Formalising Strong Normalisation proofs of Explicit Substitution Calculi in ALF[†]

Fairouz Kamareddine (fairouz@macs.hw.ac.uk)
Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK

Haiyan Qiao (qiao@cs.chalmers.se)
Computing Science, Chalmers University of Technology, Göteborg, Sweden

Abstract. The past decade has given rise to a number of explicit substitution (ES) calculi. An important question of explicit substitution calculi is that of the termination of the underlying calculus of substitution. Proofs of termination of substitutions fall under two categories. Those that are easy because a decreasing measure can be established and those that are difficult because such a decreasing measure is not easy to establish. This paper considers two styles of explicit substitution: σ and s, for which different termination proof methods apply. The termination of s is guaranteed by a decreasing weight, while a decreasing weight for showing the termination of σ has not yet been found. These termination methods for σ and s are formalised in the proof checker ALF. During our process of formally checking the termination of σ and s we comment on what is needed to make a proof formally checkable.

1. Introduction

What is explicit substitution? The classical λ-calculus deals with substitution in an implicit way. This means that the computations to perform substitution are usually described with operators which do not belong to the language of the λ-calculus. There has however been an interest in formalising substitution explicitly in order to provide a theoretical framework for the implementation of programming languages and theorem provers. Several calculi including new operators to denote substitution and new rules to handle these operators have been proposed (e.g., [10, 2, 17, 30, 4, 5, 21, 22, 28, 13]). Amongst these calculi we mention $C\lambda\xi\phi$ (cf. [14]); the calculi of categorical combinators (cf. [10]); $\lambda\sigma$, $\lambda\sigma_{\Uparrow}$, $\lambda\sigma_{SP}$ (cf. [2, 11, 30]) referred to as the $\lambda\sigma$-family; $\varphi\sigma BLT$ (cf. [20]); $\lambda\upsilon$ (cf. [4]) and $\lambda\zeta$ (cf. [28]) which are descendants of the $\lambda\sigma$-family; λs (cf. [21]) and λs_e (cf. [22]). Most of these calculi are described in de Bruijn notation and can roughly be classified under two styles: the $\lambda\sigma$ [2, 17] and the λs styles [21, 22]. The new symbols added by explicit substitution calculi to represent operations related to substitutions, and the new rules explaining how these operations work

[†] We are grafetful for the useful comments we received from Martin Hoffmann, Claude Kirchner, Joe Wells and the anonymous referees.

complicate the rewriting system behind the λ-calculus and the question whether these new rules terminate arises. This paper concentrates on the proofs of termination of these extra rules in both the $\lambda\sigma$ and λs calculi. $\lambda\sigma$ and λs are chosen because they differ in style, and because their proofs of termination are different: the set s of new rules of λs is shown to terminate via a decreasing weight whereas no decreasing weight has yet been found for the set σ of new rules of $\lambda\sigma$.

Why explicit substitutions? Explicit substitutions allow more flexibility in ordering work. Propagating substitutions through a particular subterm can wait until the subterm is the focus of computation. This allows all of these substitutions to be done at once, thus improving locality of reference. This flexibility also allows postponing unneeded work indefinitely (i.e., avoiding it completely). This can yield profits, since implicit substitution can be an inefficient, maybe even exploding, process by the many repetitions it causes. Another benefit is that explicit substitution allows formal modelling of the techniques used in real implementations, e.g., environments [18]. Furthermore, as the implementation of substitution in many theorem provers is not based on a formal system, it is not clear what properties their underlying substitution has. Thus, it helps to have a choice of explicit substitution systems whose properties have already been established. This is witnessed by the theorem prover ALF, which is based on Martin-Löf's type theory with explicit substitutions [24]. Another justification for explicit substitution in theorem proving is the belief that "tactics" can be replaced by incomplete proofs, which need explicit substitutions [28, 24].

Why formalise proofs in a proof checker? The past thirty years have seen much work on formalising proofs from paper into a proof checker (e.g., [9, 27, 31, 3]). Pioneering work on this started in 1967 with de Bruijn's influential proof checker Automath. Since, many proof checkers have been built [7, 15, 24, 29, 8] into which many proofs have been formalised. Formalisation in a proof checker is useful even if the proof on paper is fully trusted and correct. Reasons for this include:

- Some complex proofs may be unconvincing unless they are checked by a proof checker.[1]
- Formalisation in a proof checker enables the building of a library of readily available proofs that can be used in different situations and can relieve us from repeating the same proofs over and over.
- Formalisation helps one to actually find the proof of certain difficult theorems that would have been hard to solve just on paper.

