemes (1) and (4) follows from noting that the following measure $\mathcal{M}_1$ of machine states over pairs of natural numbers (ordered lexicographically) strictly decreases when schemes (1) and (4) are applied[5]:

$$\mathcal{M}_1(\mathbb{W}; w : [k, M]) =_{def} \langle |\mathbb{W}|, |M| \rangle$$

We are left to verify that the following measure $\mathcal{M}_2$ enjoys the required properties stated above:

$$\mathcal{M}_2(\mathbb{W}; w : [k, M]) =_{def} \langle |\mathbb{W}|, |k, M| - \mathit{len}(k) - m(M) \rangle$$

where $\mathit{len}(k)$ is the length of k and m is the following mapping from closed terms to positive integers:

$$\begin{aligned}
m(V) &=_{def} 0\\
m(M\, N) &=_{def} 1 + m(M)\\
m(\mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N) &=_{def} 1 + m(M)\\
m(\mathit{fetch}[w]\, M) &=_{def} 1
\end{aligned}$$

This measure descreases strictly for both (2.1) and (7), whereas it yields equal numbers for (1) and (4).

[5] It also decreases when (7) is applied. However, it does not decrease when (2) is applied.

- Case (1)

$$\begin{aligned} \mathcal{M}_2(\mathbb{W}; w : [k, M\,N]) &= \langle |\mathbb{W}|, |k, M\,N| - \mathit{len}(k) - m(M\,N)\rangle \\ &= \langle |\mathbb{W}|, |k, M\,N| - \mathit{len}(k) - 1 - m(M)\rangle \\ &= \langle |\mathbb{W}|, |k \triangleleft \circ\ N, M| - \mathit{len}(k) - 1 - m(M)\rangle \\ &= \mathcal{M}_2(\mathbb{W}; w : [k \triangleleft \circ\ N, M]) \end{aligned}$$

- Case (2.1), recall from above that N is not a value. Therefore, it is either an application, an unpack term or a fetch term. Note that for each of these $m(N) > 0$. Therefore, we reason as follows:

$$\begin{aligned} \mathcal{M}_2(\mathbb{W}; w : [k \triangleleft \circ\ N, V]) &= \langle |\mathbb{W}|, |k \triangleleft \circ\ N, V| - \mathit{len}(k) - 1 - m(V)\rangle \\ &= \langle |\mathbb{W}|, |k, N, V| + 1 - \mathit{len}(k) - 1\rangle \\ &> \langle |\mathbb{W}|, |k, V, N| + 1 - \mathit{len}(k) - 1 - m(N)\rangle \\ &= \langle |\mathbb{W}|, |k \triangleleft V\ \circ, N| - \mathit{len}(k) - 1 - m(N)\rangle \\ &= \mathcal{M}_2(\mathbb{W}; w : [k \triangleleft V\ \circ, N]) \end{aligned}$$

- Case (4)

$$\begin{aligned} &\mathcal{M}_2(\mathbb{W}; w : [k, \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N]) \\ &= \langle |\mathbb{W}|, |k, \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N| - \mathit{len}(k) - m(\mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N)\rangle \\ &= \langle |\mathbb{W}|, |k, \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N| - \mathit{len}(k) - 1 - m(M)\rangle \\ &= \langle |\mathbb{W}|, |k \triangleleft \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N, M| - \mathit{len}(k) - 1 - m(M)\rangle \\ &= \mathcal{M}_2(\mathbb{W}; w : [k \triangleleft \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N, M]) \end{aligned}$$

- Case (7). Let $n = |\{w : C{::}k; w_s\}|$.

$$\begin{aligned} &\mathcal{M}_2(\{w : C{::}k; w_s\}; w' : [\text{return } w, V]) \\ &= \langle n, |\text{return } w| + |V| - \mathit{len}(\text{return } w) - m(V)\rangle \\ &= \langle n, 1 + |V| - 1 - m(V)\rangle \\ &= \langle n, |V|\rangle \\ &> \langle n - 1, |k, V| - \mathit{len}(k)|\rangle \\ &= \langle n - 1, |k, V| - \mathit{len}(k) - m(V)|\rangle \\ &= \langle n - 1, |k, V\{w'/w\}| - \mathit{len}(k) - m(V\{w'/w\})|\rangle \\ &= \mathcal{M}_2(\{w : C; w_s\}; w : [k, V]) \end{aligned}$$

We can finally state our desired result, whose proof we have presented above.

Proposition 3. $\longrightarrow$ *is SN.*

7 Related Work

There are many foundational calculi for concurrent and distributed programming. Since the focus of this work is on logically motivated such calculi we comment on related work from this viewpoint. To the best of our knowledge, the extant literature does not address calculi for both mobility/concurrency and code certification in a unified theory. Regarding mobility, however, a number of ideas have been put forward. The closest to this article is the work of Moody [Moo04], that of Murphy et al [VCHP04,VCH05,VCH07] and that of Jia and Walker [JW04]. Moody suggests an operational reading of proofs in an intuitionistic fragment of S4 also based on a judgemental analysis of this logic [DP01a]. It takes a step further in terms of obtaining a practical programming language for mobility in that it addresses effectful computation (references and reference update). Also, the diamond connective is considered. Worlds are deliberately left implicit. The author argues this "encourages the programmer to work locally". Murphy et al also introduce a mobility inspired operational interpretation of a Natural Deduction presentation of propositional modal logic, although S5 is considered in there work (both intuitionistic [VCHP04] and classical [VCH05]). They also introduce explicit reference to worlds in their programming model. Operational semantics in terms of abstract machines is considered [VCHP04,VCH05] and also a big-step semantics on terms [Mur08]. Both necesity and possibility modalities are considered. Finally, they explore a type preserving compiler for a prototype language for client/server applications based on their programming model [VCH07]. Jia and Walker [JW04] also present a term assignment for a hybrid modal logic close to S5. They argue that the hybrid approach gives the programmer a tighter control over code distribution. Finally, Borghuis and Feij [BF00] introduce a calculus of stationary services and mobile values whose type system is based on modal logic. Mobility however may not be internalised as a proposition. For example, $\Box^o(A \supset B)$ is the type of a service located at o that computes values of B given one of type A. None of the cited works incorporate the notion of certificate in their systems.

8 Conclusion

We present a Curry-de Bruijn-Howard analysis of an intuitionistic fragment (ILP) of the Logic of Proofs LP. We start from a Natural Deduction presentation for ILP and associate propositions and proofs of this system to types and terms of a mobile calculus $\lambda^{\mathsf{Cert}}_{\Box}$. The modal type constructor $[s]A$ is interpreted as the type of *mobile units*, expressions composed of a code and certificate component. $\lambda^{\mathsf{Cert}}_{\Box}$ has thus language constructs for both code and certificates. Its type system is a unified theory in which both code and certificate construction are verified. Indeed, when mobile units are constructed from the code of other mobile units, the type system verifies not only that the former is mobile in nature (i.e. depends on no local resources) but also that the certificate for this new mobile unit is correctly assembled from the certificates of the latter.

Although we deal exclusively with the necessity modality, we hasten to mention that it would be quite straightfoward to add inference schemes for a possibility modality, in the line of related literature (cf. Sec. 7). A term of type $\Diamond A$ is generally interpeted to denote a value of a term at a remote location. However, a provability interpretation of this connective in an intuitionistic fragment of LP has first to be investigated. Since LP is based on classical logic $\Diamond$ is ignored altogether. However, in an intuitionistic setting the interpretation of $\Diamond$ in possible world semantics is not as uncontroversial as that of the necessity modality [Sim94, Ch.3]. An explicit logic of provability based on (classical) $\mathsf{S5}$ has been reported [AED99] and could be an appropriate starting point. Nevertheless one could explore this additional modality from a purely programming languages perspective.

Although $\lambda^{\mathsf{Cert}}_{\Box}$ is meant to be concept-of-proof language, it clearly does not provide the features needed to build extensive examples. Two basic additions that should be considered are references (and computation with effects) and recursion.

References

[AB04] Sergei Artëmov and Leo Beklemishev. Provability logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 13, pages 229–403. Kluwer, 2nd edition, 2004.

[AB07] Sergei N. Artëmov and Eduardo Bonelli. The intensional lambda calculus. In Sergei N. Artëmov and Anil Nerode, editors, *LFCS*, volume 4514 of *Lecture Notes in Computer Science*, pages 12–25. Springer, 2007.

[AED99] S. Artemov, E.Kazakov, and D.Shapiro. On logic of knowledge with justifications. Technical Report CFIS 99-12, Cornell University, 1999.

[Art94] Sergei N. Artëmov. Logic of proofs. *Ann. Pure Appl. Logic*, 67(1-3):29–59, 1994.

[Art95] Sergei Artemov. Operational modal logic. Technical Report MSI 95-29, Cornell University, 1995.

[Art01] Sergei Artemov. Unified semantics of modality and λ-terms via proof polynomials. *Algebras, Diagrams and Decisions in Language, Logic and Computation*, pages 89–118, 2001.

