Book Title
Book Editors
IOS Press, 2003

Machine Extraction of the Normalization-by-Evaluation Algorithm from a Normalization Proof

Dominik Schlenker, Helmut Schwichtenberg
Department of Mathematics
Ludwig-Maximilians-Universität München

Abstract. In this paper a formal proof of normalization of the simply typed λ-calculus is introduced, which was implemented in the proof system Minlog. The proof is a version of Tait's normalization proof, which relies on the notion of "Strong Computability". From the implementation a program is extracted which represents a variant of the well-known normalization-by-evaluation algorithm.

Keywords. program extraction, Tait's normalization proof, simply typed λ-calculus, de Bruijn, Minlog, proof system, normalization-by-evaluation

1. Introduction

We formalize a version of Tait's normalization proof for the simply typed λ-calculus and machine-extract a program which turns out to implement the well-known normalization-by evaluation algorithm. The fact that this algorithm is the computational content of Tait's proof has first been observed by Berger in [1]. However – as is to be expected – the formalization turned out to be not at all a trivial matter. This is partially due to the fact that the proof relied on some "Axioms", which were not proven. In fact even though the "Axioms" were chosen to be rather simple looking, the formalization revealed that also the proofs of the "Axioms" were not at all trivial. This was particularly the case for "Axiom 1". For this reason we will give the complete core of the proof of "Axiom 1" in this paper. Moreover, we intend to describe some of the choices which have simplified the task considerably. Clearly, a full formalization of the proof *is* necessary for machine extraction of a program.

[2] describes such formalizations carried out for the proof assistants *Minlog*, *Coq* and *Isabelle/HOL* which all have suitable program extraction machineries built in. This provided a useful occasion to test these machineries in a non-trivial setting and to compare the three proof assistants. The crucial questions, on which the formalizations diverge, are how to (1) model the simply typed λ-calculus, (2) represent in the given logical system the notions introduced in the proof (i.p. Tait's strong computability predicates), (3) optimize program extraction in order to get as close as possible to normalization-by evaluation.

Here we restrict ourselves to a formalization in Minlog., which has been done essentially in full detail.[1] The proof can be run interactively in the Minlog system, which is available at *http://www.minlog-system.de/*. The proof code itself can be found within the Minlog system in the directory "minlog/examples/tait/diplomarbeit_schlenker/". In this paper names in parenthesis next to definitions, lemmas and theorems usually correspond to the names in the implementation in Minlog. Further details about the formalization can be found in [3].

The paper consists of five sections. Section 2 introduces the simply typed λ-calculus in the "de Bruijn" notation, for which normalization is proven. Moreover it shows a way in this notation how to handle substitution formally. This section acts also as a brief introduction to Minlog by showing some examples of the proof code. Section 3 is about the universal information system Ω and the so-called *administrative functions*. Universal information systems come into play as we redefine the predicate SC over a new predicate SC**r** in form of $\mathrm{SC}^{\rho}_{\vec{\rho}}(r) := \exists a\, \mathrm{SC}\mathbf{r}^{\rho}_{\vec{\rho}}(a,r)$. The universal information system Ω then is an appropriate domain for the new variable type a. Section 4 presents the normalization proof. It is divided into a general part, the main part and the part with the proof of the "Axioms". Section 5 finally gives the extracted program and shows that it represents the well-known algorithm called Normalization by Evaluation.

All about the program extraction, which is used in this paper and on which the proof system Minlog is based upon, can be found in detail in [4] in chapter 4.

Related work

Related normalization algorithms have been machine-extracted from formal proofs in the type-theoretic proof checker ALF (the precursor of AGDA) [5,6]. However, there the main ingredients of normalization-by-evaluation, the evaluation function and its inverse $\downarrow$, show up explicitely in the proofs already, while in our proof these components of the algorithm are implicit in the logical argument and are made explicit by the extraction only. There exist also formalized normalization proofs for systems such as the Calculus of Constructions [7], System F [8], the typed λ-calculus with co-products [9] and λ-calculi with various weak reduction strategies [10] for which however no program extraction by machine has been carried out.

2. The Simply Typed λ-Calculus

In this section we define the simply typed λ-calculus, for which we will prove weak normalization later on. We introduce the λ-calculus in the "de Bruijn" notation, which allows us to handle substitution formally. This will be needed particularly in the proof of "Axiom 6". The "de Bruijn" notation goes without names for bound variables, hence no alpha conversion is needed here.

To convey a first idea how the formalization is implemented in the proof system Minlog, we give some simple examples of the proof code in this section. More details about Minlog can be found at its webpage [11], particularly its reference manual [12] and the work on the theoretical background of the proof system [4].

[1] The main part is implemented except for some trivial statements. The part of the "Axioms" is implemented except for some non-critical auxiliaries.

2.1. Definition of the Simply Typed λ-Calculus

For the simply typed λ-calculus (with a single ground type) the set of types is defined inductively as follows:

Definition 1. $\mathbb{T} ::= \{\iota\} \mid \mathbb{T} \Rightarrow \mathbb{T}$

In Minlog such an inductive data structure is defined via a so-called *free algebra* with the corresponding command `add-alg`:

```
(add-alg "type"
         '("Iota" "type")
         '("Arrow" "type=>type=>type"))
```

An algebra, which is here called `type`, is built from *constructors*, in this case from `Iota` and `Arrow`. While `Iota` is a nullary constructor, `Arrow` takes two elements from the algebra `type` and constructs another element of `type`. Hence for example `Iota` is an element of the algebra as well as `Arrow Iota Iota` and `Arrow (Arrow Iota Iota) Iota`.

Elements of $\mathbb{T}$ are denoted by ρ, σ and τ. In Minlog this corresponds to adding new variables with the command `add-var-name`:

```
(add-var-name "rho" "sig" "tau" (py "type"))
```

`(py "type")` determines the algebra which the variables refer to. `py` indicates to the parser that the name that follows is an algebra.

For the set of λ-terms we use the “de Bruijn” notation:

Definition 2. $\Lambda ::= \mathbb{N} \mid \Lambda\Lambda \mid \lambda^{\mathbb{T}}(\Lambda)$

Again this data structure is implemented in Minlog as an algebra:

```
(add-alg "term"
         '("Var" "nat=>term")
         '("App" "term=>term=>term")
         '("Abs" "type=>term=>term"))
```

This time the algebra is called `term`. and the constructors are `Var`, `App` and `Abs`. Here `nat` is the predefined algebra in Minlog representing the natural numbers.

Elements of Λ are denoted by r, s and t. As before this corresponds to adding variables in Minlog:

```
(add-var-name "r" "s" "t" (py "term"))
```

This time the variables denote elements of the algebra `term`.

We use the usual abbreviation for applications $r\vec{s} := rs_1 \ldots s_n$ where $\vec{s}$ is the list with the elements $s_1, \ldots, s_n$. As we use the *Church type system* for our calculus we need contexts for the free variables. In the “de Bruijn” notation contexts can be defined as lists with types as elements:

Definition 3. $\mathbb{L}_{\mathbb{T}} ::= \emptyset \mid \mathbb{T} \star \mathbb{L}_{\mathbb{T}}$

Elements of $\mathbb{L}_{\mathbb{T}}$ are denoted by $\vec{\rho}, \vec{\sigma}$ and $\vec{\tau}$. We will use $\star$ in general for lists, both as a list constructor and also for appending lists.

In Minlog, lists with elements from an arbitrary algebra form a predefined algebra, which can be generated by the keyword `list`. Hence in order to add contexts to the system it suffices to write:

```
(add-var-name "rhos" "sigs" "taus" (py "list type"))
```

Next we define the recursive function *Typ*, which determines the type of a λ-term:

Definition 4. Typ : $\mathbb{L}_{\mathbb{T}} \to \Lambda \to \mathbb{T}$

$$\begin{array}{ll}
\text{Typ}_{\emptyset}(n) & := \iota \\
\text{Typ}_{(\rho\star\vec{\rho})}(0) & := \rho \\
\text{Typ}_{(\rho\star\vec{\rho})}((n+1)) & := \text{Typ}_{\vec{\rho}}(n) \\
\text{Typ}_{\vec{\rho}}((rs)) & := \text{Valtyp}(\text{Typ}_{\vec{\rho}}(r)) \\
\text{Typ}_{\vec{\rho}}((\lambda^{\rho} r)) & := \rho \Rightarrow (\text{Typ}_{(\rho\star\vec{\rho})}(r))
\end{array}$$

where Valtyp$(\rho \Rightarrow \sigma) := \sigma$ and Valtyp$(\iota) := \iota$. The irregular cases Valtyp$(\iota) := \iota$ and Typ$_{\emptyset}(n) := \iota$ make the function total. They are absorbed by the check of correctness expressed by the function *Cor* later on.

In Minlog such a recursive function is implemented by a so-called *program constant* for which *computation rules* can be defined:

```
(add-program-constant "Typ" (py "list type=>term=>type") 1)

(add-computation-rule (pt "Typ(Nil type)(Var n)") (pt "Iota"))
(add-computation-rule (pt "Typ(rho::rhos)(Var 0)") (pt "rho"))
(add-computation-rule (pt "Typ(rho::rhos)(Var(Succ n))")
                      (pt "Typ rhos(Var n)"))
(add-computation-rule (pt "Typ rhos(r s)")
                      (pt "Valtyp(Typ rhos r)"))
(add-computation-rule (pt "Typ rhos(Abs rho r)")
                      (pt "rho to Typ(rho::rhos)r"))
```

The program constant `Typ` takes as argument an element of the algebra `list type` and one of the algebra `term` and returns an element of the algebra `type`. The number `1` in the first line indicates that the program constant is supposed to be total. `pt` indicates to the parser, similarly as `py` before, that a general *term* follows, that is an element of an algebra.

The next recursive function *Cor* checks, whether a term is well-typed:

Definition 5. Cor : $\mathbb{L}_{\mathbb{T}} \to \Lambda \to \mathbb{B}$

$$\begin{array}{ll}
\text{Cor}_{\vec{\rho}}(n) & := n < \mathcal{L}(\vec{\rho}) \\
\text{Cor}_{\vec{\rho}}(rs) & := \text{Cor}_{\vec{\rho}}(r) \text{ and } \text{Cor}_{\vec{\rho}}(s) \text{ and} \\
 & \quad\ \text{Typ}_{\vec{\rho}}(r) = (\text{Typ}_{\vec{\rho}}(s) \Rightarrow \text{Valtyp}(\text{Typ}_{\vec{\rho}}(r))) \\
\text{Cor}_{\vec{\rho}}(\lambda^{\rho} r) & := \text{Cor}_{(\rho\star\vec{\rho})}(r)
\end{array}$$

where $\mathbb{B}$ is the boolean algebra and $\mathcal{L}$ the length of a list. Finally we define the type judgement:

Definition 6. $\vdash: \mathbb{L}_{\mathbb{T}} \to \Lambda \to \mathbb{T} \to \mathbb{B}$, $(\vec{\rho} \vdash r : \rho) := \mathrm{Cor}_{\vec{\rho}}(r)$ and $\mathrm{Typ}_{\vec{\rho}}(r) = \rho$

In Minlog both, Cor and $\vdash$, are defined as program constants analogously to Typ.

2.2. *Substitution in "de Bruijn" Style*

2.2.1. *Definition of Substitution*

We will define substitution formally, based on the work of Joachimski [13]. Therefore we first introduce a "lift", which is used to adapt variables in a λ-term during substitution in accordance to the "de Bruijn" notation.

Definition 7. $\uparrow: \Lambda \to \mathbb{N} \to \mathbb{N} \to \Lambda$

$$\begin{array}{ll}
n \uparrow^k_m & := \begin{cases} n+k & : n \geq m \\ n & : otherwise \end{cases} \\
(rs) \uparrow^k_m & := (r \uparrow^k_m)(s \uparrow^k_m) \\
(\lambda^\rho r) \uparrow^k_m & := \lambda^\rho (r \uparrow^k_{m+1})
\end{array}$$

We use the following abbreviation $r \uparrow^k := r \uparrow^k_0$. As this lift is used in the example below, we give also its implemented form here:

```
(add-program-constant "Lift" (py "term=>nat=>nat=>term") 1)

(add-computation-rule (pt "Lift (Var n) m k")
                      (pt "[if (n<m) (Var n) (Var(n+k))]"))
(add-computation-rule (pt "Lift (App r s) m k")
                      (pt "App (Lift r m k)(Lift s m k)"))
(add-computation-rule (pt "Lift(Abs rho r)m k")
                      (pt "Abs rho (Lift r(m+1)k)"))
```

Next we define substitution lists, which are lists of λ-terms ending with a lift:

Definition 8. $\Theta ::= \ \uparrow^{\mathbb{N}} \ \mid \Lambda \star \Theta$

An element of Θ is denoted by θ. Besides a lift for λ-terms we also need a lift for substitution lists:

Definition 9. $\Uparrow: \Theta \to \mathbb{N} \to \Theta$

$$\begin{array}{ll}
\uparrow^m \Uparrow^n & := \uparrow^{m+n} \\
(r \star \theta) \Uparrow^n & := r \uparrow^n \ \star (\theta \Uparrow^n)
\end{array}$$

Analogously we also use $\Uparrow$ to lift common lists of λ-terms. We use the following abbreviation $\theta \Uparrow \ := \theta \Uparrow^1$. With the above definitions we are now able to define substitution recursively as follows:

Definition 10 (Substitution). $.[.] : \Lambda \to \Theta \to \Lambda$

$$\begin{array}{ll} 0[r \star \theta] & := r \\ n[r \star \theta] & := (n-1)[\theta] \\ n[\uparrow^k] & := n + k \\ (rs)[\theta] & := (r[\theta])(s[\theta]) \\ (\lambda^\rho r)[\theta] & := \lambda^\rho(r[0 \star \theta \Uparrow]) \end{array}$$

2.2.2. *Properties of Substitution*

The substitution introduced above has some nice properties. Particularly it allows to formally handle composition of substitutions. This will be needed for "Axiom 6" in the Normalization Proof. Detailled proofs can be found in [3].