This paper, adds one further case study to formalisations in proof checkers. We formalise in ALF [24] the termination of the explicit substitutions calculi s and σ. Our reasons for doing this include:

[1] We stress that all the proofs that we formalise in this paper are fully trusted.

- The development of so many calculi of explicit substitutions and the intricacies involved in proving their properties call for methods that one can take of the shelf and use for newly developed calculi.
- As most calculi with explicit substitutions have rules that are complex and properties whose proofs are intricate, it is necessary to formalise these calculi and check the proofs in some proof checker.

These reasons hold in particular for the proofs of termination (a basic property) of explicit substitutions. There are various ways to show termination of explicit substitutions, but the proofs can be very intricate. Termination proofs of explicit substitutions fall under the following:

1. A decreasing weight can be found, i.e. when terms are reduced, their weights decrease. Examples of substitutions that have this property include υ [4], s [21], and $\sigma_{\Uparrow}$ [17].
2. A strict (reduction preserving) translation from the calculus we wish to show terminating to another calculus known to be terminating can be found. E.g. the termination of s can be obtained by a strict translation from s to σ which is known to be terminating.
3. By finding an induction argument when neither 1 nor 2 above apply, this is, for example, the way σ is shown to be terminating in [12].

In this paper we formalise in ALF, termination proofs for σ and s that fall under 1, 2 and 3 above. There has been previous work on formalising properties of explicit substitutions. For example, Lescanne formalised in Coq, the substitution lemma of $\lambda\upsilon$ of [4], and Saïbi formalised $\lambda\sigma_{\Uparrow}$ [17], which is an extension of σ with meta-variables, and proved confluence of $\lambda\sigma_{\Uparrow}$ and strong normalisation of $\sigma_{\Uparrow}$ by finding a decreasing weight, see [31]. Our work concentrates on different methods of termination of explicit substitutions and considers two different calculi σ and s.

There has been a lot of work on termination of explicit substitutions recently (e.g., [5, 13, 30, 16, 12, 34, 33]). Our reasons for choosing the proof of [12] and the proof checker ALF include:

- The interesting proof of [12] does not obey a decreasing weight. Moreover, the proof of [12] is very intricate and as far as we know, this is the first formalisation of such an intricate method.
- The proof of [12] had interesting concepts such as its formulation of a calculus of contexts. That formalisation which is basic to the proof of [12], leads to a calculus of contexts that is in line with [6]. It may also lead to ways of formalising other extensions as in [26].

It should be noted that we did not choose an easy proof to formalise. There are other proofs of termination of the same calculus (e.g., the proof of [34] is much simpler to formalise than that of [12]). But, a full mechanical check of a complicated proof (which we do in this paper) can be considered as a valuable achievement in itself. We will formalise and implement the whole proof of [12] including the context calculus in ALF. It would be interesting in the future to attempt and use this work for proof checking other proofs of termination such as that of CCL [16]. That proof would be particularly informative in this context as it was obtained by considering a lot of weights in an unusual manner.

The paper is organised as follows. In Section 2 we recall $\lambda\sigma$ and its termination proof. In Section 3 we recall λs and present three different termination proofs, two of which are new. Section 4 is a brief introduction to ALF and to Martin-Löf type theory. In Section 5 we give formalisations of explicit substitution calculi, their termination and the lexicographic order induction principle. In Section 6 we formalise all termination proofs of s presented in this paper. In Section 7 we formalise a context calculus which is the main part of the termination proof of σ. In Section 8 the termination proof of σ is formalised.

2. The calculus of $\lambda\sigma$ and the termination proof of σ

$\lambda\sigma$ provides a setting for explicit substitutions, with pleasant properties. It is strongly connected with the categorical understanding of the λ-calculus, where a substitution is interpreted as a composition [10]. In this section we present the $\lambda\sigma$-calculus and some of its properties.

2.1. Definition of the calculus $\lambda\sigma$

In explicit substitution calculi, substitutions are delayed and explicitly recorded; the application of substitutions is independent, and not coupled with the β-rule. If a is a term and s is a substitution then the term $a[s]$, which is called a closure, represents a with the substitution s. When substitutions are made explicit, the β-rule with delayed substitutions, called $Beta_v$ can be expressed by: $(\lambda x.a)b \rightarrow_{Beta_v} a[(b/x)\cdot id]$ where $(b/x)\cdot id$ is the substitution that replaces x with b and affects no other variables (id is the identity substitution).

We assume familiarity with de Bruijn indices and use substitution calculi with these indices (as mostly done in explicit substitution).