[AtC06] C. Areces and B. ten Cate. Hybrid logics. In P. Blackburn, F. Wolter, and J. van Benthem, editors, *Handbook of Modal Logics*. Elsevier, 2006.

[BF00] Tijn Borghuis and Loe M. G. Feijs. A constructive logic for services and information flow in computer networks. *Comput. J.*, 43(4):274–289, 2000.

[CH00] Pierre-Louis Curien and Hugo Herbelin. The duality of computation. In *ICFP*, pages 233–243, 2000.

[DP01a] Rowan Davies and Frank Pfenning. A judgmental reconstruction of modal logic. *Mathematical Structures in Computer Science*, 11:511–540, 2001.

[DP01b] Rowan Davies and Frank Pfenning. A modal analysis of staged computation. *J. ACM*, 48(3):555–604, 2001.

[Her94] Hugo Herbelin. A lambda-calculus structure isomorphic to gentzen-style sequent calculus structure. In Leszek Pacholski and Jerzy Tiuryn, editors, *CSL*, volume 933 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 1994.

[JW04] Limin Jia and David Walker. Modal proofs as distributed programs (extended abstract). In David A. Schmidt, editor, *ESOP*, volume 2986 of *Lecture Notes in Computer Science*, pages 219–233. Springer, 2004.

[Moo04] Jonathan Moody. Logical mobility and locality types. In Sandro Etalle, editor, *LOPSTR*, volume 3573 of *Lecture Notes in Computer Science*, pages 69–84. Springer, 2004.

[Mur08] Tom Murphy, VII. *Modal Types for Mobile Code.* PhD thesis, Carnegie Mellon, January 2008. (draft).

[Sim94] Alex Simpson. *The Proof Theory and Semantics of Intuitionistic Modal Logic.* PhD thesis, University of Edinburgh, 1994.

[TS97] Walid Taha and Tim Sheard. Multi-stage programming. In *ICFP*, page 321, 1997.

[VCH05] Tom Murphy VII, Karl Crary, and Robert Harper. Distributed control flow with classical modal logic. In C.-H. Luke Ong, editor, *CSL*, volume 3634 of *Lecture Notes in Computer Science*, pages 51–69. Springer, 2005.

[VCH07] Tom Murphy VII, Karl Crary, and Robert Harper. Type-safe distributed programming with ml5. In Gilles Barthe and Cédric Fournet, editors, *TGC*, volume 4912 of *Lecture Notes in Computer Science*, pages 108–123. Springer, 2007.

[VCHP04] Tom Murphy VII, Karl Crary, Robert Harper, and Frank Pfenning. A symmetric modal lambda calculus for distributed computing. In *LICS*, pages 286–295. IEEE Computer Society, 2004.

[WLPD98] Philip Wickline, Peter Lee, Frank Pfenning, and Rowan Davies. Modal types as staging specifications for run-time code generation. *ACM Comput. Surv.*, 30(3es):8, 1998.

A Proofs - Safety

In order to prove Subject Reduction we need the following lemma whose proof is by induction on the derivation of $\Sigma; \Delta; \Gamma \rhd M : A@w'' \mid s$.

Lemma 10 (Substitution principle for worlds). *If $\Sigma; \Delta; \Gamma \rhd M : A@w'' \mid s$ and $\Sigma \vdash w'$,then $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd M\{w/w'\} : A@w'\{w/w'\} \mid s$.*

Proof. By induction over the derivation of $\Sigma; \Delta; \Gamma \rhd M : A@w'' \mid s$

- Case 1: $M = a$. Suppose that:

(1.1) $\Sigma; \Delta; \Gamma \rhd a : A@w'' \mid s$

From (1.1) by VarT, $s = a$ and $\Gamma = \Gamma_1, a : A@w'', \Gamma_2$ for some Γ_1 and Γ_2. Therefore, $\Gamma\{w/w'\} = \Gamma_1\{w/w'\}, a : A@w''\{w/w'\}, \Gamma_2\{w/w'\}$. Moreover, $a\{w/w'\} = a$. By rule VarT,

(1.2) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd a\{w/w'\} : A@w''\{w/w'\} \mid s$

- Case 2: $M = v^{\bullet}$. Suppose that:

(2.1) $\Sigma; \Delta; \Gamma \rhd v^{\bullet} : A@w'' \mid s$

From (2.1) and by VarV, $s = v^{\circ}$ y $\Delta = \Delta_1, v : A@w'', \Delta_2$ for some Δ_1 and Δ_2. Then, $\Delta\{w/w'\} = \Delta_1\{w/w'\}, v : A@w''\{w/w'\}, \Delta_2\{w/w'\}$. Moreover, $v^{\bullet}\{w/w'\} = v^{\bullet}$. By rule VarV,

(2.2) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd v^{\bullet}\{w/w'\} : A@w''\{w/w'\} \mid s$

- Case 3: $M = PQ$. Suppose that:

(3.1) $\Sigma; \Delta; \Gamma \rhd PQ : A@w'' \mid s$

From (3.1) and by $\supset E$, $\exists t_1, t_2, B$, such that $s = t_1 \cdot t_2$ and,

(3.2) $\Sigma; \Delta; \Gamma \rhd P : B \supset A@w'' \mid t_1$

(3.3) $\Sigma; \Delta; \Gamma \rhd Q : B@w'' \mid t_2$

From (3.2) y by IH

(3.4) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd P\{w/w'\} : (B \supset A)@w''\{w/w'\} \mid t_1$

From (3.3) and by IH

(3.5) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd Q\{w/w'\} : B@w''\{w/w'\} \mid t_2$

From (3.4) and (3.5) by $\supset E$

(3.6) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd P\{w/w'\}Q\{w/w'\} : A@w''\{w/w'\} \mid s$

From (3.6) and the definition of substitution:

(3.7) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd (PQ)\{w/w'\} : A@w''\{w/w'\} \mid s$

- Case 4: $M = \lambda a.P$. Suppose that:

(4.1) $\Sigma; \Delta; \Gamma \rhd \lambda a.P : A@w'' \mid s$

From (4.1) and by $\supset I$, $\exists B, B', t$ such that $s = \lambda a : B.t$, $A = B \supset B'$ and,

(4.2) $\Sigma; \Delta; \Gamma, a : B@w'' \rhd P : B'@w'' \mid t$

From (4.2) and by IH:

(4.3) $\Sigma; \Delta\{w/w'\}; (\Gamma, a : B@w'')\{w/w'\} \rhd P\{w/w'\} : B'@w''\{w/w'\} \mid t$

Since $(\Gamma, a : B@w'')\{w/w'\} = \Gamma\{w/w'\}, a : B@w''\{w/w'\}$, from (4.3) by rule $\supset I$

(4.4) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd \lambda a.P\{w/w'\} : A@w''\{w/w'\} \mid s$

From (4.4) and the definition of substitution:

(4.5) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd (\lambda a.P)\{w/w'\} : A@w''\{w/w'\} \mid s$

- Case 5: $M = box_t\, P$. Suppose that:

(5.1) $\Sigma; \Delta; \Gamma \rhd \mathit{box}_t\, P : A@w'' \mid s$

From (5.1) and by $\Box I$ $\exists B$ such that $A = [t]B$, $s =!t$ and

(5.2) $\Sigma; \Delta; \cdot \rhd P : B@w'' \mid t$

From (5.2) and by IH

(5.3) $\Sigma; \Delta\{w/w'\}; \cdot \rhd P\{w/w'\} : B@w''\{w/w'\} \mid t$

From (5.3) by rule $\Box I$

(5.4) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd \mathit{box}_t\, P\{w/w'\} : [t]B@w''\{w/w'\} \mid !t$

From (5.4) and the definition of substitution:

(5.5) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd (\mathit{box}_t\, P)\{w/w'\} : [t]B@w''\{w/w'\} \mid !t$

– Case 6: $M = \mathit{fetch}[w_f]\, P$. Suppose that:

(6.1) $\Sigma; \Delta; \Gamma \rhd \mathit{fetch}[w_f]\, P : A@w'' \mid s$

From (6.1) by Fetch $\exists B, t, r$ such that $A = [t]B$, $s = fetch(r)$ and

(6.2) $\Sigma; \Delta; \Gamma \rhd P : [t]B@w_f \mid r$

From (6.2) by IH

(6.3) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd P\{w/w'\} : [t]B@w_f\{w/w'\} \mid r$

From (6.3) by rule Fetch

(6.4) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd \mathit{fetch}[w_f\{w/w'\}]\, P\{w/w'\} : [t]B@w''\{w/w'\} \mid fetch(r)$

From (6.4) and the definition of substitution

(6.5) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd (\mathit{fetch}[w_f]\, P)\{w/w'\} : A@w''\{w/w'\} \mid s$

– Case 7: $M = \mathit{unpack}\ P\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ Q$. Suppose that:

(7.1) $\Sigma; \Delta; \Gamma \rhd \mathit{unpack}\ P\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ Q : B@w'' \mid s$

From (7.1) by rule $\Box E$, $\exists C, t_1, t_2, r$ such that $s = \mathit{letc}\ t_1\ \mathit{be}\ v : A\ \mathit{in}\ t_2$, $B = C\{v^\circ/r\}$ and,

(7.2) $\Sigma; \Delta; \Gamma \rhd P : [r]A@w'' \mid t_1$

(7.3) $\Sigma; \Delta, v : A@w''; \Gamma \rhd Q : C@w'' \mid t_2$

From (7.2) by IH

(7.4) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd P\{w/w'\} : [r]A@w''\{w/w'\} \mid t_1$

(7.5) $\Sigma; (\Delta, v : A@w'')\{w/w'\}; \Gamma\{w/w'\} \rhd Q\{w/w'\} : C@w''\{w/w'\} \mid t_2$

Since $(\Delta, v : A@w'')\{w/w'\} = \Delta\{w/w'\}, v : A@w''\{w/w'\}$, from (7.4) and (7.5) by rule $\Box E$

(7.6) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd \mathit{unpack}\ P\{w/w'\}\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ Q\{w/w'\} : B@w''\{w/w'\} \mid \mathit{letc}\, t_1\, \mathit{be}\, v :$
$A\ \mathit{in}\ t_2$

From (7.6) and the definition of substitution:

(7.7) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd (\mathit{unpack}\ P\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ Q)\{w/w'\} : B@w''\{w/w'\} \mid s$

The proof of Subject Reduction follows:

Proof. By case analysis on the reduction step applied.