First we define inductively the composition of two substitution lists:

Definition 11. $\circ : \Theta \to \Theta \to \Theta$

$$\begin{array}{ll} \uparrow^0 \circ\theta & := \theta \\ \uparrow^{n+1} \circ \uparrow^m & := \uparrow^{n+1+m} \\ \uparrow^{n+1} \circ(r \star \theta) & := \uparrow^n \circ\theta \\ (r \star \theta) \circ \theta' & := r[\theta'] \star (\theta \circ \theta') \end{array}$$

Remark 12. $\uparrow^0$ *is a neutral element on both sides with respect to composition, i.e.* $\uparrow^0 \circ\theta = \theta = \theta\circ \uparrow^0$.

We define $\nearrow^k_n := n \star \ \ldots \ \star (n+k)$ and the abbreviation $\nearrow^k := \nearrow^k_0$.

Lemma 13. *The following properties hold:*

1. $r \uparrow^n_m \uparrow^{n'}_{m+n} = r \uparrow^{n+n'}_m$
2. $r \uparrow^n_m \uparrow^{n'}_m = r \uparrow^{n+n'}_m$
3. $\theta \Uparrow^m \Uparrow^n = \theta \Uparrow^{m+n}$
4. $r \uparrow^n_m = r[\nearrow^m \ \star \ \uparrow^{n+m}]$
5. $\theta\circ \uparrow^n = \theta \Uparrow^n$
6. $l[\theta \Uparrow^m] \uparrow^n_m = l[\theta \Uparrow^{m+n}]$ *with* $l \in \mathbb{N}$
7. $r[\nearrow^m \ \star \theta \Uparrow^m] \uparrow^n_m = r[\nearrow^m \ \star \theta \Uparrow^{m+n}]$
8. $\theta \circ (\theta' \Uparrow^n) = (\theta \circ \theta') \Uparrow^n$
9. $r \uparrow^n_m [\nearrow^m \ \star (\vec{r}^n \star \theta)] = r[\nearrow^m \ \star \theta]$
10. $\theta \Uparrow^n \circ(\vec{r}^n \star \theta') = \theta \circ \theta'$

Proof: Properties 1, 2, 4, 7 and 9 hold by induction on r. Properties 3, 5, 6, 8 and 10 hold by induction on θ. □

As an example for the way a proof is written in Minlog, we sketch the implementation of property 1 from above. First the *goal* has to be given by the command `set-goal`:

```
(set-goal (pf "all r,m,n,n1.
  Lift (Lift r m n) (m+n) n1 = Lift r m (n+n1)"))
```

`pf` indicates to the parser that a *formula* follows, analogously as before `py` and `pt`. The expression `all` represents the quantifier $\forall$. The variable `n1` corresponds to n'. The program constant `Lift` was defined above.

Next comes the actual proof in form of a list of commands:

```
(ind)
(ng)
(assume "k" "m" "n" "n1")
(cases (pt "k<m"))
...
```

The command `(ind)` causes Minlog to perform an induction on the first variable of the current goal, in this case on the variable `r`. The induction is followed by several other commands, where for example `(cases (pt "k<m"))` initiates a corresponding case distinction. Note that Minlog is based on *backward chaining*, that is the order of reasoning is reverse to the order in a usual proof tree.

When the prove is finished, it can be saved for example under the name "LiftTwice":

```
(save "LiftTwice")
```

Through the command `(use "LiftTwice")` one can then use this lemma later on in another proof.

With the above properties one can finally show the main property:

Theorem 14. *(SubSub)* $\boxed{r[\theta]\,[\theta'] = r[\theta \circ \theta']}$

Proof: Induction on r, case distinction on the form of θ and induction on θ'. $\square$

3. Semantics

In order to extract a complete program we will redefine the original definition for Strong Computability "SC" in the Normalization Proof and replace it by SC $:= \exists a.$ SC$\mathbf{r}$. For the variable a we need an appropriate domain. In this subsection we introduce the *universal information system*, which will be such a domain. It is based on the notion of *information systems* developed by Dana Scott.

Besides the universal information system we introduce the so-called *administrative functions* Mod and Hat and for the ground type ι the functions ModIota and HatIota. The administrative functions will appear as realizers of the SC-Lemmas.

Information systems [14,15] can be seen as an intuitive approach to deal constructively with ideal, infinite objects in function spaces, by means of their finite approximations. One works with atomic units of information, called *tokens*, and a notion of *consistency* for finite sets of tokens. Finally there is an *entailment* relation, between consistent finite sets of tokens and single tokens. The *ideals* (or *objects*) of an information system are defined to be the consistent and deductively closed sets of tokens; we write $|\mathbf{A}|$ for the set of ideals of $\mathbf{A}$.

Let $\mathbf{N}$ and $\mathbf{\Lambda}$ be information systems representing the natural numbers $\mathbb{N}$ and the terms Λ, and $\mathbf{C}_{\rho\Rightarrow\sigma} := \mathbf{C}_\rho \to \mathbf{C}_\sigma$. The definition of $\mathbf{C}_\iota$ as $\mathbf{N} \to \mathbf{\Lambda}$, instead of just $\mathbf{\Lambda}$ is due to the specific requirements of the normalization proof. This will be explained in more detail later on.

We define $\mathbf{\Omega}$ to be the disjoint union of all $\mathbf{C}_\rho$. Since the token sets of the $\mathbf{C}_\rho$ are all disjoint, we take as set of tokens of $\mathbf{\Omega}$ simply the union of the sets of tokens of all $\mathbf{C}_\rho$. A finite set of tokens of $\mathbf{\Omega}$ is called consistent if it is a consistent in some $\mathbf{C}_\rho$. Similarly, the entailment relation is inherited from the $\mathbf{C}_\rho$.

In $\mathbf{\Omega}$, consider the set F of pairs (U, b), where for some ρ, σ, both U and b have type $\rho \Rightarrow \sigma$, and $U \vdash b$. By definition, F can be viewed as an approximable map from $\mathbf{\Omega} \to \mathbf{\Omega}$ to $\mathbf{\Omega}$, and also as an approximable map from $\mathbf{\Omega}$ to $\mathbf{\Omega} \to \mathbf{\Omega}$. By the bijective correspondence between the continuous maps $f\colon |\mathbf{A}| \to |\mathbf{B}|$ and the approximable maps from $\mathbf{A}$ to $\mathbf{B}$ we can view F as a map $\text{Hat}\colon (|\mathbf{\Omega}| \to |\mathbf{\Omega}|) \to |\mathbf{\Omega}|$ and also as a map $\text{Mod}\colon |\mathbf{\Omega}| \to |\mathbf{\Omega}| \to |\mathbf{\Omega}|$. By construction, Hat and Mod are inverse to each other.

Clearly we have canonical injections and projections, that is, continuous maps $\text{in}_\rho\colon |\mathbf{C}_\rho| \to |\mathbf{\Omega}|$ and $\text{out}_\rho\colon |\mathbf{\Omega}| \to |\mathbf{C}_\rho|$. Both are determined by the same approximable map, consisting of all (U, b) with $U \vdash b$ where both U and b are of type ρ. In the special case of the type ι, recall that $\mathbf{C}_\iota := \mathbf{N} \to \mathbf{\Lambda}$. The injection $\text{in}_\iota\colon |\mathbf{C}_\iota| \to |\mathbf{\Omega}|$ is denoted by HatIota. Similarly, the projection $\text{out}_\iota\colon |\mathbf{\Omega}| \to |\mathbf{C}_\iota|$ is called ModIota.

We can easily define a continuous map $\text{Part}(.)$ from $|\mathbf{\Omega}|$ to the information system for types, which assigns to an ideal $x \in |\mathbf{\Omega}|$ the "part" of the disjoint union it belongs to.

It is easy to verify that

$$\begin{aligned} \text{Mod}(\text{in}_{\rho\Rightarrow\sigma}(u)) &= \text{in}_\sigma \circ u \circ \text{out}_\rho, \\ \text{Hat}(h) &= \bigcup_{\rho,\sigma} \text{in}_{\rho\Rightarrow\sigma}(\text{out}_\sigma \circ h \circ \text{in}_\rho). \end{aligned}$$

Let $\text{Hat}_{\rho,\sigma} := \text{out}_{\rho\Rightarrow\sigma} \circ \text{Hat}$.

4. The Normalization Theorem

4.1. Introduction

4.1.1. "Strong Computability"

The normalization proof due to Tait relies essentially on the notion of "Strong Computability". The corresponding predicate is SC.[2] Computability implies normalizability of a λ-term, as is to be shown in "Lemma 1". Hence the proof's intrinsic ambition is to show computability of all well-typed λ-terms.

Strong computability is originally defined along the type (cf. [1]):

$$\begin{aligned} \text{SC}^\iota(r) &:= \text{FN}^\iota(r) \\ \text{SC}^{\rho\Rightarrow\sigma}(r) &:= \forall s.\text{SC}^\rho(s) \to \text{SC}^\sigma(rs) \end{aligned}$$

$\text{FN}(r)$ means, that r is normalizable. In the formalization which this paper is based on we work in terms of *weak* normalization, that is $\text{FN}(r)$ means there exists *a* long normal form of r. With respect to program extraction (cf. [4], section 4) it is essential, that $\text{FN}(r)$ und therefore $\text{SC}^\iota(r)$ and also $\text{SC}^\rho(r)$ of arbitrary type ρ do have computational content. What is special about the definition however is, that the type $C_\rho := \tau(\text{SC}_\rho)$ of the predicate depends on ρ, where the C_ρ in correspondance to ρ can get arbitrarily complex. That would make a programming language with dependent types necessary

Hence the notion of Strong Computability is strictly speaking not a single predicate, but a family of countably many different predicates. In first order logic, which the proof

[2]As we restrict ourselves in this formalization to *weak* normalization, we use a kind of "weak" computability. Nevertheless we keep the established symbol SC.

system Minlog is also based on, such a family of predicates cannot be defined. If one defined SC through axioms, corresponding labels would appear in the extracted program, for which one could not determine the computational contents.

For this reason we run a different path here, which has in addition the essential advantage, that the realizer of “Lemma 2” turns out to be simply the identity. We define the predicate SC through a new predicate SC**r**. We use SC then only as an abbreviation. In accordance to our λ-calculus we add contexts:

$$\mathrm{SC}^{\rho}_{\vec{\rho}}(r) := \exists a\, \mathrm{SC}\mathbf{r}^{\rho}_{\vec{\rho}}(a, r)$$

We will define SC**r** in such a way, that it has *no* computational content, which is then “sourced out” into a. Accordingly a has to be of a universal type, which covers all the types C_ρ of the original predicate SC. The appropriate structure for such a type is the universal information system Ω from the previous section. Simultanously Ω represents an appropriate model for the simply typed λ-calculus. That is the variable a can be looked at as a λ-term interpreted in the model. This perspective will play a role in the next section, when it comes to Normalization by Evaluation.

With respect to the constructive η-expansion we define $\mathrm{FN}^{\iota}(r)$ not just as $\exists s \mathrm{N}^{\rho}_{\vec{\rho}}(r, s)$, but as $\forall k.\ \mathrm{Fr}^{\rho}_{\vec{\rho}}(r, k) \to \exists s \mathrm{N}^{\rho}_{\vec{\rho}}(r, s)$. Accordingly C_ι is $\mathbb{N} \to \Lambda$, which fits to the definition of the universal information system in the previous section. Furthermore we need to handle the variables a as partial variables and write “$\dot{a}$” as before.

The new predicate SC**r** is defined in such a way, that the original definition of SC still holds, but now in form of lemmas. We call these lemmas *SC-Lemmas*. For example case ι reads a little simplified as:

$$\forall^{\mathrm{nc}} \vec{\rho}, r.\ \exists \dot{a} \mathrm{SC}\mathbf{r}^{\iota}_{\vec{\rho}}(\dot{a}, r) \leftrightarrow \forall k.\ \mathrm{Fr}^{\iota}_{\vec{\rho}}(r, k) \to \exists s \mathrm{N}^{\iota}_{\vec{\rho}}(r, s)$$

It will turn out that the realizers of the SC-Lemmas are exactly the administrative functions ModIota, HatIota, Mod and Hat. In the above example it holds for the direction from left to right:

$$\mathrm{ModIota}\ \underline{\mathbf{r}}\ \forall^{\mathrm{nc}} \vec{\rho}, r.\ \exists \dot{a} \mathrm{SC}\mathbf{r}^{\iota}_{\vec{\rho}}(\dot{a}, r) \to \forall k.\ \mathrm{Fr}^{\iota}_{\vec{\rho}}(r, k) \to \exists s \mathrm{N}^{\iota}_{\vec{\rho}}(r, s)$$

and therefore also

$$\forall^{\mathrm{nc}} \vec{\rho}, r.\ \forall \dot{a}.\ \mathrm{SC}\mathbf{r}^{\iota}_{\vec{\rho}}(\dot{a}, r) \to \mathrm{ModIota}(\dot{a})\ \underline{\mathbf{r}}\ \forall k.\ \mathrm{Fr}^{\iota}_{\vec{\rho}}(r, k) \to \exists s \mathrm{N}^{\iota}_{\vec{\rho}}(r, s)$$

where the part $\forall k.\ \mathrm{Fr}^{\iota}_{\vec{\rho}}(r, k) \to \exists s \mathrm{N}^{\iota}_{\vec{\rho}}(r, s)$ at the end matches the original definition of $\mathrm{SC}^{\iota}_{\vec{\rho}}(r)$. Therefore by masking the administrative functions one could say suggestively that $\mathrm{SC}\mathbf{r}(\dot{a}, r)$ is defined such that $\dot{a}$ realizes the original definition of $\mathrm{SC}(r)$:

$$\text{“}\mathrm{SC}\mathbf{r}(\dot{a}, r) \to \dot{a}\ \underline{\mathbf{r}}\ \mathrm{SC}(r)\text{”}$$

4.1.2. The structure of the proof

We devide the normalization proof into three parts:

I. The General Part
II. The Main Part
III. The Part of the Axioms

The first part introduces the general notions and global auxiliaries used in the other two parts. The second part represents the main part of the normalization proof. The third part proves the “Axioms”, which are used in the main part.