DEFINITION 1 (The $\lambda\sigma$-calculus). *The $\lambda\sigma$-calculus is a two-sorted calculus where the set of terms $\Lambda\sigma^t$ is given by the abstract syntax $a, b ::= 1 \mid ab \mid \lambda a \mid a[s]$ and the set of substitutions $\Lambda\sigma^s$ is given by the abstract syntax $s, t ::= id \mid \uparrow \mid s \cdot t \mid s \circ t$.*
The set σ of the rules which propagate the substitutions is given in Figure 2.1. $\lambda\sigma$ is the union of the σ rules and (Beta): $(\lambda a)b \rightarrow a[b\cdot id]$.[2]

[2] $Beta$ eliminates λ's creating substitutions; the σ rules eliminate substitutions.

(VrId)	$1[id] \to 1$	(App)	$(ab)[s] \to (a[s])(b[s])$
(VrCons)	$1[a \cdot s] \to a$	(Abs)	$(\lambda a)[s] \to \lambda(a[1 \cdot (s \circ \uparrow)])$
(IdL)	$id \circ s \to s$	(Clos)	$(a[s])[t] \to a[s \circ t]$
(ShId)	$\uparrow \circ id \to \uparrow$	(Map)	$(a \cdot s) \circ t \to a[t] \cdot (s \circ t)$
(ShCons)	$\uparrow \circ (a \cdot s) \to s$	(Ass)	$(s_1 \circ s_2) \circ s_3 \to s_1 \circ (s_2 \circ s_3)$

Figure 1. The σ-Rules

If s represents the infinite substitution $\{a_1/1, a_2/2, a_3/3, \cdots\}$, then the syntax of substitutions can be described intuitively as follows:

- id is the identity substitution $\{i/i\}$ (for all i).
- $\uparrow$ is the substitution $\{(i+1)/i\}$ (for all i). E.g., $1[\uparrow] = 2$. Thus, $n+1$ can be encoded as $1[\uparrow^n]$, where $\uparrow^n$ is the composition: $\uparrow \circ \cdots \circ \uparrow$.
- For all i, $i[s]$ is the value of the de Bruijn index i in the substitution s, also written $s(i)$ when s is viewed as a function.
- $a \cdot s$ is the substitution $\{a/1, s(i)/(i+1)\}$ (for all i). E.g., $a \cdot id = \{a/1, 1/2, 2/3, \cdots\}$ and $1 \cdot \uparrow = \{1/1, \uparrow(1)/2, \uparrow(2)/3, \cdots\} = id$.
- $s \circ t$ (the composition of s and t) is such that $a[s \circ t] = a[s][t]$, hence $s \circ t = \{s(i)/i\} \circ t = \{s(i)[t]/i\}$ (for all i).

2.2. σ_0: A VARIANT OF σ, AND THE PROOFS OF TERMINATION

We discuss the strong normalisation (termination/noetherianity) of σ. We start with the statement of the theorem that will be proof checked:

THEOREM 2 (SN of σ). *The calculus σ is strongly normalising.*

There are various proofs of this theorem in the literature:

1. The first strong normalisation proof of σ is based on the strong normalisation of $SUBST$ [16], which is, within CCL, the set of rewriting rules that compute the substitutions. See [16].
2. The proof in [12] shows the termination of σ via a strict translation from σ to another calculus σ_0 (Lemma 6) and the termination of σ_0 (Theorem 7). This proof is given in detail in Section 2.3 as it is the one that we shall formalise in this paper.
3. Zantema gives two proofs in [33, 34]. The first is based on a suitable generalisation of polynomial orders to show the termination of the calculus σ_0 given below (and hence the termination of σ). The second uses semantic labelling to show the termination of σ.

As the proof of [12] is given via the strong normalisation of σ_0 (an economic variant of σ), and a strict translation from σ to σ_0, we give the definition of σ_0. The calculus σ_0 is one sorted and treats both $\circ$ and $[\,]$ as $\circ$, observing that $\circ$ and $[\,]$ behave in the same way.

DEFINITION 3 (The σ_0-calculus). *The set of terms $\Lambda\sigma_0$ of the σ_0-calculus has the abstract syntax $s, t ::= 1 \mid id \mid \uparrow \mid \lambda s \mid s \circ t \mid s \cdot t$.*
The set, denoted σ_0, of rules of the calculus is the following:

(VrId)	$1 \circ id \to 1$	*(ShId)*	$\uparrow \circ id \to \uparrow$
(VrCons)	$1 \circ (s \cdot t) \to s$	*(Abs)*	$(\lambda s) \circ t \to \lambda(s \circ (1 \cdot (t \circ \uparrow)))$
(ShCons)	$\uparrow \circ (s \cdot t) \to t$	*(Map)*	$(s \cdot t) \circ u \to (s \circ u) \cdot (t \circ u)$
(IdL)	$id \circ s \to s$	*(Ass)*	$(s \circ t) \circ u \to s \circ (t \circ u)$