– Case 1:

- $\mathbb{N} = \mathbb{W}\,; w : [k, MN]$
- $\mathbb{N}' = \mathbb{W}\,; w : [k \lhd \circ N, M]$

Suppose $\Sigma \vdash \mathbb{W}\,; w : [k, MN]$. Then by *MState*, there exist B, s' such that:

(1.1) $\Sigma; \cdot; \cdot \rhd MN : B@w \mid s'$

(1.2) $\Sigma \vdash \mathbb{W}\,; k : B@w$

From (1.1) and by $\supset E$, there exist s, t, A such that $s' = s.t$ and

(1.3) $\Sigma; \cdot; \cdot \rhd M : A \supset B@w \mid s$

(1.4) $\Sigma; \cdot; \cdot \rhd N : A@w \mid t$

From (1.2) and (1.4), by $C.Abs$:

(1.5) $\Sigma \vdash \mathbb{W}\,; k \lhd \circ N : A \supset B@w$

From (1.3) and (1.5) by *MState*

$$\Sigma \vdash \mathbb{W}\,; w : [k \lhd \circ N, M]$$

– Case 2:

- $\mathbb{N} = \mathbb{W}\,; w : [k \lhd \circ N, V]$
- $\mathbb{N}' = \mathbb{W}\,; w : [k \lhd V \circ, N]$

Suppose $\Sigma \vdash \mathbb{W}\,; w : [k \lhd \circ N, V]$. Then by *MState* there exist A', s' such that:

(2.1) $\Sigma; \cdot; \cdot \rhd V : A'@w \mid s'$

(2.2) $\Sigma \vdash \mathbb{W}\,; k \lhd \circ N : A'@w$

From (2.2) and $C.Abs$ there exist A, B, s such that $A' = A \supset B$ and, moreover,

(2.3) $\Sigma \vdash \mathbb{W}\,; k : B@w$

(2.4) $\Sigma; \cdot; \cdot \rhd N : A@w \mid s$

From (2.1) and (2.3) by $C.App$:

(2.5) $\Sigma \vdash \mathbb{W}\,; k \lhd \circ V : A@w$

From (2.4) and (2.5) by *MState*

$$\Sigma \vdash \mathbb{W}\,; w : [k \lhd \circ V, N]$$

– Case 3:

- $\mathbb{N} = \mathbb{W}\,; w : [k \lhd (\lambda a.M)\circ, V]$
- $\mathbb{N}' = \mathbb{W}\,; w : [k, M\{a/V\}]$

Suppose $\Sigma \vdash \mathbb{W}\,; w : [k \lhd (\lambda a.M)\circ, V]$. Then by *MState* there exist A, s' such that:

(3.1) $\Sigma; \cdot; \cdot \rhd V : A@w \mid s'$

(3.2) $\Sigma \vdash \mathbb{W}\,; k \lhd (\lambda a.M)\circ : A@w$

From (3.2) and $C.App$ there exist B, t such that:

(3.3) $\Sigma \vdash \mathbb{W}\,; k : B@w$

(3.4) $\Sigma; \cdot; \cdot \rhd (\lambda a.M) : A \supset B@w \mid t$

From (3.4) and by $\supset I$ there exists s such that $t = \lambda a : A.s$ and, moreover,

(3.5) $\Sigma; \cdot; a : A@w \rhd M : B@w \mid s$

From (3.1), (3.5) and the substitution principle for truth hypothesis:

(3.6) $\Sigma; \cdot; \cdot \rhd M\{a/V\} : B@w \mid s\{a/s'\}$

From (3.3) and (3.6) by *MState*

$$\Sigma \vdash \mathbb{W}\,; w : [k, M\{a/V\}]$$

– Case 4:

- $\mathbb{N} = \mathbb{W}\,; w : [k, \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N]$
- $\mathbb{N}' = \mathbb{W}\,; w : [k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N, M]$

Suppose $\Sigma \vdash \mathbb{W}\,; w : [k, \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N]$. Then by *MState* there exists B, s' such that:

(4.1) $\Sigma; \cdot; \cdot \rhd \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N : B@w \mid s'$

(4.2) $\Sigma \vdash \mathbb{W}\,; k : B@w$

From (4.1) and $\Box E$ there exist B', r, s, t such that $s' = \mathit{letc}\ s\ \mathit{be}\ v : A\ \mathit{in}\ t$ and $B = B'\{v^\circ/r\}$ and

(4.3) $\Sigma; \cdot; \cdot \rhd M : [r]A@w \mid s$

(4.4) $\Sigma; v : A; \cdot \rhd N : B'@w \mid t$

From (4.2), (4.4) and by $C.Box$:

(4.5) $\Sigma \vdash \mathbb{W}\,; k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N : [r]A@w$

From (4.3) and (4.5) by MState

(4.6) $\Sigma \vdash \mathbb{W}\,; w : [k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N, M]$

– Case 5:
- $\mathbb{N} = \mathbb{W}\,; w : [k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N, \mathit{box}_s\, M]$
- $\mathbb{N}' = \mathbb{W}\,; w : [k, N\{v^\circ/s\}\{v^\bullet/M\}]$

Suppose $\Sigma \vdash \mathbb{W}\,; w : [k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N, \mathit{box}_s\, M]$. Then by MState there exist A', s' such that:

(5.1) $\Sigma; \cdot; \cdot \rhd \mathit{box}_s\, M : A'@w \mid s'$

(5.2) $\Sigma \vdash \mathbb{W}\,; k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N : A'@w$

From (5.1) and by $\Box I$ there exist s, A such that $A' = [s]A$ and $s' =!s$:

(5.3) $\Sigma; \cdot; \cdot \rhd \mathit{box}_s\, M : [s]A@w \mid !s$

From (5.2) and by $C.Box$ there exist B, t such that:

(5.4) $\Sigma \vdash \mathbb{W}\,; k : B@w$

(5.5) $\Sigma; v : A; \cdot \rhd N : B\{v^\circ/s\}@w \mid t$

From (5.3) and by $\Box I$:

(5.6) $\Sigma; \cdot; \cdot \rhd M : A@\omega \mid s$

From (5.5) and (5.6) by the substitution principle por validity hypothesis:

(5.7) $\Sigma; \cdot; \cdot \rhd N\{v^\circ/s\}\{v^\bullet/M\} : \{v^\circ/s\}\{v^\circ/s\}@w \mid t\{v^\circ/s\}$.

Since $B\{v^\circ/s\}\{v^\circ/s\} = B$, from (5.4) and (5.6) by MState we deduce:

$$\Sigma \vdash \mathbb{W}\,; w : [k, N\{v^\circ/s\}\{v^\bullet/M\}]$$

– Case 6:
- $\mathbb{N} = \{w : C; w_s\}\,; w : [k, \mathit{fetch}[w']\, M]$
- $\mathbb{N}' = \{w : C{::}\,k; \mathrm{w}_s\}\,; w' : [\mathrm{return}\ w, M]$

Suppose $\Sigma \vdash \{w : C; w_s\}\,; w : [k, \mathit{fetch}[w']\, M]$. Then by MState there exist A', s' such that:

(6.1) $\Sigma \vdash \{w : C; w_s\}\,; k : A'@w$

(6.2) $\Sigma; \cdot; \cdot \rhd \mathit{fetch}[w']\, M : A'@w \mid s'$

From (6.2) and by Fetch there exist s, A such that $A' = [s]A$ and moreover:

(6.3) $\Sigma; \cdot; \cdot \rhd M : [s]A@w' \mid s'$

From (6.1) and by $C.Return$:

(6.4) $\Sigma \vdash \{w : C{::}\,k; \mathrm{w}_s\}\,; \mathrm{return}\ w : [s]A@w'$

From (6.3) and (6.4) by MState

$$\Sigma \vdash \{w : C{::}\,k; \mathrm{w}_s\}\,; w' : [\mathrm{return}\ w, M]$$

– Case 7:
- $\mathbb{N} = \{w : C{::}\,k; \mathrm{w}_s\}\,; w' : [\mathrm{return}\ w, V]$
- $\mathbb{N}' = \{w : C; \mathrm{w}_s\}\,; w : [k, V\{w'/w\}]$

Suppose $\Sigma \vdash \{w : C{::}\,k; \mathrm{w}_s\}\,; w' : [\mathrm{return}\ w, V]$. Then by MState, there exist A', s' such that:

(7.1) $\varSigma \vdash \{w : C{:}{:}\, k; \mathrm{w}_s\}\,; \mathrm{return}\ w : A'@w'$
(7.2) $\varSigma; \cdot; \cdot \rhd V : A'@w' \mid s'$
From (7.1) and by *C.Return*:
(7.3) $\varSigma \vdash \{w : C; \mathrm{w}_s\}\,; k : A'@w$
From (7.2) and by the substitution principle for worlds:
(7.4) $\varSigma; \cdot; \cdot \rhd V\{w'/w\} : A'@w \mid s'$
From (7.3) and (7.4) by *MState*

$$\varSigma \vdash \{w : C; \mathrm{w}_s\}\,; w : [k, V\{w'/w\}]$$

The proof of Progress follows.