The “Axioms” from the main part have no computational content and therefore have no effect on the extracted program. In the main part the predicates “N”, “A” and “Head” are not defined concretely, but are just delimited by the “Axioms”. That is the proof solely uses the properties of the predicates, which are given through the “Axioms”.

Even though the main part does give the complete extracted program, it does not provide a proof, that such predicates can be defined at all. Moreover it remains open, that there are such definitions, which grasp the intuitive meaning of the predicates and hence, that the extracted program does in fact what it is supposed to do: to compute the long normal form of a well-typed λ-term.

In the third part we finally give appropriate definitions for the predicates “N”, “A” and “Head”. That is we define them in such a way that they carry the intended meaning and prove that they satisfy the “Axioms” from the main part. All three parts together make up the complete normalization proof.

4.2. Part I: The General Part of the Normalization Proof

4.2.1. Particular Definitions for the Normalization Proof

The Predicates N, A, Head *and* Fr. As explained above we declare the predicates N, A and Head without giving concrete definitions. Those will be given in the part of the “Axioms”. Moreover we define the predicate Fr. All four predicates are supposed to have no computational content. The intuitive meaning of the predicates is as follows:

$\mathrm{N}^{\rho}_{\vec\rho}(r,s)$:	r and s are of type ρ in the context $\vec\rho$, and s is *a* long normal form of r, i.e. there is a β-normal form of r, whose η-expanded form is s.
$\mathrm{A}^{\rho}_{\vec\rho}(r,s)$:	r and s are of type ρ in the context $\vec\rho$; r is of the form $r = kr_1r_2\dots r_n$ and s of the form $s = ks_1s_2\dots s_n$ and $\mathrm{N}^{\rho}_{\vec\rho}(r_i,s_i)$ holds for all $i \in \{1,\dots,n\}$.
$\mathrm{Head}(r_1,r_2)$:	r_1 is of the form $r_1 = (\lambda^\rho t)s\vec{s}$ and r_2 of the form $r_2 = (t[s \star \uparrow^0])\vec{s}$
$\mathrm{Fr}^{\rho}_{\vec\rho}(r,k)$:	r is of type ρ in the context $\vec\rho$, and k is greater than all the free variables in r.

For simplicity we define the predicate Fr already here in the general part of the proof.

Definition 15 (Fr). *We define* Fr : $(\mathbb{L}_{\mathbb{T}}, \mathbb{T}, \Lambda, \mathbb{N})$ *as*

$$\boxed{\mathrm{Fr}^{\rho}_{\vec\rho}(r,k) :\Leftrightarrow (\vec\rho \vdash r : \rho) \wedge (k \geq \mathcal{L}(\vec\rho))}$$

Even though the definition slightly differs from the intuitive meaning above, it suffices for our purpose to get a *fresh* variable k.

FN *and* FA. We introduce FN and FA just as abbreviations.

Definition 16. $\boxed{\mathrm{FN}^{\rho}_{\vec{\rho}}(r) \equiv \forall k.\ \mathrm{Fr}^{\rho}_{\vec{\rho}}(r,k) \to \exists s \mathrm{N}^{\rho}_{\vec{\rho}}(r,s)}$

Definition 17. $\boxed{\mathrm{FA}^{\rho}_{\vec{\rho}}(r) \equiv \forall k.\ \mathrm{Fr}^{\rho}_{\vec{\rho}}(r,k) \to \exists s \mathrm{A}^{\rho}_{\vec{\rho}}(r,s)}$

The use of $\mathrm{Fr}^{\rho}_{\vec{\rho}}(r,k)$ is justified from a computational point of view, since we will need a fresh variable k to construct the long normal form.

SC *and* SC**r**. As explained above we introduce SC by means of the new predicate SC**r**, which is defined as a predicate constant by the following axioms:

Definition 18 (SCr). *We define* SC**r** : $(\mathbb{L}_{\mathbb{T}}, \mathbb{T}, \Omega, \Lambda)$ *as*

$$\boxed{\begin{array}{ll} \mathrm{SC}\mathbf{r}^{\iota}_{\vec{\rho}}(\dot{a}, r) & \leftrightarrow (\vec{\rho} \vdash r : \iota) \wedge (\mathrm{Part}(\dot{a}) = \iota) \wedge \\ & \quad (\forall k.\ \mathrm{Fr}^{\iota}_{\vec{\rho}}(r,k) \to \mathrm{N}^{\iota}_{\vec{\rho}}(r, \mathrm{ModIota}(\dot{a})(k))) \\ \mathrm{SC}\mathbf{r}^{\rho\Rightarrow\sigma}_{\vec{\rho}}(\dot{a}, r) & \leftrightarrow (\vec{\rho} \vdash r : \rho \Rightarrow \sigma) \wedge (\mathrm{Part}(\dot{a}) = \rho \Rightarrow \sigma) \wedge \\ & \quad (\forall \vec{\sigma}, \dot{b}, s.\ \mathrm{SC}\mathbf{r}^{\rho}_{\vec{\rho}\star\vec{\sigma}}(\dot{b}, s) \to \mathrm{SC}\mathbf{r}^{\sigma}_{\vec{\rho}\star\vec{\sigma}}(\mathrm{Mod}(\dot{a})(\dot{b}), rs)) \end{array}}$$

The original predicate SC is then only used as the following abbreviation:

Definition 19. $\boxed{\mathrm{SC}^{\rho}_{\vec{\rho}}(r) \equiv \exists \dot{a} \mathrm{SC}\mathbf{r}^{\rho}_{\vec{\rho}}(\dot{a}, r)}$

4.2.2. The SC-Lemmas

The original definition of SC still holds after its redefinition, now in form of lemmas, which we call *SC-Lemmas*.

Lemma 20 (LemmaSCIota). $\boxed{\forall^{\mathrm{nc}} \vec{\rho}, r.\ \mathrm{SC}^{\iota}_{\vec{\rho}}(r) \leftrightarrow ((\vec{\rho} \vdash r : \iota) \wedge \mathrm{FN}^{\iota}_{\vec{\rho}}(r))}$

Proof: Case "$\to$" is obvious. Case "$\leftarrow$" is provable with the Axiom of Choice. □

Lemma 21 (LemmaSC).

$$\boxed{\begin{array}{l} \forall^{\mathrm{nc}} \vec{\rho}, \rho, \sigma, r.\ \mathrm{SC}^{\rho\Rightarrow\sigma}_{\vec{\rho}}(r) \to ((\vec{\rho} \vdash r : \rho \Rightarrow \sigma) \wedge \forall^{\mathrm{nc}} \vec{\sigma}, s.\ \mathrm{SC}^{\rho}_{\vec{\rho}\star\vec{\sigma}}(s) \to \mathrm{SC}^{\sigma}_{\vec{\rho}\star\vec{\sigma}}(rs)) \\ \forall^{\mathrm{nc}} \vec{\rho}, r \forall \rho, \sigma.\ \mathrm{SC}^{\rho\Rightarrow\sigma}_{\vec{\rho}}(r) \leftarrow ((\vec{\rho} \vdash r : \rho \Rightarrow \sigma) \wedge \forall^{\mathrm{nc}} \vec{\sigma}, s.\ \mathrm{SC}^{\rho}_{\vec{\rho}\star\vec{\sigma}}(s) \to \mathrm{SC}^{\sigma}_{\vec{\rho}\star\vec{\sigma}}(rs)) \end{array}}$$

Proof: Case "$\to$" is obvious. Case "$\leftarrow$" is provable with the Axiom of Choice, Independence of Premises and a Uniformity Principle. □

$\overline{\mathrm{SC}}$ *and* $\overline{\mathrm{SC}\mathbf{r}}$. We introduce pluralforms $\overline{\mathrm{SC}}$ and $\overline{\mathrm{SC}\mathbf{r}}$ respectively with lists as arguments, meaning that SC and SC**r** apply on each single element of the list. In order to allow case distinction on lists with partial elements we introduce the notion of *structurally total* and its corresponding predicate $\mathrm{STotal}(\dot{\vec{a}})$.

Definition 22 (SCrs). *We define* $\overline{\mathrm{SC}\mathbf{r}}$: $(\mathbb{L}_{\mathbb{T}}, \mathbb{L}_{\mathbb{T}}, \mathbb{L}_{\Omega}, \mathbb{L}_{\Lambda})$ *as*

$$\boxed{\begin{array}{ll} \text{“SCrsDefNil”} & : \overline{\mathrm{SC}\mathbf{r}}^{\emptyset}_{\vec{\sigma}}(\emptyset, \emptyset) \\ \text{“SCrsDef”} & : \mathrm{STotal}(\dot{\vec{a}}) \to \mathrm{SC}\mathbf{r}^{\rho}_{\vec{\sigma}}(\dot{a}, s) \to \overline{\mathrm{SC}\mathbf{r}}^{\vec{\rho}}_{\vec{\sigma}}(\dot{\vec{a}}, \vec{s}) \to \overline{\mathrm{SC}\mathbf{r}}^{\rho\star\vec{\rho}}_{\vec{\sigma}}(\dot{a} \star \dot{\vec{a}}, s \star \vec{s}) \end{array}}$$

Then it follows by induction on $\overline{\text{SC}\mathbf{r}}$:

Lemma 23 (SCrsSTotal). $\overline{\text{SC}\mathbf{r}}^{\vec{\rho}}_{\vec{\sigma}}(\dot{\vec{a}}, \vec{s}) \to \text{STotal}(\dot{\vec{a}})$

4.2.3. Specific Definitions

The abbreviation $\lambda^{\rho}_{k}s$ represents the binding of the k-th variable in the term s by a leading λ^{ρ}.

Definition 24 (ABS). *We define* $\lambda^{-}_{-} : \mathbb{N} \to \mathbb{T} \to \Lambda \to \Lambda$ *as*

$$\lambda^{\rho}_{k}s := \lambda^{\rho}(s[\nearrow^{k}_{1} \star 0 \star \uparrow^{k+2}])$$

Furthermore we need the extension $\varepsilon^{\rho}_{k}(\vec{\rho})$ of a context $\vec{\rho}$, such that k lies in $\vec{\rho} \star \varepsilon^{\rho}_{k}(\vec{\rho})$, which we also abbreviate as $\vec{\rho} \star \varepsilon^{\rho}_{k}$:

Definition 25 (ExtCtx). *We define* $\varepsilon : \mathbb{L}_{\mathbb{T}} \to \mathbb{N} \to \mathbb{T} \to \mathbb{L}_{\mathbb{T}}$ *by*

$$\begin{array}{ll}
\varepsilon^{\rho}_{0}(\emptyset) & := \rho \\
\varepsilon^{\rho}_{0}(\sigma \star \vec{\rho}) & := \emptyset \\
\varepsilon^{\rho}_{k+1}(\emptyset) & := \rho \star (\varepsilon^{\rho}_{k}(\emptyset)) \\
\varepsilon^{\rho}_{k+1}(\sigma \star \vec{\rho}) & := \varepsilon^{\rho}_{k}(\vec{\rho})
\end{array}$$

Definition 26 (thof). *We define* $k \gg \vec{l}$ *as the* k*-th element of an arbitrary list.*

We introduce the plural form $\Vdash$ of the type judgement $\vdash$:

Definition 27 (TypJs). *We define* $\Vdash : \mathbb{L}_{\mathbb{T}} \to \mathbb{L}_{\Lambda} \to \mathbb{L}_{\mathbb{T}} \to \mathbb{B}$ *as*

$$\begin{array}{ll}
\vec{\sigma} \Vdash \emptyset : \emptyset & := T \\
\vec{\sigma} \Vdash \emptyset : (\rho \star \vec{\rho}) & := F \\
\vec{\sigma} \Vdash (r \star \vec{r}) : \emptyset & := F \\
\vec{\sigma} \Vdash (r \star \vec{r}) : (\rho \star \vec{\rho}) & := (\vec{\sigma} \vdash r : \rho) \text{ and } (\vec{\sigma} \Vdash \vec{r} : \vec{\rho})
\end{array}$$

4.2.4. The "Axioms"

The following "Axioms" are assumed in the main part and proven in the third part. As the predicates are supposed to have no computational content, the "Axioms" have none either.

$$\boxed{\begin{array}{ll}
\text{“Ax1”} & : \text{Fr}^{\rho\Rightarrow\sigma}_{\vec{\rho}}(r,k) \to \text{N}^{\sigma}_{\vec{\rho}\star\varepsilon^{\rho}_{k}}(rk,s) \to \text{N}^{\rho\Rightarrow\sigma}_{\vec{\rho}}(r,\lambda^{\rho}_{k}s) \\
\text{“Ax2”} & : \text{A}^{\iota}_{\vec{\rho}}(r,s) \to \text{N}^{\iota}_{\vec{\rho}}(r,s) \\
\text{“Ax3”} & : (\vec{\rho} \vdash k : \rho) \to \text{A}^{\rho}_{\vec{\rho}}(k,k) \\
\text{“Ax4”} & : \text{A}^{\rho\Rightarrow\sigma}_{\vec{\rho}}(r_1,s_1) \to (\vec{\rho} \vdash r_2 : \rho) \to \text{N}^{\rho}_{\vec{\rho}}(r_2,s_2) \to \text{A}^{\sigma}_{\vec{\rho}}(r_1r_2,s_1s_2) \\
\text{“Ax5”} & : \text{Head}(r_1,r_2) \to \text{N}^{\rho}_{\vec{\rho}}(r_2,\dot{s}) \to (\vec{\rho} \vdash r_1 : \rho) \to \text{N}^{\rho}_{\vec{\rho}}(r_1,\dot{s}) \\
\text{“Ax6”} & : \text{Head}((\lambda^{\rho}r)[\theta]\,s, r[s \star \theta]) \\
\text{“Ax7”} & : \text{Head}(r,s) \to \text{Head}(rt,st) \\
\text{“Ax8”} & : \text{Head}(r,s) \to \text{Fr}^{\rho}_{\vec{\rho}}(r,k) \to \text{Fr}^{\rho}_{\vec{\rho}}(s,k) \\
\text{“Ax9”} & : \text{N}^{\rho}_{\vec{\rho}}(r,\dot{s}) \to \text{N}^{\rho}_{\vec{\rho}\star\vec{\sigma}}(r,\dot{s}) \\
\text{“Ax10”} & : \text{A}^{\rho}_{\vec{\rho}}(r,s) \to \text{A}^{\rho}_{\vec{\rho}\star\vec{\sigma}}(r,s)
\end{array}}$$