REMARK 4. *σ_0 is a particular case of the system Subst of CCL. Rules $(VrId)$ and $(ShId)$ are particular cases of the right identity rule.*

We shall often interpret a calculus C_1 into another calculus C_2. We call *strict interpretation* a function which maps a reduction step of C_1 into one or many reduction steps in C_2. Termination of C_2 and the existence of a strict interpretation of C_1 into C_2 yield termination of C_1. The interpretation function from $\Lambda\sigma$ to $\Lambda\sigma_0$ is given by the following:

DEFINITION 5 (Interpreting $\Lambda\sigma$ in $\Lambda\sigma_0$). *Let $F : \Lambda\sigma \to \Lambda\sigma_0$ be:*

$F(1) = 1$	$F(ab) = F(a) \cdot F(b)$	$F(\lambda a) = \lambda(F(a))$
$F(\uparrow) = \uparrow$	$F(a \cdot s) = F(a) \cdot F(s)$	$F(a[s]) = F(a) \circ F(s)$
$F(id) = id$	$F(s \circ t) = F(s) \circ F(t)$	

Then we have the following lemma that was easily checked in ALF:

LEMMA 6 (F preserves reductions). *If $a \to_\sigma b$ then $F(a) \to_{\sigma_0} F(b)$.*

Of course, with Lemma 6, it is enough to show the termination of σ_0 in order to guarantee the termination of σ. Hence the next theorem:

THEOREM 7 (SN of σ_0). *The calculus σ_0 is strongly normalising.*

2.3. The proof of [12]

[12] notes that it is easy to define an R.P.O to show the termination of $\sigma_0 - \{(Abs)\}$ but that it was not possible to extend this R.P.O. to all of σ_0. Hence, they prove that those terms which do not contain λs terminate. For this, they start by the following definition:

DEFINITION 8 (W-terms and L-terms). *In σ_0 a term is called a W-term if no λ occurs in it. Otherwise it is called a L-term.*

It is obvious that in σ_0, a term is either a W-term or an L-term.

Now, the following lemma is shown in [12] (SN is the set of strongly normalising terms in $\Lambda\sigma_0$):

LEMMA 9 (SN of W-terms).

1. *For $s \in \Lambda\sigma_0$, $\lambda s \in \mathrm{SN}$ iff $s \in \mathrm{SN}$.*
2. *For $s, t \in \Lambda\sigma_0$, $s \cdot t \in \mathrm{SN}$ iff both $s \in \mathrm{SN}$ and $t \in \mathrm{SN}$.*
3. *The σ_0-reducts of W-terms are also W-terms.*
4. *For s, t W-terms, if $s \in \mathrm{SN}$ and $t \in SN$ then $s \circ t \in SN$.*
5. *If s is a W-term, then $s \in \mathrm{SN}$.*

But we want to show that all σ_0-terms are strongly normalising. To do so, the (Abs) rule must be handled. [12] hence comes to the conclusion that what remains for this to be shown is to establish the property:

$$(*) \text{ if } s \in \mathrm{SN}, \text{ then } s\circ \uparrow \in \mathrm{SN}.$$

Proving $(*)$ is difficult and one needs to prove a more general result: any "increment" of a strongly normalising term is strongly normalising. In other words, $(*)$ needs to be strengthened in order to make an induction argument work. To this end, [12] introduced the notion of "context" and a machinery for the contexts when the "increment" is reduced. Of particular relevance are the notions of (very) good contexts (see below).

The basic idea of the context calculus is to think of a term t as a "context" with multi-holes filled by its sub-terms, and to check the machinery of these contexts while reducing t.

DEFINITION 10 (Contexts). *Contexts with multi-holes are given inductively by:* $\mathrm{Cont} ::= \Box_n \mid 1 \mid id \mid \uparrow \mid \lambda C \mid C \cdot D \mid C \circ D$ *where $n \geq 1$ and $\Box_n$ denotes a hole.*

NOTATION 11. *Let γ be an occurrence within the context C. The notation C/γ stands for the subcontext of C at the occurrence γ and $C\{\gamma \leftarrow s\}$ stands for the context obtained by replacing in C the subcontext C/γ by the term s. An analogous notation is used for terms: s/γ and $s\{\gamma \leftarrow s\}$. When C/γ is a hole, then $C[s]_\gamma$ is written instead of $C\{\gamma \leftarrow s\}$. The notation $\Box_k \in C$ means that there exists an occurrence γ in the set of occurrences of C such that $C/\gamma = \Box_k$.*