Proof. By cases on k and M.

If $\varSigma \vdash \mathbb{N}$, then there exist A, s such that:

(a) $\varSigma; \cdot; \cdot \rhd M : A@w \mid s$
(b) $\varSigma \vdash \mathbb{W}\,; k : A@w$

From (a) $M \neq a, v^\bullet$ since $\varGamma, \varDelta$ are empty. Thus we consider the remaining possibilities for M and k.

- Case 1: M is a value $V = box_t\, P$ or $V = \lambda x.P$.
  - Subcase 1.1: $k =$ finish. $\mathbb{N}$ is terminal and hence the result holds.
  - Subcase 1.2: $k = k' \lhd \circ\ N$. By the machine reduction scheme (2), $\mathbb{N} \to \mathbb{W}\,; w : [k' \lhd V \circ, N]$.
  - Subcase 1.3: $k = k' \lhd V' \circ$. By the typing scheme *C.App* there exist B, t' such that $\varSigma; \cdot; \cdot \rhd V' : A \supset B@w \mid t'$. Therefore, from $\supset I$, $V' = \lambda b.N$. Finally, from the reduction scheme (3), $\mathbb{N} \to \mathbb{W}\,; w : [k', N\{a/V\}]$.
  - Subcase 1.4: $k = k' \lhd unpack\ \circ\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N$. By the typing scheme *C.Box* there exist t', A' such that $A = [t']A'$. Therefore $V = box_t\, P$. Finally, by the reduction scheme (5), $\mathbb{N} \to \mathbb{W}\,; w : [k', N\{v^\circ/s\}\{v^\bullet/M\}]$.
  - Subcase 1.5: $k =$ return w'. By the typing scheme *C.Return* $\mathbb{W} = \{w' : C :: k'; w_s\}$. Therefore, from the reduction scheme (7) $\mathbb{N} \to \{w' : C; w_s\}\,; w' : [k', V\{w'/w\}]$.
- Case 2: $M = PQ$. By the reduction scheme (1), $\mathbb{N} \to \mathbb{W}\,; w : [k \lhd \circ\ Q, P]$.
- Case 3: $M = unpack\ P\ to\ \langle v^\bullet, v^\circ\rangle\ in\ Q$. By the reduction schem (5), $\mathbb{N} \to \mathbb{W}\,; w : [k \lhd unpack\ \circ\ to\ \langle v^\bullet, v^\circ\rangle\ in\ Q, P]$.
- Case 4: $M = fetch[w']\, P$. By the reduction scheme (6), $\mathbb{N} \to \{w : C :: k; w_s\}\,; w' :$ [return w, P].

B Proofs - Strong Normalization

B.1 From Machine Reduction in $\lambda^{\mathsf{Cert}}_{\Box}$ to Lambda Reduction in $\lambda^{\mathsf{Cert}}_{\Box}$

Lemma 11. *If $\varSigma; \varDelta; \varGamma \rhd M : A@w \mid s$ and $\varSigma \vdash \bullet$, then $\varSigma; \overline{\varDelta}; \overline{\varGamma} \rhd \overline{M} : A@\bullet \mid s$.*

Proof. Corollary of the substitution principle for worlds.

The following three results relate overlining and substitution. The last two are required in order to prove that overlining preserves reduction (see below) while the first one is required in lemma 16. The first states that overlining cancels any world substitution. The second that overlining commutes with term variable substitution. Finally, overlining also commutes with mobile code/certificate substitution. All results are proved by straightforward induction on M.

Lemma 12. $\overline{M\{w'/w\}} = \overline{M}$.

Lemma 13. $\overline{M}\{a/\overline{N}\} = \overline{M\{a/N\}}$.

Lemma 14. $\overline{N}\{v^\bullet/\overline{M}\}\{v^\circ/s\} = \overline{N\{v^\bullet/M\}\{v^\circ/s\}}$.

Overlining also preserves lambda reduction.

Lemma 15. *If* $M \longrightarrow_{\beta,\beta_\Box,ftch} M'$*, then* $\overline{M} \longrightarrow_{\beta,\beta_\Box,ftch} \overline{M'}$*.*

Proof. By induction on M. The base cases are trivial since no reduction steps may originate from a or $v^\bullet$. We illustrate the cases where reduction takes place at the root of M.

- Case $\longrightarrow_\beta$. Suppose $(\lambda a.M)\,N \longrightarrow_\beta M\{a/N\}$. We reason as follows:

$$\begin{array}{rcl} \overline{(\lambda a.M)\,N} & = & \overline{\lambda a.M}\,\overline{N} \\ & = & (\lambda a.\overline{M})\,\overline{N} \\ & \longrightarrow_\beta & \overline{M}\{a/\overline{N}\} \\ & = & \overline{M\{a/N\}} \quad \text{(Lemma 13)} \end{array}$$

- Case $\longrightarrow_{\beta_\Box}$. Suppose $unpack\ box_s\,M\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N \longrightarrow_{\beta_\Box} N\{v^\bullet/M\}\{v^\circ/s\}$. We reason as follows:

$$\begin{array}{rll} & \overline{unpack\ box_s\,M\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N} & \\ = & unpack\ \overline{box_s\,M}\ to\ \langle v^\bullet, v^\circ\rangle\ in\ \overline{N} & \\ = & unpack\ box_s\,\overline{M}\ to\ \langle v^\bullet, v^\circ\rangle\ in\ \overline{N} & \\ \longrightarrow_\beta & \overline{N}\{v^\bullet/\overline{M}\}\{v^\circ/s\} & \\ = & \overline{N\{v^\bullet/M\}\{v^\circ/s\}} & \text{(Lemma 14)} \end{array}$$

- Case $\longrightarrow_{ftch}$. Suppose $fetch[w]\,M \longrightarrow_{ftcs} M$. We reason as follows:

$$\overline{fetch[w]\,M} = fetch[\bullet]\,\overline{M} \longrightarrow_\beta \overline{M}$$

Lemma 16. *Let* $\mathbb{N}$ *be* $\mathbb{W}\,;w : [k, M]$*. If* $\Sigma \vdash \mathbb{N}$ *and* $\overline{M} = \overline{M'}$*, then* $F(\mathbb{W}\,;w : [k, M]) = F(\mathbb{W}\,;w : [k, M'])$*.*

In particular, since by lemma 12 $\overline{M\{w'/w\}} = \overline{M}$, $F(\mathbb{W}\,;w : [k, M\{w'/w\}]) = F(\mathbb{W}\,;w : [k, M])$.

Proof. By induction over $\langle|\mathbb{W}|, k\rangle$.

- Case $\langle|\mathbb{W}|, \text{finish}\rangle$. $\mathbb{N} = \mathbb{W}\,;w : [\text{finish}, M]$.
  $F(\mathbb{W}\,;w : [\text{finish}, M]) = \overline{M} = \overline{M'} = F(\mathbb{W}\,;w : [\text{finish}, M'])$.