4.2.5. Global Auxiliaries

The following global auxiliaries are used in Part II and in Part III of the proof:

Lemma 28 (TypJAppIntro). $(\vec{\rho} \vdash r : \rho \Rightarrow \sigma) \to (\vec{\rho} \star \vec{\sigma} \vdash s : \rho) \to (\vec{\rho} \star \vec{\sigma} \vdash rs : \sigma)$

Lemma 29 (SubVar). $k < \mathcal{L}(\vec{s}) \to k[\vec{s} \star \uparrow^m] = k \gg \vec{s}$

Lemma 30 (SCrsLh). $\overline{\mathrm{SC}\mathbf{r}}^{\vec{\rho}}_{\vec{\sigma}}(\dot{\vec{a}}, \vec{s}) \to \mathcal{L}(\dot{\vec{a}}) = \mathcal{L}(\vec{\rho}) \wedge \mathcal{L}(\vec{\rho}) = \mathcal{L}(\vec{s})$

Lemma 31 (TypJSub). $(\vec{\rho} \vdash r : \rho) \to (\vec{\sigma} \Vdash \vec{s} : \vec{\rho}) \to (\vec{\sigma} \vdash r[\vec{s} \star \uparrow^0] : \rho)$

Lemma 32 (TypJExtCtx). $(\vec{\rho} \vdash r : \rho) \to (\vec{\rho} \star \vec{\sigma} \vdash r : \rho)$

Proof: Lemma 28 clearly holds. Lemma 29 holds by induction on k and $\vec{s}$. Lemma 30 holds by induction on $\overline{\mathrm{SC}\mathbf{r}}$. Lemma 31 holds by induction on r and ρ. Lemma 32 holds by induction on r and ρ. □

4.3. Part II: The Main Part of the Normalization Proof

Traditionally the proof of the main part consists of three main lemmas and the final Normalization Theorem. The detailed proofs can be found in [3].

4.3.1. "Lemma 1", "Lemma 2" and "Lemma 3"

Lemma 33 ("Lemma 1", LemmaOne).

$$\boxed{\forall\rho\forall^{\mathrm{nc}}\vec{\rho}, r.\ (\vec{\rho} \vdash r : \rho) \to (\mathrm{SC}^{\rho}_{\vec{\rho}}(r) \to \mathrm{FN}^{\rho}_{\vec{\rho}}(r)) \wedge (\mathrm{FA}^{\rho}_{\vec{\rho}}(r) \to \mathrm{SC}^{\rho}_{\vec{\rho}}(r))}$$

Proof: Induction on ρ and by means of the "Axioms 1", "Axioms 2", "Axioms 3", "Axioms 4" and "Axioms 10". □

Lemma 34 ("Lemma 2", LemmaTwo).

$$\boxed{\forall^{\mathrm{nc}}\rho, \vec{\rho}, r, s.(\vec{\rho} \vdash r : \rho) \to \mathrm{SC}^{\rho}_{\vec{\rho}}(s) \to \mathrm{Head}(r, s) \to \mathrm{SC}^{\rho}_{\vec{\rho}}(r)}$$

Proof: By the generalization

$$\forall^{\mathrm{nc}}\rho, \vec{\rho}, r, s, \dot{a}.(\vec{\rho} \vdash r : \rho) \to \mathrm{SC}\mathbf{r}^{\rho}_{\vec{\rho}}(\dot{a}, s) \to \mathrm{Head}(r, s) \to \mathrm{SC}\mathbf{r}^{\rho}_{\vec{\rho}}(\dot{a}, r)$$

which holds by induction on ρ and by "Axioms 5", "Axioms 7" and "Axioms 8". □

Lemma 35 ("Lemma 3", LemmaThree).

$$\boxed{\forall r, \vec{\rho}\forall^{\mathrm{nc}}\vec{\sigma}, \rho, \vec{s}.\ (\vec{\rho} \vdash r : \rho) \to \overline{\mathrm{SC}}^{\vec{\rho}}_{\vec{\sigma}}(\vec{s}) \to \mathrm{SC}^{\rho}_{\vec{\sigma}}(r[\vec{s} \star \uparrow^0])}$$

Proof: Induction on r and by means of "Axioms 6" and "Axioms 9". □

4.3.2. *The Normalization Theorem*

The following auxiliaries are used in the Normalization Theorem:

Lemma 36 (TypJVar). $(\vec{\sigma} \star \rho \star \vec{\rho} \vdash \mathcal{L}(\vec{\sigma}) : \rho)$

Lemma 37 (SubIds). $\mathrm{Cor}_{\vec{\rho}}(r) \to r[\nearrow^{\mathcal{L}(\vec{\rho})} \star \uparrow^0] = r$

Lemma 38 (FrIntro1). $(\vec{\rho} \vdash r : \rho) \to \mathrm{Fr}^{\rho}_{\vec{\rho}}(r, \mathcal{L}(\vec{\rho}))$

Lemma 39 (SCrsSeq). $\overline{\mathrm{SC}}^{\vec{\rho}}_{\vec{\sigma}\star\vec{\rho}}(\nearrow^{\mathcal{L}(\vec{\rho})}_{\mathcal{L}(\vec{\sigma})})$

Proof: Lemma 36 holds by simple induction on $\vec{\sigma}$. Lemma 37 holds by induction on r. Lemma 38 holds by definition of Fr. Lemma 39 holds by induction on $\vec{\rho}$ and by means of "Axiom 3" and lemma 36. □

Remark 40. *The above lemma 39 ("SCrsSeq"), which also uses "Axiom 3", has computational content and does appear in the extracted program.*

Theorem 41 ("Normalization Theorem", NTheorem).

$$\boxed{\forall \vec{\rho}, r.\ \mathrm{Cor}_{\vec{\rho}}(r) \to \exists s \mathrm{N}^{\mathrm{Typ}_{\vec{\rho}}(r)}_{\vec{\rho}}(r, s)}$$

Proof: $\mathrm{Cor}_{\vec{\rho}}(r)$ is given. By setting $\rho := \mathrm{Typ}_{\vec{\rho}}(r)$ we have $\vec{\rho} \vdash r : \rho$. From lemma 39 it follows that $\overline{\mathrm{SC}}^{\vec{\rho}}_{\vec{\rho}}(\nearrow^{\mathcal{L}(\vec{\rho})})$. From this and by "Lemma 3" we get $\mathrm{SC}^{\rho}_{\vec{\rho}}(r[\nearrow^{\mathcal{L}(\vec{\rho})} \star \uparrow^0])$, which can be simplified by lemma 37 to $\mathrm{SC}^{\rho}_{\vec{\rho}}(r)$. By "Lemma 1" $\mathrm{FN}^{\rho}_{\vec{\rho}}(r)$ follows. Due to lemma 38 we have $\mathrm{Fr}^{\rho}_{\vec{\rho}}(r, \mathcal{L}(\vec{\rho}))$. Thus by the definition of FN we get $\exists s \mathrm{N}^{\rho}_{\vec{\rho}}(r, s)$. □

4.4. *Part III: The Part of the Axioms*

We will give concrete definitions of the predicates "N", "A" and "Head" and then prove the "Axioms" from the main part (see section 4.2.4). "Axiom 1" is by far the most complex. We will therefore focus on this "Axiom" and give the core of its proof in full detail.

4.4.1. *The Definition of the Predicates*

The Definition of "N". $\mathrm{N}^{\rho}_{\vec{\rho}}(r, s)$ is supposed to express, that r and s are of type ρ in the context $\vec{\rho}$ and that s is *a* long normal form of r. For that purpose we define two more predicates "BN" (for β-normal form) and "Exp" (for η-expansion), and simultaneously their plural forms "$\overline{\mathrm{BN}}$" and "$\overline{\mathrm{Exp}}$".

Definition 42 (BN, BNs). $\mathrm{BN} : (\Lambda, \Lambda)$, $\overline{\mathrm{BN}} : (\mathbb{L}_\Lambda, \mathbb{L}_\Lambda)$

"BNVar"	$: \overline{\mathrm{BN}}(\vec{r}, \vec{s}) \to \mathrm{BN}(n\vec{r}, n\vec{s})$
"BNAbs"	$: \mathrm{BN}(r, s) \to \mathrm{BN}(\lambda^\rho r, \lambda^\rho s)$
"BNBeta"	$: \mathrm{BN}(r[s \star \uparrow^0]\,\vec{r}, t) \to \mathrm{BN}(\lambda^\rho r(s \star \vec{r}), t)$
"BNsNil"	$: \overline{\mathrm{BN}}(\emptyset, \emptyset)$
"BNsCons"	$: \mathrm{BN}(r, s) \to \overline{\mathrm{BN}}(\vec{r}, \vec{s}) \to \overline{\mathrm{BN}}(r \star \vec{r}, s \star \vec{s})$

Remark 43. *The definition is motivated by the alternative representation of the λ-calculus $\Lambda ::= n\vec{r} \mid \lambda^\rho r \mid (\lambda^\rho r)(s \star \vec{r})$, where the set of β-normal forms can be generated by the first two cases as* $\mathrm{NF}_\beta ::= n\vec{r} \mid \lambda^\rho r$.

For the predicate Exp we need a context due to η-expansion. One could try to define Exp recursively unlike in the case of BN, where the proof of totality would in fact correspond to a proof of (strong) normalization. But the inductive approach for Exp turns out to be advantageous in the proof.

The part of Exp however, which is called the *outside expansion* and denoted by η, we define recursively. Here the recursive approach is more straightforward and clearer.

Definition 44 (Eta). *We define* $\eta : \mathbb{T} \to \Lambda \to \Lambda$ *as*

$$\begin{aligned} \eta^\iota(r) &:= r \\ \eta^{\rho \Rightarrow \sigma}(r) &:= \lambda^\rho(\eta^\sigma(r \uparrow^1 \eta^\rho(0))) \end{aligned}$$

Definition 45. *We define* $\twoheadrightarrow : \mathbb{L}_\mathbb{T} \to \mathbb{T} \to \mathbb{T}$ *as*

$$\begin{aligned} \varnothing \twoheadrightarrow \rho &:= \rho \\ (\sigma \star \vec{\rho}) \twoheadrightarrow \rho &:= \sigma \Rightarrow (\vec{\rho} \twoheadrightarrow \rho) \end{aligned}$$

Definition 46 (Exp, Exps). $\mathrm{Exp} : (\mathbb{L}_\mathbb{T}, \mathbb{T}, \Lambda, \Lambda)$, $\overline{\mathrm{Exp}} : (\mathbb{L}_\mathbb{T}, \mathbb{L}_\mathbb{T}, \mathbb{L}_\Lambda, \mathbb{L}_\Lambda)$

$$\boxed{\begin{array}{ll} \text{“ExpVar”} & : (\vec{\rho} \vdash k : \vec{\sigma} \twoheadrightarrow \rho) \to \overline{\mathrm{Exp}}^{\vec{\sigma}}_{\vec{\rho}}(\vec{r}, \vec{s}) \to (t = \eta^\rho(k\vec{s})) \to \mathrm{Exp}^\rho_{\vec{\rho}}(k\vec{r}, t) \\ \text{“ExpAbs”} & : (\vec{\rho}_1 \star \vec{\rho}_2 \star \vec{\sigma} \vdash k : \rho) \to \mathrm{Exp}^\sigma_{\vec{\rho}_1 \star \vec{\rho}_2}(r, s) \to \\ & \quad (\vec{\rho}_1 \vdash \lambda^\rho_k r : \rho \Rightarrow \sigma) \to \mathrm{Exp}^{\rho \Rightarrow \sigma}_{\vec{\rho}_1 \star \vec{\tau}}(\lambda^\rho_k r, \lambda^\rho_k s) \\ & \\ \text{“ExpsNil”} & : \overline{\mathrm{Exp}}^{\varnothing}_{\vec{\rho}}(\varnothing, \varnothing) \\ \text{“ExpsCons”} & : \mathrm{Exp}^\rho_{\vec{\rho}}(r, s) \to \overline{\mathrm{Exp}}^{\vec{\sigma}}_{\vec{\rho}}(\vec{r}, \vec{s}) \to \overline{\mathrm{Exp}}^{\rho \star \vec{\sigma}}_{\vec{\rho}}(r \star \vec{r}, s \star \vec{s}) \end{array}}$$

Remark 47. *The definition of* Exp *is motivated by the alternative representation of the λ-calculus as in case of* BN *before. Since we need* Exp *only as part of* N, *we can restrict the predicate to λ-terms which are already β-reduced (cf. remark 43) thus making proofs shorter.*

The definition of Exp particularly takes into account the requirements of “Axiom 1” and “Axiom 9”. “Axiom 1” requires that the context can be reduced, while “Axiom 9” on the contrary requires an extension of the context. It suffices to take this into consideration in the second case “ExpAbs” only, however one has to be all the more careful here.

If there the contexts in the subformulas were kept fixed, the definition would clearly not suffice. Induction on Exp would fail for the extension as well as for the reduction of the context. If on the other hand an arbitrary context would be in the last part of the formula, cases would occur, where $\mathrm{Exp}^\rho_{\vec{\rho}}(r, s)$ holds and even r and s are of type ρ in the context $\vec{\rho}$, but nevertheless s not being the long normal form of r.

The type judgement $(\vec{\rho}_1 \vdash \lambda^\rho_k r : \rho \Rightarrow \sigma)$ and the extended context $\vec{\tau}$ in the last part of the formula however do lead to this *precise* definition of the predicate. While $\vec{\tau}$ obviously allows to extend the context, the type judgement allows to reduce the original context $(\vec{\rho}_1 \star \vec{\rho}_2)$ appropriately to $\vec{\rho}_1$ avoiding irregular cases. In fact the type judgement even suffices to get the following desirable result:

Lemma 48 (ExpTypJ). $\mathrm{Exp}^{\rho}_{\vec{\rho}}(r,s) \to (\vec{\rho} \vdash r : \rho) \wedge (\vec{\rho} \vdash s : \rho)$

Proof: $(\vec{\rho} \vdash r : \rho)$ can be proven by induction on Exp. $(\vec{\rho} \vdash s : \rho)$ holds due to the fact, that in the definition of $\mathrm{Exp}^{\rho}_{\vec{\rho}}(r,s)$ the free variables of r and s are kept identical at each stage. □

We can now define the predicate N as follows:

Definition 49 (N). *We define* $\mathrm{N} : (\mathbb{L}_{\mathbb{T}}, \mathbb{T}, \Lambda, \Lambda)$

$$\boxed{\text{“NIntro”} : (\vec{\rho} \vdash r : \rho) \to \mathrm{BN}(r,t) \to \mathrm{Exp}^{\rho}_{\vec{\rho}}(t,s) \to \mathrm{N}^{\rho}_{\vec{\rho}}(r,s)}$$

Remark 50. *In the definition* total *variables are used (as in all definitions of the predicates). This guarantees that from* $\mathrm{N}^{\rho}_{\vec{\rho}}(\dot{r},\dot{s})$ *it follows that r and s are* total*.*

As required in the informal description of N from subsection 4.2.1, it remains to show, that also s is of correct type. For that we need the so-called *Subject Reduction*, which also appears in the proofs of “Axiom 1” and “Axiom 8”.

Lemma 51 (“Subject Reduction”, SR). $(\vec{\rho} \vdash (\lambda^{\sigma} r)s : \rho) \to (\vec{\rho} \vdash r[s \star \uparrow^0] : \rho)$

Proof: Provable on the basis of lemma 31. □

Remark 52.

1. *The suggesting attempt to prove the Subject Reduction by induction on r fails in the abstraction case.*
2. *The converse of the Subject Reduction* $(\vec{\rho} \vdash r[s \star \uparrow^0] : \rho) \to (\vec{\rho} \vdash (\lambda^{\sigma} r)s : \rho)$ *is clearly* not *derivable.*

With the Subject Reduction we also get preservation of type for BN:

Lemma 53 (BNTypJ). $\mathrm{BN}(r,s) \to \forall \vec{\rho}, \rho.\ (\vec{\rho} \vdash r : \rho) \to (\vec{\rho} \vdash s : \rho)$

Proof: Provable on the basis of the *Subject Reduction* (lemma 51). □

Remark 54. *The converse* $\forall r, s.\ \mathrm{BN}(r,s) \to \forall \vec{\rho}, \rho.\ (\vec{\rho} \vdash s : \rho) \to (\vec{\rho} \vdash r : \rho)$ *does* not *hold in general (cf. remark 52). That's why the type judgement* $(\vec{\rho} \vdash r : \rho)$ *is in the definition of* N*. Otherwise one could derive the type for r already from* $\mathrm{Exp}^{\rho}_{\vec{\rho}}(t,s)$*.*

We finally get preservation of type for N:

Lemma 55 (NTypJ). $\mathrm{N}^{\rho}_{\vec{\rho}}(r,s) \to (\vec{\rho} \vdash r : \rho) \wedge (\vec{\rho} \vdash s : \rho)$

Proof: By means of lemma 53 and lemma 48. □

Hence the predicate N corresponds with its informal description from section 4.2.1.

Definition 56 (Head). *We define* $\mathrm{Head} : (\Lambda, \Lambda)$ *as*

$$\boxed{\text{“HeadCon”} : \mathrm{Head}((\lambda^{\rho} r)s\vec{r}, r[s \star \uparrow^0]\,\vec{r})}$$

Definition 57 (A). *We define* $\mathrm{A} : (\mathbb{L}_{\mathbb{T}}, \mathbb{T}, \Lambda, \Lambda)$ *as*

$$\boxed{\begin{array}{l}\text{“AIndVar”}\ : (\vec\rho \vdash k : \rho) \to \mathrm{A}^{\rho}_{\vec\rho}(k,k) \\ \text{“AIndApp”} : \mathrm{A}^{\rho\Rightarrow\sigma}_{\vec\rho}(r,s) \to (\vec\rho \vdash r_1 : \rho) \to \mathrm{N}^{\rho}_{\vec\rho}(r_1,s_1) \to \mathrm{A}^{\sigma}_{\vec\rho}(rr_1, ss_1)\end{array}}$$

Lemma 58 (ATypJ). $\forall \vec\rho, \rho, r, s.\ \mathrm{A}^{\rho}_{\vec\rho}(r,s) \to (\vec\rho \vdash r : \rho)$

Proof: Induction on A. $\square$

This finishes the definition of the predicates. We now give the proofs of the “Axioms”. Some of the basic proof ideas of “Axiom 1” and “Axiom 2” are due to Pierre Letouzey.

4.4.2. The Proof of Axiom 1

For “Axiom 1” we need the auxiliaries listed below. Moreover “Axiom 1” requires the possibility to *reduce* the context in the predicate $\mathrm{Exp}^{\rho}_{\vec\rho}(r,s)$. This is allowed by the specific definition of Exp (see definition 46).

Lemma 59 (BNsTypJ). $\overline{\mathrm{BN}}(\vec r, \vec s) \to (\vec\rho \Vvdash \vec r : \vec\sigma) \to (\vec\rho \Vvdash \vec s : \vec\sigma)$

Lemma 60 (ExpsRedCtx). $\overline{\mathrm{Exp}}^{\vec\sigma}_{\vec\rho_1 \star \vec\rho_2}(\vec r, \vec s) \to (\vec\rho_1 \Vvdash \vec r : \vec\sigma) \to \overline{\mathrm{Exp}}^{\vec\sigma}_{\vec\rho_1}(\vec r, \vec s)$

Lemma 61 (ExpsTypJs). $\overline{\mathrm{Exp}}^{\vec\sigma}_{\vec\rho}(\vec r, \vec s) \to (\vec\rho \Vvdash \vec r : \vec\sigma) \wedge (\vec\rho \Vvdash \vec s : \vec\sigma)$

Lemma 62 (SubThroughEta). $\eta^{\rho}(r)[\theta] = \eta^{\rho}(r[\theta])$

Lemma 63 (ABSLift). $\mathrm{Fr}^{\rho}_{\vec\rho}(r,k) \to r[\nearrow^{k}_{1} \star 0 \star \uparrow^{k+2}] = r \uparrow^{1}$

Remark 64. *From lemma 63 it immediately follows that* $\mathrm{Fr}^{\rho}_{\vec\rho}(r,k) \to \lambda^{\rho}_{k} r = \lambda^{\rho}(r \uparrow^{1})$

Lemma 65 (Ax1Aux21). $\mathrm{BN}(r_1[k \star \uparrow^{0}], t) \to \exists t_1.\ t = t_1[k \star \uparrow^{0}] \wedge \mathrm{BN}(r_1, t_1)$

Lemma 66 (Ax1Aux22). $\mathrm{Fr}^{\rho\Rightarrow\sigma}_{\vec\rho}(\lambda^{\rho} r, k) \to \mathrm{Fr}^{\sigma}_{\rho\star\vec\rho}(r, (k+1))$

Lemma 67 (Ax1Aux23). $\mathrm{Fr}^{\tau}_{\vec\rho}(r, (k+1)) \to \lambda^{\rho} r = \lambda^{\rho}_{k}(r[k \star \uparrow^{0}])$

Lemma 68 (Ax1Aux24). $\mathrm{BN}(r,t) \to \mathrm{Fr}^{\rho}_{\vec\rho}(r,k) \to \mathrm{Fr}^{\rho}_{\vec\rho}(t,k)$

Proof: Lemma 59 holds by means of lemma 53. Lemma 60 holds by induction on $\vec r$ and by the definition of Exp. Lemma 61 holds by means of lemma 48. Lemma 62 holds by induction on ρ. Lemma 63 follows from the more general lemma

$$\mathrm{Fr}^{\rho}_{\vec\rho}(r,k) \to r[\nearrow^{m} \star \nearrow^{k}_{m+1} \star m \star \uparrow^{(m+k)+2}] = r \uparrow^{1}_{m}$$

which holds by induction on r. Lemma 65 holds by induction on BN. Lemma 66 holds by the definition of Fr. Lemma 67 holds by induction on k. Lemma 68 holds by means of lemma 53. $\square$

Lemma 69 (“Axiom 1”, Ax1). $\boxed{\mathrm{Fr}^{\rho\Rightarrow\sigma}_{\vec\rho}(r,k) \to \mathrm{N}^{\sigma}_{\vec\rho\star\varepsilon^{\rho}_{k}}(rk, s) \to \mathrm{N}^{\rho\Rightarrow\sigma}_{\vec\rho}(r, \lambda^{\rho}_{k} s)}$

Proof: Given $\mathrm{Fr}_{\vec{\rho}}^{\rho\Rightarrow\sigma}(r,k)$ and $\mathrm{N}^{\sigma}_{\vec{\rho}\star\varepsilon^{\rho}_{k}}(rk,s)$, we need to show $\mathrm{N}_{\vec{\rho}}^{\rho\Rightarrow\sigma}(r,\lambda^{\rho}_{k}s)$. Through inversion on $\mathrm{N}^{\sigma}_{\vec{\rho}\star\varepsilon^{\rho}_{k}}(rk,s)$ we get $(\vec{\rho}\star\varepsilon^{\rho}_{k}\vdash rk:\sigma)$, $\mathrm{BN}(rk,t)$ and $\mathrm{Exp}^{\sigma}_{\vec{\rho}\star\varepsilon^{\rho}_{k}}(t,s)$. In order to obtain a strong enough induction hypothesis, we set $\tilde{r}:=rk$ and show the generalization

$$\forall r_{\alpha}.\, \tilde{r}=r_{\alpha}k\to \mathrm{Exp}^{\sigma}_{\vec{\rho}\star\varepsilon^{\rho}_{k}}(t,s)\to \mathrm{Fr}_{\vec{\rho}}^{\rho\Rightarrow\sigma}(r_{\alpha},k)\to \mathrm{N}_{\vec{\rho}}^{\rho\Rightarrow\sigma}(r_{\alpha},\lambda^{\rho}_{k}s)$$

We prove this by induction on $\mathrm{BN}(\tilde{r},t)$:

1. **Case** “BNVar” :
$\mathrm{BN}(n\vec{r},n\vec{t})$ is given through $\overline{\mathrm{BN}}(\vec{r},\vec{t})$. Moreover we have $n\vec{r}=r_{\alpha}k$, $\mathrm{Exp}^{\sigma}_{\vec{\rho}\star\varepsilon^{\rho}_{k}}(n\vec{t},s)$ and $\mathrm{Fr}_{\vec{\rho}}^{\rho\Rightarrow\sigma}(r_{\alpha},k)$. We need to show $\mathrm{N}_{\vec{\rho}}^{\rho\Rightarrow\sigma}(r_{\alpha},\lambda^{\rho}_{k}s)$. From the assumptions it follows that r_{α} is of the form $r_{\alpha}=n\vec{r}_1$ and $\vec{r}$ of the form $\vec{r}=(\vec{r}_1\star k)$. Hence we have $\overline{\mathrm{BN}}((\vec{r}_1\star k),\vec{t})$ and therefore $\vec{t}=(\vec{t}_1\star k)$. We obtain

   $\triangle\mathbf{1}$: $\mathrm{Exp}^{\sigma}_{\vec{\rho}\star\varepsilon^{\rho}_{k}}(n(\vec{t}_1\star k),s)$
   $\triangle\mathbf{2}$: $\mathrm{Fr}_{\vec{\rho}}^{\rho\Rightarrow\sigma}(n\vec{r}_1,k)$
   $\triangle\mathbf{3}$: $\overline{\mathrm{BN}}(\vec{r}_1,\vec{t}_1)$

   We need to show $\mathrm{N}_{\vec{\rho}}^{\rho\Rightarrow\sigma}(n\vec{r}_1,\lambda^{\rho}_{k}s)$. By the definition of N it suffices to show:

   $\mathbf{I}$: $(\vec{\rho}\vdash n\vec{r}_1:\rho\Rightarrow\sigma)$
   $\mathbf{II}$: $\mathrm{BN}(n\vec{r}_1,n\vec{t}_1)$
   $\mathbf{III}$: $\mathrm{Exp}_{\vec{\rho}}^{\rho\Rightarrow\sigma}(n\vec{t}_1,\lambda^{\rho}_{k}s)$

   $\mathbf{I}$ follows from $\triangle\mathbf{2}$ by the definition of Fr. $\mathbf{II}$ follows from $\triangle\mathbf{3}$ by the definition of BN. $\mathbf{III}$ we prove by inversion on $\triangle\mathbf{1}$: We obtain

   $\triangle\mathbf{4}$: $(\vec{\rho}\star\varepsilon^{\rho}_{k}\vdash n:\vec{\sigma}\twoheadrightarrow\sigma)$
   $\triangle\mathbf{5}$: $s=\eta^{\sigma}(n\vec{s})$
   $\triangle\mathbf{6}$: $\overline{\mathrm{Exp}}^{\vec{\sigma}}_{\vec{\rho}\star\varepsilon^{\rho}_{k}}((\vec{t}_1\star k),\vec{s})$