- Case $\langle|\mathbb{W}|, k \lhd \circ N\rangle$. $\mathbb{N} = \mathbb{W}\,; w : [k \lhd \circ N, M]$.
  $F(\mathbb{W}\,; w : [k \lhd \circ N, M]) = F(\mathbb{W}\,; w : [k, M\,N])$. If $\Sigma \vdash \mathbb{W}\,; w : [k \lhd \circ N, M]$, then by *MState* there exists C, s such that:
  (1) $\Sigma; \cdot; \cdot \rhd M : C@w \,|\, s$
  (2) $\Sigma \vdash \mathbb{W}\,; k \lhd \circ N : C@w$
  From (2) by rule $C.Abs$ there exists A, B, t such that $C = A \supset B$ and
  (3) $\Sigma; \cdot; \cdot \rhd N : A@w \,|\, t$
  (4) $\Sigma \vdash \mathbb{W}\,; k : B@w$
  From (1) and (3) by $\supset E$:
  (5) $\Sigma; \cdot; \cdot \rhd M\,N : B@w \,|\, s \cdot t$
  From (4) and (5) by *MState*, $\Sigma \vdash \mathbb{W}\,; w : [k, M\,N]$. Since $\overline{M\,N} = \overline{M}\,\overline{N} = \overline{M'}\,\overline{N} = \overline{M'\,N}$, then by IH, $F(\mathbb{W}\,; w : [k, M\,N]) = F(\mathbb{W}\,; w : [k, M'\,N]) = F(\mathbb{W}\,; w : [k \lhd \circ N, M'])$
- Case $\langle|\mathbb{W}|, k \lhd V \circ\rangle$. $\mathbb{N} = \mathbb{W}\,; w : [k \lhd V \circ, N]$.
  $F(\mathbb{W}\,; w : [k \lhd V \circ, N]) = F(\mathbb{W}\,; w : [k, V\,N])$. If $\Sigma \vdash \mathbb{W}\,; w : [k \lhd V \circ, N]$, then by *MState* there exists A and s such that:
  (1) $\Sigma; \cdot; \cdot \rhd N : A@w \,|\, s$
  (2) $\Sigma \vdash \mathbb{W}\,; k \lhd V \circ : A@w$
  From (2) by $C.App$ there exist B and t such that:
  (3) $\Sigma; \cdot; \cdot \rhd V : A \supset B@w \,|\, t$
  (4) $\Sigma \vdash \mathbb{W}\,; k : B@w$
  From (1) and (3) by $\supset E$:
  (5) $\Sigma; \cdot; \cdot \rhd V\,N : B@w \,|\, s \cdot t$
  From (4) and (5) by *MState*, $\Sigma \vdash \mathbb{W}\,; w : [k, V\,N]$. Since $\overline{V\,N} = \overline{V}\,\overline{N} = \overline{V}\,\overline{N'} = \overline{V\,N'}$. Then, by IH, $F(\mathbb{W}\,; w : [k, V\,N]) = F(\mathbb{W}\,; w : [k, V\,N']) = F(\mathbb{W}\,; w : [k \lhd V \circ, N'])$
- Case $\langle|\mathbb{W}|, k \lhd \mathit{unpack} \circ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ M\rangle$. $\mathbb{N} = \mathbb{W}\,; w : [k \lhd \mathit{unpack} \circ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N, M]$.
  $F(\mathbb{W}\,; w : [k \lhd \mathit{unpack} \circ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N, M]) = F(\mathbb{W}\,; w : [k, \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N])$.
  If $\Sigma \vdash \mathbb{W}\,; w : [k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N, M]$, then by *MState* there exists D, s such that:
  (1) $\Sigma; \cdot; \cdot \rhd M : D@w \,|\, s$
  (2) $\Sigma \vdash \mathbb{W}\,; k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N : D@w$
  From (2) by $C.Box$ there exists A, C, t, r such $D = [r]A$ and
  (3) $\Sigma; v : A; \cdot \rhd N : C@w \,|\, t$
  (4) $\Sigma \vdash \mathbb{W}\,; k : C\{v^\circ/r\}@w$
  From (1) and (3) by $\Box E$:
  (5) $\Sigma; \Delta; \Gamma \rhd \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N : C\{v^\circ/r\}@w \,|\, \mathit{letc}\ s\ \mathit{be}\ v : A\ \mathit{in}\ t$
  From (4) and (5) by *MState*, $\Sigma \vdash \mathbb{W}\,; w : [k, \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N]$. Since $\overline{\mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N} = \mathit{unpack}\ \overline{M}\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ \overline{N} = \mathit{unpack}\ \overline{M'}\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ \overline{N} = \overline{\mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N'}$, then by IH, $F(\mathbb{W}\,; w : [k, \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N]) = F(\mathbb{W}\,; w : [k, \mathit{unpack}\ M'\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N]) = F(\mathbb{W}\,; w : [k \lhd \mathit{unpack} \circ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N, M'])$
- Case $\langle|\mathbb{W}|, \text{return } w\rangle$. Since $k = \text{return } w$ and $\mathbb{N}$ is typable, it follows that $\mathbb{W} = \{w : C\!::\!k_1; w_s\}$ for some C, k_1, w_s. Therefore, $\mathbb{N} = \{w : C\!::\!k_1; w_s\}\,; w' : [\text{return } w, M]$.
  $F(\{w : C\!::\!k_1; w_s\}\,; w' : [\text{return } w, M]) = F(\{w : C; w_s\}\,; w : [k_1, M])$.
  Since $\Sigma \vdash \{w : C\!::\!k_1; w_s\}\,; w' : [\text{return } w, M]$, by *MState* there exist A and s such that:

(1) $\Sigma; \cdot; \cdot \rhd M : A@w' \,|\, s$
(2) $\Sigma \vdash \{w : C :: k_1; w_s\}\,; \mathrm{return}\ w : A@w'$
From (2) by $C.Return$:
(3) $\Sigma \vdash \{w : C; w_s\}\,; k_1 : A@w$
From (1) and by the world substitution lemma:
(4) $\Sigma; \cdot; \cdot \rhd M\{w'/w\} : A@w \,|\, s$
From (3) and (4) by $MState$ $\Sigma \vdash \underline{\{w : C; w_s\}\,; w} : [k_1, M\{w'/w\}]$ Since $|\{w : C; w_s\}| < |\{w : C :: k_1; w_s\}|$ and $\overline{M} = \overline{M\{w'/w\}} = \overline{M'}$ then, by HI, $F(\{w : C; w_s\}\,; w : [k_1, M]) = F(\{w : C; w_s\}\,; w : [k_1, M\{w'/w\}]) = F(\{w : C; w_s\}\,; w : [k_1, M']) = F(\{w : C :: k_1; w_s\}\,; w' : [\mathrm{return}\ w, M'])$.

We now have the necessary results to prove lemma 4.

Proof. By induction over $\langle |\mathbb{W}|, k\rangle$.

- Case $\langle |\mathbb{W}|, finish\rangle$. $\mathbb{N} = \mathbb{W}\,; w : [finish, M]$.
  $F(\mathbb{W}\,; w : [finish, M]) = \overline{M}$. If $\Sigma \vdash \mathbb{W}\,; w : [finish, M]$, then by $MState$ there exists A and s such that $\Sigma; \cdot; \cdot \rhd M : A@w \,|\, s$ is derivable. Then, by Lemma 11 $\Sigma; \cdot; \cdot \rhd \overline{M} : A@\bullet \,|\, s$
- Case $\langle |\mathbb{W}|, k \lhd \circ N\rangle$. $\mathbb{N} = \mathbb{W}\,; w : [k \lhd \circ N, M]$.
  $F(\mathbb{W}\,; w : [k \lhd \circ N, M]) = F(\mathbb{W}\,; w : [k, M\, N])$. If $\Sigma \vdash \mathbb{W}\,; w : [k \lhd \circ N, M]$, then by $MState$ there exists C, s such that:
  (1) $\Sigma; \cdot; \cdot \rhd M : C@w \,|\, s$
  (2) $\Sigma \vdash \mathbb{W}\,; k \lhd \circ N : C@w$
  From (2) by rule $C.Abs$ there exists A, B, t such that $C = A \supset B$ and
  (3) $\Sigma; \cdot; \cdot \rhd N : A@w \,|\, t$
  (4) $\Sigma \vdash \mathbb{W}\,; k : B@w$
  From (1) and (3) by $\supset E$:
  (5) $\Sigma; \cdot; \cdot \rhd M\, N : B@w \,|\, s \cdot t$
  From (4) and (5) by $MState$, $\Sigma \vdash \mathbb{W}\,; w : [k, M\, N]$. Then by IH, there exists A', s' such that $\Sigma; \cdot; \cdot \rhd F(\mathbb{W}\,; w : [k, M\, N]) : A'@\bullet \,|\, s'$.
- Case $\langle |\mathbb{W}|, k \lhd V \circ\rangle$. $\mathbb{N} = \mathbb{W}\,; w : [k \lhd V \circ, N]$.
  $F(\mathbb{W}\,; w : [k \lhd V \circ, N]) = F(\mathbb{W}\,; w : [k, V\, N])$. If $\Sigma \vdash \mathbb{W}\,; w : [k \lhd V \circ, N]$, then by $MState$ there exists A and s such that:
  (1) $\Sigma; \cdot; \cdot \rhd N : A@w \,|\, s$
  (2) $\Sigma \vdash \mathbb{W}\,; k \lhd V \circ : A@w$
  From (2) by $C.App$ there exist B and t such that:
  (3) $\Sigma; \cdot; \cdot \rhd V : A \supset B@w \,|\, t$
  (4) $\Sigma \vdash \mathbb{W}\,; k : B@w$
  From (1) and (3) by $\supset E$:
  (5) $\Sigma; \cdot; \cdot \rhd V\, N : B@w \,|\, s \cdot t$
  From (4) and (5) by $MState$, $\Sigma \vdash \mathbb{W}\,; w : [k, V\, N]$. Then by IH, there exists A', s' such that $\Sigma; \cdot; \cdot \rhd F(\mathbb{W}\,; w : [k, V\, N]) : A'@\bullet \,|\, s'$.
- Case $\langle |\mathbb{W}|, k \lhd unpack \circ to \langle v^\bullet, v^\circ\rangle\ in\ M\rangle$. $\mathbb{N} = \mathbb{W}\,; w : [k \lhd unpack \circ to \langle v^\bullet, v^\circ\rangle\ in\ N, M]$.
  $F(\mathbb{W}\,; w : [k \lhd unpack \circ to \langle v^\bullet, v^\circ\rangle\ in\ N, M]) = F(\mathbb{W}\,; w : [k, unpack\ M\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N])$.
  If $\Sigma \vdash \mathbb{W}\,; w : [k \lhd unpack\ \circ\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N, M]$, then by $MState$ there exists D, s such that:

(1) $\Sigma; \cdot; \cdot \rhd M : D@w \,|\, s$
(2) $\Sigma \vdash \mathbb{W}\,; k \lhd \mathit{unpack} \;\circ\; \mathit{to}\ \langle v^{\bullet}, v^{\circ}\rangle\ \mathit{in}\ N : D@w$
From (2) by $C.Box$ there exists A, C, t, r such $D = [r]A$ and
(3) $\Sigma; v : A; \cdot \rhd N : C@w \,|\, t$
(4) $\Sigma \vdash \mathbb{W}\,; k : C\{v^{\circ}/r\}@w$
From (1) and (3) by $\Box E$:
(5) $\Sigma; \Delta; \Gamma \rhd \mathit{unpack}\ M\ \mathit{to}\ \langle v^{\bullet}, v^{\circ}\rangle\ \mathit{in}\ N : C\{v^{\circ}/r\}@w \,|\, \mathit{letc}\ s\ \mathit{be}\ v : A\ \mathit{in}\ t$
From (4) and (5) by $MState$, $\Sigma \vdash \mathbb{W}\,; w : [k, \mathit{unpack}\ M\ \mathit{to}\ \langle v^{\bullet}, v^{\circ}\rangle\ \mathit{in}\ N]$. Then by IH, there exists A', s' such that $\Sigma; \cdot; \cdot \rhd F(\mathbb{W}\,; w : [k, \mathit{unpack}\ M\ \mathit{to}\ \langle v^{\bullet}, v^{\circ}\rangle\ \mathit{in}\ N]) : A'@\bullet \,|\, s'$.

- Case $\langle |\mathbb{W}|, \mathrm{return}\ w\rangle$. Since $k = \mathrm{return}\ w$ and $\mathbb{N}$ is typable, it follows that $\mathbb{W} = \{w : C{::}\,k_1; w_s\}$ for some C, k_1, w_s. Therefore, $\mathbb{N} = \{w : C{::}\,k_1; w_s\}\,; w' : [\mathrm{return}\ w, M]$.
$F(\{w : C{::}\,k_1; w_s\}\,; w' : [\mathrm{return}\ w, M]) = F(\{w : C; w_s\}\,; w : [k_1, M])$.
Since $\Sigma \vdash \{w : C{::}\,k_1; w_s\}\,; w' : [\mathrm{return}\ w, M]$, by $MState$ there exist A and s such that:
(1) $\Sigma; \cdot; \cdot \rhd M : A@w' \,|\, s$
(2) $\Sigma \vdash \{w : C{::}\,k_1; w_s\}\,; \mathrm{return}\ w : A@w'$
From (2) by $C.Return$:
(3) $\Sigma \vdash \{w : C; w_s\}\,; k_1 : A@w$
From (1) and by the world substitution lemma:
(4) $\Sigma; \cdot; \cdot \rhd M\{w'/w\} : A@w \,|\, s$
From (3) and (4) by $MState$ $\Sigma \vdash \{w : C; w_s\}\,; w : [k_1, M\{w'/w\}]$. Since $|\{w : C; w_s\}| < |\{w : C{::}\,k_1; w_s\}|$ by IH there exists A' s' such that
(5) $\Sigma; \cdot; \cdot \rhd F(\{w : C; w_s\}\,; w : [k_1, M\{w'/w\}]) : A'@\bullet \,|\, s'$.
From (5) and by lemma 16
(6) $\Sigma; \cdot; \cdot \rhd F(\{w : C; w_s\}\,; w : [k_1, M]) : A'@\bullet \,|\, s'$.

The proof of Lem. 5 requires the following auxiliary result which states that $F(\cdot)$ preserves lambda reduction, under any network environment and context.

Lemma 17. *If $M \longrightarrow_{\beta,\beta_{\Box},ftch} M'$ and $\Sigma \vdash \mathbb{W}\,; w : [k, M]$ is derivable, then $F(\Sigma \vdash \mathbb{W}\,; w : [k, M]) \longrightarrow_{\beta,\beta_{\Box},ftch} F(\Sigma \vdash \mathbb{W}\,; w : [k, M'])$.*

Proof. By induction over $\langle |\mathbb{W}|, k\rangle$.

- Case $\langle |\mathbb{W}|, \mathit{finish}\rangle$.
If $M \longrightarrow_{\beta,\beta_{\Box},ftch} M'$ then by lemma 15, $\overline{M} \longrightarrow_{\beta,\beta_{\Box},ftch} \overline{M'}$. Therefore
$$F(\Sigma \vdash \mathbb{W}\,; w : [\mathit{finish}, M]) = \overline{M} \longrightarrow_{\beta,\beta_{\Box},ftch} \overline{M'} = F(\Sigma \vdash \mathbb{W}\,; w : [\mathit{finish}, M'])$$
- Case $\langle |\mathbb{W}|, k \lhd \circ\, N\rangle$.
$$F(\Sigma \vdash \mathbb{W}\,; w : [k \lhd \circ\, N, M]) = F(\Sigma \vdash \mathbb{W}\,; w : [k, M\, N])$$
If $M \longrightarrow_{\beta,\beta_{\Box},ftch} M'$, then $M\,N \longrightarrow_{\beta,\beta_{\Box},ftch} M'\,N$. So by IH:
$$F(\Sigma \vdash \mathbb{W}\,; w : [k, M\,N]) \longrightarrow_{\beta,\beta_{\Box},ftch} F(\Sigma \vdash \mathbb{W}\,; w : [k, M'\,N]) = F(\Sigma \vdash \mathbb{W}\,; w : [k \lhd \circ\, N, M'])$$

- Case $\langle |\mathbb{W}|, k \lhd V \circ \rangle$ is similar to the previous case.
- Case $\langle |\mathbb{W}|, k \lhd \mathit{unpack} \ \circ \ \mathit{to} \ \langle v^\bullet, v^\circ \rangle \ \mathit{in} \ N \rangle$.

$$\begin{gathered} F(\Sigma \vdash \mathbb{W}\,; w : [k \lhd \mathit{unpack} \ \circ \ \mathit{to} \ \langle v^\bullet, v^\circ \rangle \ \mathit{in} \ N, M]) = \\ F(\Sigma \vdash \mathbb{W}\,; w : [k, \mathit{unpack} \ M \ \mathit{to} \ \langle v^\bullet, v^\circ \rangle \ \mathit{in} \ N]) \end{gathered}$$

If $M \longrightarrow_{\beta,\beta_\Box,\mathit{ftch}} M'$, then $\mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N \longrightarrow_{\beta,\beta_\Box,\mathit{ftch}} \mathit{unpack}\ M'\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N$. So by IH:

$$\begin{gathered} F(\Sigma \vdash \mathbb{W}\,; w : [k, \mathit{unpack} \ M \ \mathit{to} \ \langle v^\bullet, v^\circ \rangle \ \mathit{in} \ N]) \longrightarrow_{\beta,\beta_\Box,\mathit{ftch}} \\ F(\Sigma \vdash \mathbb{W}\,; w : [k, \mathit{unpack} \ M' \ \mathit{to} \ \langle v^\bullet, v^\circ \rangle \ \mathit{in} \ N]) = \\ F(\Sigma \vdash \mathbb{W}\,; w : [k \lhd \mathit{unpack} \ \circ \ \mathit{to} \ \langle v^\bullet, v^\circ \rangle \ \mathit{in} \ N, M']) \end{gathered}$$

- Case $\langle |\mathbb{W}|, \mathrm{return}\, w \rangle$. Since the machine state is typed, it follows that $\mathbb{W} = \{w : C{:}{:}\,k_1; w_s\}$, for some C, k_1 and w_s.

$$F(\Sigma \vdash \{w : C{:}{:}\,k_1; w_s\}\,; w' : [\mathrm{return}\, w, M]) = F(\Sigma \vdash \{w : C; w_s\}\,; w : [k_1, M])$$

Since $|\{w : C; w_s\}| < |\{w : C{:}{:}\,k_1; w_s\}|$, then by IH:

$$\begin{gathered} F(\Sigma \vdash \{w : C; w_s\}\,; w : [k_1, M]) \longrightarrow_{\beta,\beta_\Box,\mathit{ftch}} F(\Sigma \vdash \{w : C; w_s\}\,; w : [k_1, M']) = \\ F(\Sigma \vdash \{w : C{:}{:}\,k_1; w_s\}\,; w' : [\mathrm{return}\, w, M']) \end{gathered}$$

Proof of Lem. 5.