   $\triangle\mathbf{6}$ implies that $\vec{s}$ is of the form $\vec{s}=\vec{s}_1\star s_1$ and $\vec{\sigma}$ of the form $\vec{\sigma}=\vec{\sigma}_1\star\sigma_1$. Moreover it follows that

   $\triangle\mathbf{7}$: $\overline{\mathrm{Exp}}^{\vec{\sigma}_1}_{\vec{\rho}\star\varepsilon^{\rho}_{k}}(\vec{t}_1,\vec{s}_1)$
   $\triangle\mathbf{8}$: $\mathrm{Exp}^{\sigma_1}_{\vec{\rho}\star\varepsilon^{\rho}_{k}}(k,s_1)$

   Together with $\triangle\mathbf{2}$ it follows that $\sigma_1=\rho$. Now in order to show $\mathbf{III}$ it suffices by the definition of Exp to show:

   $\mathbf{III}.\mathbf{1}$: $\vec{\rho}\vdash n:\vec{\sigma}_1\twoheadrightarrow(\rho\Rightarrow\sigma)$
   $\mathbf{III}.\mathbf{2}$: $\overline{\mathrm{Exp}}^{\vec{\sigma}_1}_{\vec{\rho}}(\vec{t}_1,\vec{s}_1)$
   $\mathbf{III}.\mathbf{3}$: $\lambda^{\rho}_{k}s=\eta^{\rho\Rightarrow\sigma}(n\vec{s}_1)$

   $\mathbf{III}.\mathbf{1}$ follows from $\triangle\mathbf{4}$, the form of $\vec{\sigma}$ and by means of $\triangle\mathbf{2}$. This implies as well $\vec{\rho}\Vdash\vec{r}_1:\vec{\sigma}_1$. By lemma 59 and $\triangle\mathbf{3}$ we get $\vec{\rho}\Vdash\vec{t}_1:\vec{\sigma}_1$. By lemma 60 using $\triangle\mathbf{7}$ this implies

   $\triangle\mathbf{9}$: $\overline{\mathrm{Exp}}^{\vec{\sigma}_1}_{\vec{\rho}}(\vec{t}_1,\vec{s}_1)$

   that is $\mathbf{III}.\mathbf{2}$. It remains to show $\mathbf{III}.\mathbf{3}$. Lemma 61, $\triangle\mathbf{2}$ and $\triangle\mathbf{9}$ lead to

   $\triangle\mathbf{10}$:: $\mathrm{Fr}_{\vec{\rho}}^{\rho\Rightarrow\sigma}(n\vec{s}_1,k)$

From $\triangle\mathbf{8}$ we get by inversion $s_1 = \eta^\rho(k)$. Together with $\triangle\mathbf{5}$ and the form of $\vec{s}$ this implies $s = \eta^\sigma((n\vec{s}_1)\eta^\rho(k))$. Hence we have to show

$\mathbf{III.3'}$: $\lambda^\rho_k(\eta^\sigma((n\vec{s}_1)\eta^\rho(k))) = \eta^{\rho\Rightarrow\sigma}(n\vec{s}_1)$

For the right side it holds that

$$\eta^{\rho\Rightarrow\sigma}(n\vec{s}_1) = \lambda^\rho(\eta^\sigma((n\vec{s}_1) \uparrow^1 \eta^\rho(0)))$$

For the left side it holds that

$$\begin{aligned}
&\lambda^\rho_k(\eta^\sigma((n\vec{s}_1)\eta^\rho(k))) \\
&= \lambda^\rho(\eta^\sigma((n\vec{s}_1)\eta^\rho(k))[\nearrow^k_1 \star 0 \star \uparrow^{k+2}]) \\
&= \lambda^\rho(\eta^\sigma(((n\vec{s}_1)\eta^\rho(k))[\nearrow^k_1 \star 0 \star \uparrow^{k+2}])) \quad \text{Lem. 62} \\
&= \lambda^\rho(\eta^\sigma((n\vec{s}_1)[\ldots]\,\eta^\rho(k)[\ldots])) \\
&= \lambda^\rho(\eta^\sigma((n\vec{s}_1) \uparrow^1 \eta^\rho(k)[\nearrow^k_1 \star 0 \star \uparrow^{k+2}])) \text{ Lem. 63 u. } \triangle\mathbf{10} \\
&= \lambda^\rho(\eta^\sigma((n\vec{s}_1) \uparrow^1 \eta^\rho(k[\nearrow^k_1 \star 0 \star \uparrow^{k+2}]))) \text{ Lem. 62} \\
&= \lambda^\rho(\eta^\sigma((n\vec{s}_1) \uparrow^1 \eta^\rho(0)))
\end{aligned}$$

This shows $\mathbf{III.3'}$, hence also $\mathbf{III.3}$.

2. **Case** “BNAbs” : We have $\lambda^\rho r_1 = r_\alpha k$. This leads to falsum and therefore yields the claim.
3. **Case** “BNBeta” : We have BN$(\lambda^\tau r_1(s_1 \star \vec{r}_1), t)$ through

   $\triangle\mathbf{1}$: BN$(r_1[s_1 \star \uparrow^0]\,\vec{r}_1, t)$

   By instantiating r_α we get

   $\triangle\mathbf{2}$: $\lambda^\tau r_1(s_1 \star \vec{r}_1) = r_\alpha k$
   $\triangle\mathbf{3}$: Exp$^\sigma_{\vec{\rho}\star\varepsilon^\rho_k}(t, s)$
   $\triangle\mathbf{4}$: Fr$^{\rho\Rightarrow\sigma}_{\vec{\rho}}(r_\alpha, k)$

   Moreover by the induction hypothesis we additionally have

   $\triangle\mathbf{5}$: $\forall r_\beta.\,(r_1[s_1 \star \uparrow^0]\,\vec{r}_1 = r_\beta k) \to$ Exp$^\sigma_{\vec{\rho}\star\varepsilon^\rho_k}(t, s) \to$
   Fr$^{\rho\Rightarrow\sigma}_{\vec{\rho}}(r_\beta, k) \to$ N$^{\rho\Rightarrow\sigma}_{\vec{\rho}}(r_\beta, \lambda^\rho_k s)$

   We have to show N$^{\rho\Rightarrow\sigma}_{\vec{\rho}}(r_\alpha, \lambda^\rho_k s)$. From $\triangle\mathbf{2}$ it follows for some $\vec{r}_2$ that

   $$s_1 \star \vec{r}_1 = \vec{r}_2 \star k \text{ and } r_\alpha = (\lambda^\tau r_1)\vec{r}_2$$

   We do a case distinction on $\vec{r}_2$:

   (a) **Case** $\vec{r}_2 = ø$: It follows that $\vec{r}_1 = ø$, $s_1 = k$ and $r_\alpha = \lambda^\tau r_1$. This implies

   $\triangle\mathbf{1'}$: BN$(r_1[k \star \uparrow^0], t)$
   $\triangle\mathbf{4'}$: Fr$^{\rho\Rightarrow\sigma}_{\vec{\rho}}(\lambda^\tau r_1, k)$

   From $\triangle\mathbf{4'}$ we get $\tau = \rho$. By lemma 65 and $\triangle\mathbf{1'}$ it follows for some t_1 that

   $\triangle\mathbf{6}$: $t = t_1[k \star \uparrow^0]$
   $\triangle\mathbf{7}$: BN(r_1, t_1)

   We need to show N$^{\rho\Rightarrow\sigma}_{\vec{\rho}}(\lambda^\rho r_1, \lambda^\rho_k s)$. By the definition of N it suffices to show

   I: $\vec{\rho} \vdash \lambda^\rho r_1 : \rho \Rightarrow \sigma$
   II: BN$(\lambda^\rho r_1, \lambda^\rho t_1)$
   III: Exp$^{\rho\Rightarrow\sigma}_{\vec{\rho}}(\lambda^\rho t_1, \lambda^\rho_k s)$

$\mathbf{I}$ follows from $\triangle\mathbf{4}'$. $\mathbf{II}$ follows from $\triangle\mathbf{7}$ and the definition of BN. It remains to show $\mathbf{III}$: From $\triangle\mathbf{4}'$ it follows by lemma 66 that

$\triangle\mathbf{8}$: $\mathrm{Fr}^{\sigma}_{\rho\star\vec{\rho}}(r_1, (k+1))$

From this, $\triangle\mathbf{7}$ and lemma 68 it follows that $\mathrm{Fr}^{\sigma}_{\rho\star\vec{\rho}}(t_1, (k+1))$. By lemma 67 we get

$\triangle\mathbf{9}$: $\lambda^{\rho} t_1 = \lambda^{\rho}_{k}(t_1[k \star \uparrow^0])$

Hence we need to show $\mathrm{Exp}^{\rho\Rightarrow\sigma}_{\vec{\rho}}(\lambda^{\rho}_{k}(t_1[k \star \uparrow^0], \lambda^{\rho}_{k} s)$. By the definition of Exp it suffices to show

$\mathbf{III.1}$: $\vec{\rho} \star \varepsilon^{\rho}_{k} \vdash k : \rho$

$\mathbf{III.2}$: $\mathrm{Exp}^{\sigma}_{\vec{\rho}\star\varepsilon^{\rho}_{k}}(t_1[k \star \uparrow^0], s)$

$\mathbf{III.3}$: $\vec{\rho} \vdash \lambda^{\rho}_{k}(t_1[k \star \uparrow^0]) : \rho \Rightarrow \sigma$

$\mathbf{III.1}$ follows from $\triangle\mathbf{4}'$. $\mathbf{III.2}$ follows from $\triangle\mathbf{3}$ and $\triangle\mathbf{6}$. $\mathbf{III.3}$ we show as follows: $\triangle\mathbf{8}$ entails $(\rho \star \vec{\rho} \vdash r_1 : \sigma)$. By lemma 53 and $\triangle\mathbf{7}$ we get $(\rho \star \vec{\rho} \vdash t_1 : \sigma)$. This implies $(\vec{\rho} \vdash \lambda^{\rho} t_1 : \rho \Rightarrow \sigma)$. Using $\triangle\mathbf{9}$ finally shows $\mathbf{III.3}$.

(b) **Case** $\vec{r}_2 = (r_3 \star \vec{r}_3)$: We have $s_1 = r_3$, $\vec{r}_1 = (\vec{r}_3 \star k)$ and therefore $r_\alpha = (\lambda^{\tau} r_1)(s_1 \star \vec{r}_3)$. Thus we have

$\triangle\mathbf{1}''$: $\mathrm{BN}(r_1[s_1 \star \uparrow^0](\vec{r}_3 \star k), t)$

$\triangle\mathbf{2}''$: $\lambda^{\tau} r_1(s_1 \star \vec{r}_3 \star k) = r_\alpha k$

$\triangle\mathbf{4}''$: $\mathrm{Fr}^{\rho\Rightarrow\sigma}_{\vec{\rho}}(\lambda^{\tau} r_1(s_1 \star \vec{r}_3), k)$

We need to show

$\mathbf{I}$: $\mathrm{N}^{\rho\Rightarrow\sigma}_{\vec{\rho}}(\lambda^{\tau} r_1(s_1 \star \vec{r}_3), \lambda^{\rho}_{k} s)$

Firstly we show $\mathrm{N}^{\rho\Rightarrow\sigma}_{\vec{\rho}}(r_1[s_1 \star \uparrow^0]\vec{r}_3, \lambda^{\rho}_{k} s)$ by means of the induction hypothesis $\triangle\mathbf{5}$, that is we instantiate r_β with $r_1[s_1 \star \uparrow^0]\vec{r}_3$. For this it remains to show $\mathrm{Fr}^{\rho\Rightarrow\sigma}_{\vec{\rho}}(r_1[s_1 \star \uparrow^0]\vec{r}_3, k)$ and for this again it remains to show $(\vec{\rho} \vdash r_1[s_1 \star \uparrow^0]\vec{r}_3 : \rho \Rightarrow \sigma)$. By $\triangle\mathbf{4}''$ it holds that

$\triangle\mathbf{10}$: $\vec{\rho} \vdash \lambda^{\tau} r_1(s_1 \star \vec{r}_3) : \rho \Rightarrow \sigma$

By means of the *Subject Reduction* (lemma 51) we get $(\vec{\rho} \vdash r_1[s_1 \star \uparrow^0]\vec{r}_3 : \rho \Rightarrow \sigma)$. Hence it holds that

$\triangle\mathbf{11}$: $\mathrm{N}^{\rho\Rightarrow\sigma}_{\vec{\rho}}(r_1[s_1 \star \uparrow^0]\vec{r}_3, \lambda^{\rho}_{k} s)$

By inversion on $\triangle\mathbf{11}$ we get

$\triangle\mathbf{12}$: $\mathrm{BN}(r_1[s_1 \star \uparrow^0]\vec{r}_3, t')$

$\triangle\mathbf{13}$: $\mathrm{Exp}^{\rho\Rightarrow\sigma}_{\vec{\rho}}(t', \lambda^{\rho}_{k} s)$

From $\triangle\mathbf{12}$ it follows by the definition of BN that

$\triangle\mathbf{14}$: $\mathrm{BN}(\lambda^{\tau} r_1(s_1 \star \vec{r}_3), t')$

$\mathbf{I}$ follows from $\triangle\mathbf{10}$, $\triangle\mathbf{13}$, $\triangle\mathbf{14}$ and the definition of N. This completes the proof.

□

4.4.3. The proofs of "Axiom 2" to "Axiom 10"

For the proof of "Axiom 2" we introduce an auxiliary predicate Ã, which is implied by A.