Proof. We address both items by case analysis on the reduction rule.

- Case (1): $\mathbb{W}; w : [k, MN] \longrightarrow \mathbb{W}; w : [k \lhd \circ\ N, M]$

$$F(\mathbb{W}; w : [k, MN]) = F(\mathbb{W}; w : [k \lhd \circ\ N, M])$$

- Case (2.1): $\mathbb{W}; w : [k \lhd \circ\ N, V] \longrightarrow \mathbb{W}; w : [k \lhd V\ \circ, N]$, N is not a value

$$F(\mathbb{W}; w : [k \lhd \circ\ N, V]) = F(\mathbb{W}; w : [k, V\ N]) = F(\mathbb{W}; w : [k \lhd V\ \circ, N])$$

- Case (4): $\mathbb{W}; w : [k, \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N] \longrightarrow \mathbb{W}; w : [k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N, M]$

$$\begin{gathered} F(\mathbb{W}; w : [k, \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N]) = F(\mathbb{W}; w : \\ [k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N, M]) \end{gathered}$$

- Case (7): $\{w : C{:}{:}\,k; w_s\}; w' : [\mathrm{return}\ w, V] \longrightarrow \{w : C; w_s\}; w : [k, V\{w'/w\}]$

$$\begin{aligned} &F(\{w : C{:}{:}\,k; w_s\}; w' : [\mathrm{return}\ w, V]) = \\ &F(\{w : C; w_s\}; w : [k, V]) \qquad\qquad = (\text{lemma } 16) \\ &F(\{w : C; w_s\}; w : [k, V\{w'/w\}]) \end{aligned}$$

- Case (2.2): $\mathbb{W}; w : [k \lhd \circ\ V, \lambda a.M] \longrightarrow \mathbb{W}; w : [k, M\{a/V\}]$

$$F(\mathbb{W}; w : [k \lhd \circ\ V, \lambda a.M]) = F(\mathbb{W}; w : [k, (\lambda a.M)\, V])$$

Since $(\lambda a.M)\, V \longrightarrow_\beta M\{a/V\}$, then by Lemma 17:

$$F(\mathbb{W}; w : [k, (\lambda a.M)\, V]) \longrightarrow_{\beta,\beta_\Box,\mathit{ftch}} F(\mathbb{W}; w : [k, M\{a/V\}])$$

- Case (3) is developed similarly to that of (2.2).
- Case (5): $\mathbb{W}; w : [k \triangleleft \mathit{unpack} \circ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N, \mathit{box}_s\, M] \longrightarrow \mathbb{W}; w : [k, N\{v^\circ/s\}\{v^\bullet/M\}]$

$$F(\mathbb{W}; w : [k \triangleleft \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N, \mathit{box}_s\, M]) = F(\mathbb{W}; w : [k, \mathit{unpack}\ \mathit{box}_s\, M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N])$$

Since $\mathit{unpack}\ \mathit{box}_s\, M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N \longrightarrow_{\beta_\Box} N\{v^\circ/s\}\{v^\bullet/M\}$, then by Lemma 17:

$$F(\mathbb{W}; w : [k, \mathit{unpack}\ \mathit{box}_s\, M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N]) \longrightarrow_{\beta,\beta_\Box,\mathit{ftch}} F(\mathbb{W}; w : [k, N\{v^\circ/s\}\{v^\bullet/M\}])$$

- Case (6): $\{w : C; w_s\}; w : [k, \mathit{fetch}[w']\, M] \longrightarrow \{w : C{::}\,k; w_s\}; w' : [\text{return } w, M]$

$$F(\{w : C{::}\,k; w_s\}; w' : [\text{return } w, M]) = F(\{w : C; w_s\}; w : [k, M])$$

Since $\mathit{fetch}[w']\, M \longrightarrow_{\mathit{ftch}} M$, then by Lemma 17:

$$F(\{w : C; w_s\}; w : [k, \mathit{fetch}[w']\, M]) \longrightarrow_{\beta,\beta_\Box,\mathit{ftch}} F(\{w : C; w_s\}; w : [k, M])$$

B.2 From lambda reduction in $\lambda^{\mathsf{Cert}}_{\Box}$ to reduction in $\lambda^{\mathbf{1},\to}$

Proof of Lem. 6.

Proof. By induction on the derivation of $\Sigma; \Delta; \Gamma \rhd M : A@w \,|\, s$. The base cases are straightforward. We include some sample inductive cases.

- Case $\supset E$.

$$\frac{\Sigma; \Delta; \Gamma \rhd M : A \supset B@w \,|\, s \quad \Sigma; \Delta; \Gamma \rhd N : A@w \,|\, t}{\Sigma; \Delta; \Gamma \rhd M\, N : B@w \,|\, s \cdot t} \supset E$$

Then by the IH we know $\Delta', \Gamma' \rhd T(M) : T(A \supset B)$ is derivable and that $T(A \supset B) = T(A) \supset T(B)$. Therefore:

$$\frac{\Delta', \Gamma' \rhd T(M) : T(A) \supset T(B) \quad \Delta', \Gamma' \rhd T(N) : T(A)}{\Delta', \Gamma' \rhd T(M)\ T(N) : T(B)} \supset E$$

- Case $\Box I$.

$$\frac{\Sigma; \Delta; \cdot \rhd M : A@w \,|\, s}{\Sigma; \Delta; \Gamma \rhd \mathit{box}_s\, M : [s]A@w \,|\, !s} \Box I$$

From the IH $\Delta' \rhd T(M) : T(A)$ is derivable. From weakening in $\lambda^{\mathbf{1},\to}$ we obtain $\Delta', a : \mathbf{1} \rhd T(M) : T(A)$. Thus $\Delta' \rhd \lambda a.T(M) : \mathbf{1} \supset T(A)$ is derivable.
- Case Fetch.

$$\frac{\Sigma; \Delta; \Gamma \rhd M : [s]A@w' \,|\, t \quad \Sigma \vdash w}{\Sigma; \Delta; \Gamma \rhd \mathit{fetch}[w']\, M : [s]A@w \,|\, \mathit{fetch}(t)} \text{Fetch}$$

Then

$$\dfrac{\Delta', \Gamma' \rhd \lambda a.a : (\mathbf{1} \supset T(A)) \supset (\mathbf{1} \supset T(A)) \quad \Delta', \Gamma' \rhd T(M) : \mathbf{1} \supset T(A)}{\Delta', \Gamma' \rhd (\lambda a.a)\, T(M) : \mathbf{1} \supset T(A)} \supset E$$

- Case $\Box E$.

$$\dfrac{\Sigma; \Delta; \Gamma \rhd M : [r]A@w \,|\, s \quad \Sigma; \Delta, v : A@w; \Gamma \rhd N : C@w \,|\, t}{\Sigma; \Delta; \Gamma \rhd \mathit{unpack}\ M\ \mathit{to}\ \langle v^{\bullet}, v^{\circ} \rangle\ \mathit{in}\ N : C\{v^{\circ}/r\}@w \,|\, \mathit{letc}\ s\ \mathit{be}\ v : A\ \mathit{in}\ t} \Box E$$

  From the IH we have
  - $\Delta', \Gamma' \rhd T(M) : \mathbf{1} \supset T(A)$
  - $\Delta', v : \mathbf{1} \supset T(A), \Gamma' \rhd T(N) : T(C)$

  We conclude as follows, noting that $T(C) = T(C\{v^{\circ}/r\})$

$$\dfrac{\dfrac{\Delta', v : \mathbf{1} \supset T(A), \Gamma' \rhd T(N) : T(C)}{\Delta', \Gamma' \rhd \lambda v.T(N) : (\mathbf{1} \supset T(A)) \supset T(C)} \quad \Delta', \Gamma' \rhd T(M) : \mathbf{1} \supset T(A)}{\Delta', \Gamma' \rhd (\lambda v.T(N))\, T(M) : T(C)} \supset E$$

Finally, we address Lem. 7. Two auxiliary results are required first (Lem. 18 and 19 below).

Lemma 18. *$T(M)\{a/T(N)\} = T(M\{a/N\})$.*

Proof. By induction over M. We illustrate two sample cases:

- Case $v^{\bullet}$.

$$\begin{aligned} T(v^{\bullet})\{a/T(N)\} &= (v\, \mathit{unit})\{a/T(N)\} \\ &= v\, \mathit{unit} \\ &= T(v^{\bullet}) \\ &= T(v^{\bullet}\{a/N\}) \end{aligned}$$

- Case $M\, M'$.

$$\begin{aligned} T(M\, M')\{a/T(N)\} &= (T(M)\, T(M'))\{a/T(N)\} \\ &= T(M)\{a/T(N)\}\, T(M')\{a/T(N)\} \\ &= T(M\{a/N\}\, M'\{a/N\}) && \text{(IH)} \\ &= T((M\, M')\{a/N\}) \end{aligned}$$

Lemma 19. *$T(M)\{v/\lambda a.T(N)\} \longrightarrow_{\beta}^{*} T(M\{v^{\bullet}/N\}\{v^{\circ}/s\})$.*

Proof. By induction over M.

- Case b.

$$T(b)\{v/\lambda a.T(N)\} = b\{v/\lambda a.T(N)\} = b = T(b) = T(b\{v^{\bullet}/N\}\{v^{\circ}/s\})$$

- Case $v^{\bullet}$.

$$T(v^{\bullet})\{v/\lambda a.T(N)\} = (v\, \mathit{unit})\{v/\lambda a.T(N)\} = (\lambda a.T(N))\, \mathit{unit} \longrightarrow_{\beta} T(N) = T(v^{\bullet}\{v^{\bullet}/N\}\{v^{\circ}/s\})$$

- Case $u^{\bullet} \neq v^{\bullet}$.

$$T(u^\bullet)\{v/\lambda a.T(N)\} = (u\,unit)\{v/\lambda a.T(N)\} = u\,unit = T(u^\bullet) = \\ T(u^\bullet\{v^\bullet/N\}\{v^\circ/s\})$$

- Case $M\,M'$.

$$T(M\,M')\{v/\lambda a.T(N)\} = (T(M)\,T(M'))\{v/\lambda a.T(N)\} = \\ T(M)\{v/\lambda a.T(N)\}\,T(M')\{v/\lambda a.T(N)\}$$

By IH,

$$T(M)\{v/\lambda a.T(N)\} \longrightarrow^*_\beta T(M\{v^\bullet/N\}\{v^\circ/s\}) \\ T(M')\{v/\lambda a.T(N)\} \longrightarrow^*_\beta T(M'\{v^\bullet/N\}\{v^\circ/s\})$$

Then,

$$T(M)\{v/\lambda a.T(N)\}\,T(M')\{v/\lambda a.T(N)\} \longrightarrow^*_\beta \\ T(M\{v^\bullet/N\}\{v^\circ/s\})\,T(M'\{v^\bullet/N\}\{v^\circ/s\}) = \\ T(M\{v^\bullet/N\}\{v^\circ/s\}\,M'\{v^\bullet/N\}\{v^\circ/s\}) = T((M\,M')\{v^\bullet/N\}\{v^\circ/s\})$$

- Case $\lambda b.M$ (note that $a \neq b$ since a is fresh).

$$T(\lambda b.M)\{v/\lambda a.T(N)\} = (\lambda b.T(M))\{v/\lambda a.T(N)\} = \lambda b.T(M)\{v/\lambda a.T(N)\}$$

By IH,

$$T(M)\{v/\lambda a.T(N)\} \longrightarrow^*_\beta T(M\{v^\bullet/N\}\{v^\circ/s\})$$

Then,

$$\lambda b.T(M\{v^\bullet/N\}\{v^\circ/s\}) \longrightarrow^*_\beta \lambda b.T(M\{v^\bullet/N\}\{v^\circ/s\}) = \\ T(\lambda b.M\{v^\bullet/N\}\{v^\circ/s\}) = T((\lambda b.M)\{v^\bullet/N\}\{v^\circ/s\})$$

- Case $box_t\,M$.

$$T(box_t\,M)\{v/\lambda a.T(N)\} = (\lambda b.T(M))\{v/\lambda a.T(N)\} = \lambda b.T(M)\{v/\lambda a.T(N)\}$$

By IH,

$$T(M)\{v/\lambda a.T(N)\} \longrightarrow^*_\beta T(M\{v^\bullet/N\}\{v^\circ/s\})$$

Then,

$$\lambda b.T(M)\{v/\lambda a.T(N)\} \longrightarrow^*_\beta \lambda b.T(M\{v^\bullet/N\}\{v^\circ/s\}) = \\ T(box_{t\{s/v^\circ\}}\,M\{v^\bullet/N\}\{v^\circ/s\}) = T((box_t\,M)\{v^\bullet/N\}\{v^\circ/s\})$$

- Case $unpack\ M\ to\ \langle u^\bullet, u^\circ\rangle\ in\ M'$.

$$T(unpack\ M\ to\ \langle u^\bullet, u^\circ\rangle\ in\ M')\{v/\lambda a.T(N)\} = \\ ((\lambda u.T(M'))\,T(M))\{v/\lambda a.T(N)\} = \\ ((\lambda u.T(M')\{v/\lambda a.T(N)\})\,T(M)\{v/\lambda a.T(N)\})$$

By IH,

$$T(M)\{v/\lambda a.T(N)\} \longrightarrow^*_\beta T(M\{v^\bullet/N\}\{v^\circ/s\}) \\ T(M')\{v/\lambda a.T(N)\} \longrightarrow^*_\beta T(M'\{v^\bullet/N\}\{v^\circ/s\})$$

Then,

$$\begin{gathered}((\lambda u.T(M')\{v/\lambda a.T(N)\})\,\mathrm{T}(\mathrm{M})\{v/\lambda a.T(N)\}) \longrightarrow^*_\beta \\ ((\lambda u.T(M'\{v^\bullet/N\}\{v^\circ/s\}))\,\mathrm{T}(\mathrm{M}\{v^\bullet/N\}\{v^\circ/s\}))= \\ \mathrm{T}(\mathit{unpack}\ M\{v^\bullet/N\}\{v^\circ/s\}\ \mathit{to}\ \langle u^\bullet, u^\circ\rangle\ \mathit{in}\ M'\{v^\bullet/N\}\{v^\circ/s\})= \\ \mathrm{T}((\mathit{unpack}\ M\ \mathit{to}\ \langle u^\bullet, u^\circ\rangle\ \mathit{in}\ M')\{v^\bullet/N\}\{v^\circ/s\})\end{gathered}$$

- Case $\mathit{fetch}[w]\,M$.

  $$\begin{gathered}T(\mathit{fetch}[w]\,M)\{v/\lambda a.T(N)\} = ((\lambda b.b)\,T(M))\{v/\lambda a.T(N)\} = \\ (\lambda b.b)\,(T(M)\{v/\lambda a.T(N)\})\end{gathered}$$

  By IH,

  $$T(M)\{v/\lambda a.T(N)\} \longrightarrow^*_\beta T(M\{v^\bullet/N\}\{v^\circ/s\})$$

  Then,

  $$\begin{gathered}(\lambda b.b)\,(T(M)\{v/\lambda a.T(N)\}) \longrightarrow^*_\beta (\lambda b.b)\,(T(M\{v^\bullet/N\}\{v^\circ/s\})) = \\ T(\mathit{fetch}[w]\,M\{v^\bullet/N\}\{v^\circ/s\}) = T((\mathit{fetch}[w]\,M)\{v^\bullet/N\}\{v^\circ/s\})\end{gathered}$$

  The proof of Lem. 7 follows.

Proof. By induction on M. The base cases are trivial since no reduction steps may originate from a or $v^\bullet$. The inductive cases follow from the fact that reduction under all constructors in $\lambda^{\mathbf{1},\rightarrow}$ is considered. We illustrate the cases where reduction takes place at the root of M.

- Case $\longrightarrow_\beta$. Suppose $(\lambda a.M)\,N \longrightarrow_\beta M\{a/N\}$. We reason as follows:

  $$\begin{array}{rll} T((\lambda a.M)\,N) & = & T(\lambda a.M)\,T(N) \quad (\text{Def. of } T(\cdot)) \\ & = & (\lambda a.T(M))\,T(N) \quad (\text{Def. of } T(\cdot)) \\ & \longrightarrow_\beta & T(M)\{a/T(N)\} \\ & = & T(M\{a/N\}) \qquad\quad (\text{Lemma 18}) \end{array}$$

- Case $\longrightarrow_{\beta_\Box}$. Suppose $\mathit{unpack}\ \mathit{box}_s\,M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N \longrightarrow_{\beta_\Box} N\{v^\bullet/M\}\{v^\circ/s\}$. We reason as follows:

  $$\begin{array}{rll} & T(\mathit{unpack}\ \mathit{box}_s\,M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N) & \\ = & (\lambda v.T(N))\,T(\mathit{box}_s\,M) & (\text{Def. of } T(\cdot)) \\ = & (\lambda v.T(N))\,\lambda a.T(M) & (\text{Def. of } T(\cdot)) \\ \longrightarrow_\beta & T(N)\{v/\lambda a.T(M)\} & \\ \longrightarrow^*_\beta & T(N\{v^\bullet/M\}\{v^\circ/s\}) & (\text{Lemma 19}) \end{array}$$

- Case $\longrightarrow_{\mathit{ftch}}$. Suppose $\mathit{fetch}[w]\,M \longrightarrow_{\mathit{ftcs}} M$. We reason as follows:

  $$\begin{array}{rll} T(\mathit{fetch}[w]\,M) & = & (\lambda a.a)\,T(M) \quad (\text{Def. of } T(\cdot)) \\ & \longrightarrow_\beta & T(M) \end{array}$$