Definition 70 (Atwo). *We define* $\tilde{\mathrm{A}} : (\mathbb{L}_{\mathbb{T}}, \mathbb{T}, \Lambda, \Lambda)$ *as*

$$\text{“AtwoIntro”} : \overline{\mathrm{BN}}(\vec{r}, \vec{t}) \to \overline{\mathrm{Exp}}^{\vec{\sigma}}_{\vec{\rho}}(\vec{t}, \vec{s}) \to (\vec{\rho} \vdash n : \vec{\sigma} \twoheadrightarrow \rho) \to \tilde{\mathrm{A}}^{\rho}_{\vec{\rho}}(n\vec{r}, n\vec{s})$$

Lemma 71 (AAtwo). $\mathrm{A}^{\rho}_{\vec{\rho}}(r, s) \to \tilde{\mathrm{A}}^{\rho}_{\vec{\rho}}(r, s)$

Proof: Induction on A. $\square$

Lemma 72 (“Axiom 2”, Ax2). $\boxed{\mathrm{A}^{\iota}_{\vec{\rho}}(r, s) \to \mathrm{N}^{\iota}_{\vec{\rho}}(r, s)}$

Proof: By means of lemma 71 and inversion. $\square$

Remark 73. *Axiom 3 and Axiom 4 follow immediately from the definition of* A.

Lemma 74 (“Axiom 5”, Ax5). $\boxed{\mathrm{Head}(r_1, r_2) \to \mathrm{N}^{\rho}_{\vec{\rho}}(r_2, \dot{s}) \to (\vec{\rho} \vdash r_1 : \rho) \to \mathrm{N}^{\rho}_{\vec{\rho}}(r_1, \dot{s})}$

Proof: Induction on Head and N. $\square$

Lemma 75 (Ax6Aux). $(\theta \Uparrow) \circ (s \star \uparrow^0) = \theta$

Proof: Induction on θ. $\square$

Lemma 76 (“Axiom 6”, Ax6). $\boxed{\mathrm{Head}((\lambda^{\rho} r)[\theta]\, s, r[s \star \theta]\,)}$

Proof: By means of lemma 75 and theorem 14. $\square$

Remark 77. *Note that the proof of “Axiom 6” uses theorem 14 about composition of substitutions, which was the main result of section 2.*

Lemma 78 (“Axiom 7”, Ax7). $\boxed{\mathrm{Head}(r, s) \to \mathrm{Head}(rt, st)}$

Proof: Induction on Head. $\square$

Lemma 79 (HeadTypJ). $\mathrm{Head}(r, s) \to (\vec{\rho} \vdash r : \rho) \to (\vec{\rho} \vdash s : \rho)$

Proof: By means of the *Subject Reduction* (lemma 51). $\square$

Lemma 80 (“Axiom 8”, Ax8). $\boxed{\mathrm{Head}(r, s) \to \mathrm{Fr}^{\rho}_{\vec{\rho}}(r, k) \to \mathrm{Fr}^{\rho}_{\vec{\rho}}(s, k)}$

Proof: By lemma 79 and the definition of Fr. $\square$

The following auxiliary represents the core in the proof of “Axiom 9”. Its proof particularly requires the possibility to *extend* the context in the predicate $\mathrm{Exp}^{\rho}_{\vec{\rho}}(r, s)$. This is allowed by the specific definition of Exp (see definition 46).

Lemma 81 (ExpExtCtx). $\mathrm{Exp}^{\rho}_{\vec{\rho}}(r, s) \to \mathrm{Exp}^{\rho}_{\vec{\rho} \star \vec{\tau}}(r, s)$

Proof: Simultaneous induction on Exp and $\overline{\mathrm{Exp}}$. $\square$

Lemma 82 (“Axiom 9”, Ax9). $\boxed{\mathrm{N}^{\rho}_{\vec{\rho}}(r, \dot{s}) \to \mathrm{N}^{\rho}_{\vec{\rho} \star \vec{\sigma}}(r, \dot{s})}$

Proof: Induction on N. $\square$

Lemma 83 ("Axiom 10", Ax10). $\boxed{\mathrm{A}^{\rho}_{\vec\rho}(r,s) \to \mathrm{A}^{\rho}_{\vec\rho\star\vec\sigma}(r,s)}$

Proof: Induction on A. □

5. Normalization by Evaluation

From the normalization proof of the previous section, the corresponding program can be extracted. As we implemented the proof in the proof system Minlog, this task can be undertaken automatically. We call the extracted program "cNTheorem" in accordance to Minlog's notation.

According to the soundness theorem on program extraction (cf. [4], section 4) cN-Theorem realizes the Normalization Theorem, that is cNTheorem $\underline{\mathbf{r}}$ NTheorem, hence

$$\text{cNTheorem } \underline{\mathbf{r}}\ \forall\vec\rho, r.\ \mathrm{Cor}_{\vec\rho}(r) \to \exists s \mathrm{N}^{\mathrm{Typ}_{\vec\rho}(r)}_{\vec\rho}(r,s)$$

which by the definition of realizability leads to

$$\forall\vec\rho, r.\ \mathrm{Cor}_{\vec\rho}(r) \to \mathrm{N}^{\mathrm{Typ}_{\vec\rho}(r)}_{\vec\rho}(r, \text{cNTheorem}(\vec\rho)(r))$$

That is, for every term r, which is correctly typed in the context $\vec\rho$, cNTheorem$(\vec\rho)(r)$ is indeed the long normal form of r.

It will turn out, that cNTheorem is the well-known algorithm called *Normalization by Evaluation*. This relation was first shown by Berger in [1]. We now introduce the Normalization by Evaluation and then show its accordance to the extracted algorithm cNTheorem.

5.1. Definition of the Normalization by Evaluation

For terms r, which are correctly typed in a context $\vec\rho$, we define the function NbE, which represents the Normalization by Evaluation:

$$\mathrm{NbE}_{\vec\rho}(r) := \ \downharpoonleft^{\mathrm{Typ}_{\vec\rho}(r)} (\llbracket r\rrbracket^{\vec\rho}_{\upharpoonright_{\vec\rho}})(\mathcal{L}(\vec\rho))$$

The role of the upper $\vec\rho$ in the evaluation $\llbracket r\rrbracket^{\vec\rho}_{\upharpoonright_{\vec\rho}}$ under the assignment $\upharpoonright_{\vec\rho}$ will be explained below. In order for the Normalization by Evaluation to be correct, we need in case of $\mathrm{Cor}_{\vec\rho}(r)$ that

$$\mathrm{NbE}_{\vec\rho}(r) = \mathrm{lnf}_{\vec\rho}(r)$$

where $\mathrm{lnf}_{\vec\rho}(r)$ denotes the long normal form of r within the context $\vec\rho$.

As model we choose the universal information system Ω. We then can define the arrows $\downharpoonleft$ and $\upharpoonright$, which are called *reflect* and *reify* respectively. In case of $\downharpoonleft$ we need the possibility to choose a fresh variable in order to perform the η-expansion. That's why in accordance to our model with its $C_\iota := \mathbb{N} \to \Lambda$ the arrows do not operate directly on Λ, but on functions $f : \mathbb{N} \to \Lambda$.

Remark 84. *Due to the "de Bruijn" notation we don't need term families, which are used in the usual notation to ensure that variables, which are supposed to be fresh, don't already appear as bound variables.*

Definition 85. *We define simultaneously* $\downharpoonleft \colon \Omega \to \mathbb{N} \to \Lambda$ *and* $\upharpoonleft \colon (\mathbb{N} \to \Lambda) \to \Omega$ *by*

$$\boxed{\begin{aligned} \downharpoonleft^{\iota} (a)(k) &:= a(k) \\ \downharpoonleft^{\rho \Rightarrow \sigma} (a)(k) &:= \lambda^{\rho}_{k} \downharpoonleft^{\sigma} (a \upharpoonleft^{\rho} (\overline{k}))(k+1) \\ \\ \upharpoonleft^{\iota} (f) &:= f \\ \upharpoonleft^{\rho \Rightarrow \sigma} (f(k))(b) &:= \upharpoonleft^{\sigma} (f(k) \downharpoonleft^{\rho} (b)(k)) \end{aligned}}$$

Remark 86.

1. $\overline{k}$ *denotes the constant function* k. *Provided that* k *is great enough,* $\overline{k}$ *plays the role of a fresh variable. In* $\mathrm{NbE}_{\vec{\rho}}(r)$ *the arrow* $\downharpoonleft$ *is initialized with such a* k, *namely* $k = \mathcal{L}(\vec{\rho})$.
2. *For better readability the administrative functions are not given explicitly here. The notation* $a(k)$ *is in fact a shortform of* $\mathrm{ModIota}(a)(k)$ *and* $a \upharpoonleft^{\rho} (\overline{k})$ *of* $\mathrm{Mod}(a)(\upharpoonleft^{\rho} (\overline{k}))$. *Likewise the value* f *is a shortform of* $\mathrm{HatIota}(f)$ *and* $\upharpoonleft^{\sigma} (f(k) \downharpoonleft^{\rho} (b)(k))$ *of* $\mathrm{Hat}_{\rho,\sigma}(\upharpoonleft^{\sigma} (f(k) \downharpoonleft^{\rho} (b)(k)))$.

Via $\upharpoonleft^{\rho}$ we are able to define the assignment $\upharpoonleft_{\vec{\rho}}$. We introduce it in accordance to the "de Bruijn" notation as lists of type $\mathbb{L}_{\Omega}$.

Definition 87. *We define* $\upharpoonleft \colon \mathbb{L}_{\mathbb{T}} \to \mathbb{L}_{\Omega}$ *as*

$$\boxed{\upharpoonleft_{\vec{\rho}} := \upharpoonleft^{\rho_0} (\overline{0}) \star \upharpoonleft^{\rho_1} (\overline{1}) \star \ldots \star \upharpoonleft^{\rho_{\mathcal{L}(\vec{\rho})-1}} (\overline{\mathcal{L}(\vec{\rho}) - 1})}$$

For the empty list we define $\upharpoonleft_{\varnothing} := \varnothing$. The type ρ_i denotes the ith element within $\vec{\rho}$. The notation $\overline{n}$ means as before the constant function n. Next we give the evaluation function $[\![r]\!]^{\vec{\rho}}_{\vec{a}}$, where $\vec{a}$ plays the role of an assignment:

Definition 88. *We define* $[\![\ ]\!] : \Lambda \to \mathbb{L}_{\mathbb{T}} \to \mathbb{L}_{\Omega} \to \Omega$ *as*

$$\boxed{\begin{aligned} [\![k]\!]^{\vec{\rho}}_{\vec{a}} &:= k \gtrdot \vec{a} \\ [\![rs]\!]^{\vec{\rho}}_{\vec{a}} &:= [\![r]\!]^{\vec{\rho}}_{\vec{a}} [\![s]\!]^{\vec{\rho}}_{\vec{a}} \\ [\![\lambda^{\rho} r]\!]^{\vec{\rho}}_{\vec{a}}(b) &:= [\![r]\!]^{\rho \star \vec{\rho}}_{b \star \vec{a}} \end{aligned}}$$

Remark 89. *Again the administrative functions are left out for better readability. The application case is to be understood as* $\mathrm{Mod}([\![r]\!]^{\vec{\rho}}_{\vec{a}})([\![s]\!]^{\vec{\rho}}_{\vec{a}})$. *The abstraction case reads in detail as* $\mathrm{Hat}_{\rho,\mathrm{Typ}_{\rho \star \vec{\rho}}(r)}([\![r]\!]^{\rho \star \vec{\rho}}_{b \star \vec{a}})$. *From this it gets clear, why the definition of* $[\![\ ]\!]$ *needs to have a context* $\vec{\rho}$.

We are now able to define the Normalization by Evaluation:

Definition 90. *We define* $\mathrm{NbE} : \mathbb{L}_{\mathbb{T}} \to \Lambda \to \Lambda$ *as*

$$\boxed{\mathrm{NbE}_{\vec{\rho}}(r) := \downharpoonleft^{\mathrm{Typ}_{\vec{\rho}}(r)} ([\![r]\!]^{\vec{\rho}}_{\upharpoonleft_{\vec{\rho}}})(\mathcal{L}(\vec{\rho}))}$$

It remains to show that Normalization by Evaluation is sound, that is we need to prove $\mathrm{NbE}_{\vec{\rho}}(r) = \mathrm{lnf}_{\vec{\rho}}(r)$ for $\mathrm{Cor}_{\vec{\rho}}(r)$. This could be done by induction on the syntactical structure of r (cf. [16]). However we can skip this step here, as in the next section we will show that $\mathrm{NbE}_{\vec{\rho}}(r)$ matches exactly the extracted program from the normalization proof, which computes the long normal form as we already know.

5.2. *The Extracted Program of the Normalization Proof*

The entire extracted program can be divided into the extracted parts corresponding to lemma 1, lemma 2, lemma 3, the auxiliary SCrsSeq and finally the NTheorem. In accordance to Minlog's notation we call them "cLemmaOne", "cLemmaTwo", "cLemmaThree", "cSCrsSeq" and "cNTheorem" respectively.

The following table gives the symbols of Minlog's output and the corresponding notation used in the text. As already mentioned all about the program extraction, which is used here, can be found in detail in [4] in section 4. In order to distinguish more easily between the simply typed λ-calculus from the normalization proof and the λ-calculus from the Curry-Howard-Correspondance the $\underline{\lambda}$ of the latter is underlined.

Explanation	**Symbol**	**Minlog's Output**
λ-abstraction:	$\underline{\lambda} x.M$	`([x]M)`
pair:	$\langle M \mid N \rangle$	`(M@N)`
left element of a pair:	$(M\ \underline{0})$	`left M`
right element of a pair:	$(M\ \underline{1})$	`right M`
arrow for types:	$\Rightarrow$	`=>`
product for types:	$\times$	`@@`
recursion operator:	$\mathcal{R}$	`Rec`

Besides we use `p`, `q` and `d` respectively for functions of typ `(omega=>nat=>term)@@((nat=>term)=>omega)`, `list type=>list omega=>omega` and `list type=>list omega` respectively.

5.2.1. *The Realizers of the* SC-*Lemmas*

If we had defined the predicate SC in its original version by means of axioms, labels for it would have appeared in the extracted program for which we had no realizers. Hence we could not get a complete extracted program. However by the redefinition through the predicate SC**r** we *do* get such realizers. They turn out to be exactly the administrative functions ModIota, HatIota, Mod and Hat from section 3.

5.2.2. *The Extracted Program for Lemma 1*

“Lemma 1” reads as follows:

$$\forall\rho\forall^{\mathrm{nc}}\vec{\rho}, r.\ (\vec{\rho}\vdash r:\rho)\to(\mathrm{SC}^{\rho}_{\vec{\rho}}(r)\to\mathrm{FN}^{\rho}_{\vec{\rho}}(r))\wedge(\mathrm{FA}^{\rho}_{\vec{\rho}}(r)\to\mathrm{SC}^{\rho}_{\vec{\rho}}(r))$$

The corresponding extracted program “cLemmaOne” is:

```
(Rec type=>(omega=>nat=>term)@@((nat=>term)=>omega))
(ModIota@OmegaInIota)
([rho3,rho4,p5,p6]
  ([a7,n8] Abs rho3
    (Sub(left p6(Mod a7(right p5([n9]Var n8)))(Succ n8))
      (Wrap(Succ(Succ n8))((Var map Seq 1 n8):+:(Var 0):))))@
  ([g7]Hat rho3 rho4([a8]right p6([n9]g7 n9(left p5 a8 n9)))))
```

We translate this into the notation used previously in the text. For better readability we keep the denotations for `left` and `right`. For the administrative functions we use again shortforms. Variables are renamed in an obvious way. Furthermore the abbreviation $\lambda^{\rho}_{k}r = \lambda^{\rho}r[\nearrow^{k}_{\overline{1}}\ \star 0 \star\ \uparrow^{k+2}]$ introduced earlier is used. The constant function $\underline{\lambda}m.k$ can be written as $\overline{k}$. Thus we get for cLemmaOne:

$$\begin{aligned}
&\mathcal{R}:\mathbb{T}\to(\Omega\to\mathbb{N}\to\Lambda)\times(\mathbb{N}\to\Lambda)\to\Omega\\
&\langle\underline{\lambda}a,k.\ a(k)\mid\underline{\lambda}f.\ f\rangle\\
&(\underline{\lambda}\rho,\sigma,p_1,p_2.\\
&\quad\langle\underline{\lambda}a,k.\ \lambda^{\rho}_{k}\ \text{left}\ p_2(a\ \text{right}\ p_1(\overline{k}))(k+1)\mid\\
&\quad\underline{\lambda}f,b.\ \text{right}\ p_2(\underline{\lambda}k.\ f(k)(\text{left}\ p_1(b)(k)))\rangle)
\end{aligned}$$

Lemma 91. cLemmaOne$(\rho)=\langle\downharpoonleft^{\rho}\mid\upharpoonleft^{\rho}\rangle$

Proof: Induction on ρ. $\square$

5.2.3. *The Extracted Program for Lemma 2*

“Lemma 2” reads as follows:

$$\forall^{\mathrm{nc}}\rho,\vec{\rho},r,s.(\vec{\rho}\vdash r:\rho)\to\mathrm{SC}^{\rho}_{\vec{\rho}}(s)\to\mathrm{Head}(r,s)\to\mathrm{SC}^{\rho}_{\vec{\rho}}(r)$$

The corresponding extracted program “cLemmaTwo” is:

```
[a0]a0
```

This result could be achieved by the specific definition of SC through SC**r**. Also the nc-quantifiers, supported by the proof system Minlog, play a vital role here. They could be used because SC$\mathbf{r}^{\rho}_{\vec{\rho}}$ unlike SC$^{\rho}_{\vec{\rho}}$ has *no* computational content.

5.2.4. *The Extracted Program for Lemma 3*

“Lemma 3” reads as follows:

$$\forall r, \vec{\rho} \forall^{\mathrm{nc}} \vec{\sigma}, \rho, \vec{s}.\ (\vec{\rho} \vdash r : \rho) \to \overline{\mathrm{SC}}^{\vec{\rho}}_{\vec{\sigma}}(\vec{s}) \to \mathrm{SC}^{\rho}_{\vec{\sigma}}(r[\vec{s} \star \ \uparrow^0])$$

The corresponding extracted program “cLemmaThree” is:

```
(Rec term=>list type=>list omega=>omega)
([n3,rhos4](ListRef omega)n3)
([r3,r4,q5,q6,rhos7,as8]Mod(q5 rhos7 as8)(q6 rhos7 as8))
([rho3,r4,q5,rhos6,as7] Hat rho3(Typ(rho3::rhos6)r4)
  ([a8]cLemmaTwo(q5(rho3::rhos6)(a8::as7))))
```

For the translation we proceed as before. Note that we can resolve cLemmaTwo as the identity. The function `ListRef` is the function “$\gg$” which picks the corresponding element from a list. Thus we get for cLemmaThree:

$$\begin{array}{l}\mathcal{R} : \Lambda \to \mathbb{L}_{\mathbb{T}} \to \mathbb{L}_{\Omega} \to \Omega \\ (\underline{\lambda} k, \vec{\rho}.\ k \gg) \\ (\underline{\lambda} r, s, q_1, q_2, \vec{\rho}, \vec{a}.\ (q_1 \ \ \vec{\rho} \ \ \vec{a})(q_2 \ \ \vec{\rho} \ \ \vec{a})) \\ (\underline{\lambda} \rho, r, q, \vec{\rho}, \vec{a}.\ \underline{\lambda} b(q \ \ (\rho \star \vec{\rho}) \ \ (b \star \vec{a})))\end{array}$$

Lemma 92. cLemmaThree$(r)(\vec{\rho})(\vec{a}) = [\![r]\!]^{\vec{\rho}}_{\vec{a}}$

Proof: Induction on r. □

5.2.5. *The Extracted Program for Lemma “SCrsSeq”*

The auxiliary “SCrsSeq” (lemma 39) reads as follows:

$$\forall \vec{\rho}, \vec{\sigma}.\ \overline{\mathrm{SC}}^{\vec{\rho}}_{\vec{\sigma} \star \vec{\rho}}(\nearrow^{\mathcal{L}(\vec{\rho})}_{\mathcal{L}(\vec{\sigma})})$$

The corresponding extracted program “cSCrsSeq” is:

```
(Rec list type=>list type=>list omega)
([rhos2](Nil omega))
([rho2,rhos3,d4,rhos5]
  right(cLemmaOne rho2)([n6]Var Lh rhos5)::d4(rhos5:+:rho2:))
```

For the translation we can resolve the right element of cLemmaOne in accordance with the above lemma as $\upharpoonleft$. Furthermore `([n6]Var Lh rhos5)` can be written as a constant sequence. Thus we get:

$$\begin{array}{l}\mathcal{R} : \mathbb{L}_{\mathbb{T}} \to \mathbb{L}_{\mathbb{T}} \to \mathbb{L}_{\Omega} \\ (\underline{\lambda} \vec{\sigma}.\ \emptyset) \\ (\underline{\lambda} \rho, \vec{\rho}, d, \vec{\sigma}.\ (\upharpoonleft^{\rho} \ (\overline{\mathcal{L}(\vec{\sigma})})) \star d(\vec{\sigma} \star \rho))\end{array}$$

To show that (cSCrsSeq $\vec{\rho}$ $\emptyset$) is the assignment $\upharpoonleft_{\vec{\rho}}$ we need a more general lemma. For this we define the generalization $\upharpoonleft^{\vec{\sigma}}_{\vec{\rho}}$ of the assignment $\upharpoonleft_{\vec{\rho}}$:

Definition 93. *We define* $\upharpoonleft : \mathbb{L}_{\mathbb{T}} \to \mathbb{L}_{\mathbb{T}} \to \mathbb{L}_{\Omega}$ *as*

$$\upharpoonleft^{\vec{\sigma}}_{\vec{\rho}} := \upharpoonleft^{\rho_0} \ (\overline{\mathcal{L}(\vec{\sigma})}) \star \ \upharpoonleft^{\rho_1} \ (\overline{\mathcal{L}(\vec{\sigma})+1}) \star \ \ldots \ \star \ \upharpoonleft^{\rho_{\mathcal{L}(\vec{\rho})-1}} \ (\overline{\mathcal{L}(\vec{\sigma}) + (\mathcal{L}(\vec{\rho}) - 1)})$$

$\upharpoonleft^{\vec{\sigma}}_{\text{ø}}$ is again defined as ø.

Lemma 94. cSCrsSeq($\vec{\rho}$)($\vec{\sigma}$) $=\upharpoonleft^{\vec{\sigma}}_{\vec{\rho}}$

Proof: Induction on $\vec{\rho}$. □

Lemma 95. $\boxed{\text{cSCrsSeq}(\vec{\rho})(\text{ø}) =\upharpoonleft_{\vec{\rho}}}$

Proof: With the above lemma 94 and $\upharpoonleft^{\text{ø}}_{\vec{\rho}}=\upharpoonleft_{\vec{\rho}}$. □

5.2.6. The Extracted Program of the Normalization Theorem

The Normalization Theorem reads as follows:

$$\forall \vec{\rho}, r. \ \text{Cor}_{\vec{\rho}}(r) \to \exists s \text{N}^{\text{Typ}_{\vec{\rho}}(r)}_{\vec{\rho}}(r, s)$$

The corresponding extracted program "cNTheorem" is:

```
[rhos0,r1]
  left(cLemmaOne(Typ rhos0 r1))
  (cLemmaThree r1 rhos0(cSCrsSeq rhos0(Nil type))) Lh rhos0
```

With the preceding results we get $\downharpoonleft$ for the left element of `cLemmaOne`, the evaluation function $[\![\ ]\!]$ for `cLemmaThree`, and $\upharpoonleft_{\vec{\rho}}$ for `cSCrsSeq` with parameters $\vec{\rho}$ and ø. Thus we finally get for cNTheorem:

$$\underline{\lambda} \vec{\rho}, r. \ \downharpoonleft^{\text{Typ}_{\vec{\rho}}(r)} \ ([\![r]\!]^{\vec{\rho}}_{\upharpoonleft_{\vec{\rho}}})(\mathcal{L}(\vec{\rho}))$$

From this it immediately follows that

$$\boxed{\text{cNTheorem}(\vec{\rho})(r) = \text{NbE}_{\vec{\rho}}(r)}$$

Hence we have finally shown that the Normalization by Evaluation is sound and that the extracted program of our normalization proof is exactly this Normalization by Evaluation.

References

[1] Ulrich Berger. Program extraction from normalization proofs. In M. Bezem and J.F. Groote, editors, *Typed Lambda Calculi and Applications*, volume 664 of *LNCS*, pages 91–106. Springer Verlag, Berlin, Heidelberg, New York, 1993.

[2] U. Berger, S. Berghofer, P. Letouzey, and H. Schwichtenberg. Program extraction from normalization proofs. *Studia Logica*, 2005. Special issue, to appear.

[3] Dominik Schlenker. Programmextraktion aus einem Normalisierungsbeweis für den einfach getypten λ-Kalkül. Master's thesis, LMU München, 2005. http://www.math.lmu.de/∼schlenke/.

[4] Helmut Schwichtenberg. Minimal logic for computable functionals, 2005. Mathematisches Institut der Universität München.

[5] Thierry Coquand and Peter Dybjer. Intuitionistic model constructions and normalization proofs. *Mathematical Structures in Computer Science*, 7:73–94, 1997.

[6] Catarina Coquand. From semantics to rules: A machine assisted analysis. In E. Börger, Y. Gurevich, and K. Meinke, editors, *Computer Science Logic, 7th Workshop, Swansea 1993*, volume 832 of *LNCS*, pages 91–105. Springer Verlag, Berlin, Heidelberg, New York, 1994.

[7] Thorsten Altenkirch. Proving strong normalization of CC by modifying realizability semantics. In H. Barendregt and T. Nipkow, editors, *Types for Proofs and Programs. International Workshop TYPES '93. Nijmegen, The Netherlands, May 1993.*, volume 806 of *LNCS*, pages 3–18. Springer Verlag, Berlin, Heidelberg, New York, 1994.

[8] Thorsten Altenkirch, Martin Hofmann, and Thomas Streicher. Reduction-free normalization for a polymorphic system. In *LICS'96*, 1996.

[9] Thorsten Altenkirch, Peter Dybjer, Martin Hofmann, and Phil Scott. Normalization by evaluation for typed lambda calculus with coproducts. In *LICS '01: Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science*, pages 203–210, Washington, DC, USA, 2001. IEEE Computer Society.

[10] Malgorzata Biernacka, Olivier Danvy, and Kristian Stovring. Program extraction from weak head normalization proofs. Preliminary proceedings of MFPS XXI, Birmingham, UK, 2005.

[11] The Minlog System - http://www.minlog-system.de/.

[12] Helmut Schwichtenberg. Minlog reference manual, 2004. Mathematisches Institut der Universität München.

[13] Felix Joachimski. *Reduction Properties of* Π*IE-Systems*. PhD thesis, LMU München, 2001.

[14] Dana Scott. Domains for denotational semantics. In E. Nielsen and E.M. Schmidt, editors, *Automata, Languages and Programming*, volume 140 of *LNCS*, pages 577–613. Springer Verlag, Berlin, Heidelberg, New York, 1982. A corrected and expanded version of a paper prepared for ICALP'82, Aarhus, Denmark.

[15] Helmut Schwichtenberg. Recursion on the partial continuous functionals. In C. Dimitracopoulos, L. Newelski, D. Normann, and J. Steel, editors, *Logic Colloquium '05*, volume 28 of *Lecture Notes in Logic*, pages 173–201. Association for Symbolic Logic, 2006. http://www.math.lmu.de/∼schwicht/papers/athen05/total06.pdf.

[16] Ulrich Berger, Matthias Eberl, and Helmut Schwichtenberg. Term rewriting for normalization by evaluation. *Information and Computation*, 183:19–42, 2003.