DEFINITION 12 (Hole filling). *Let C be a context, $n_C = max\{m : \Box_m \in C\}$, $n \geq n_C$ and $u = (u_1, \ldots u_n)$ a tuple of terms. Then $C[u] = C[u_1, \ldots, u_n]$ is the term obtained by placing u_k in all the holes $\Box_k$ of C for $1 \leq k \leq n$.*

NOTATION 13. *Let C be a context and $q \geq 0$. C_q denotes the context obtained from C by renaming the holes $\Box_k$ as $\Box_{k+q}$.*

Let $u = (u_1, \cdots, u_m)$ and $v = (v_1, \cdots, v_n)$. The juxtaposition of u and v will be denoted by $u@v = (u_1, \cdots, u_m, v_1, \cdots, v_n)$. We will denote the length of u by $Lg(u)$ or $|u|$.

EXAMPLE 14. *Let $C = ((\lambda\Box_4)\cdot(\Box_2\circ\uparrow))\cdot(1\cdot\Box_4)$, $s = (\lambda(1\circ\uparrow))\cdot(1\cdot id)$ and $t = \uparrow\circ\lambda 1$. Then:*

$C/1 = (\lambda\Box_4)\cdot(\Box_2\circ\uparrow)$	$C\{12 \leftarrow t\} = ((\lambda\Box_4)\cdot(\uparrow\circ\lambda 1))\cdot(1\cdot\Box_4)$
$C/12 = \Box_2\circ\uparrow$	$s\{11 \leftarrow t\} = (\lambda(\uparrow\circ\lambda 1))\cdot(1\cdot id)$
$s/2 = 1\cdot id$	$C[t]_{121} = ((\lambda\Box_4)\cdot((\uparrow\circ\lambda 1)\circ\uparrow))\cdot(1\cdot\Box_4)$
$s/11 = 1\circ\uparrow$	$C_3 = ((\lambda\Box_7)\cdot(\Box_5\circ\uparrow))\cdot(1\cdot\Box_7)$
$t/21 = 1$	$C[\uparrow, 1\cdot id, \uparrow\cdot\uparrow, 1] = ((\lambda 1)\cdot((1\cdot id)\circ\uparrow))\cdot(1\cdot 1)$

Now we define the relation between contexts and terms:

DEFINITION 15 (Relative contexts). *A context C* relative to *s is a context such that $s = C[u]$ for some u. A hole $\Box_m$ in a context C relative to s is called a W-hole if the corresponding sub-term u_m is a W-term, otherwise it is called a L-hole.*

DEFINITION 16 (Inflations). *An inflation of s is a pair (C, w) where C is a context and w an n-tuple of W-terms such that there is a n-tuple of terms u which satisfies $s = C[u]$. We shall also say (s, C, u, w) or (C, u, w) is an inflation.*

The result of inflation (s, C, u, w) is $s' = C[u']$ where u' is given by:

$$u'_k = \begin{cases} w_k & \text{if } u_k \text{ is a W-term} \\ u_k \circ w_k & \text{otherwise} \end{cases}$$

We shall also call the result $s' = C[u']$ as an increment of s.

One can consider $'$ as an operator which takes u and w and gives back u'. The phrase '(C, w) is an inflation of $s = C[u]$' will stand for '(C, w) is an inflation of s and u is $Lg(w)$-tuple such that $s = C[u]$'.

REMARK 17. *$C[u'] = C[v']$ if (C, w) is an inflation of $s = C[u] = C[v]$.*

Below we give the restrictions on contexts. These restrictions were introduced in [12] in order to prove the Preservation Theorem 24.[3] A motivating example for introducing these restrictions is the following:

Let $C = \Box_1 \circ \Box_2$ and $s = C[t, \lambda u]$ where t is not a W-term. Take the inflation $(C(w_1, w_2))$ of s whose result is $(t\circ w_1)\circ((\lambda u)\circ w_2)$. It is possible to have $(t\circ w_1)\circ((\lambda u)\circ w_2) \to t\circ(w_1\circ((\lambda u)\circ w_2))$, but such

[3] Especially when considering the (Abs) case whose non-straightforwardness was commented on at the begin of this section.

a reduction must be forbidden because the reduct cannot be treated as the result of an inflation of s, when $w_1 \circ ((\lambda u) \circ w_2)$ is not a W-term.

The solution [12] proposed for this problem is to prevent $\Box_2$ from being a λ-hole. Hence, the following definition which says that a context C is good for s if it is a context for s, and whenever $A \circ B$ is a sub-context of C and there exists a hole in A, then B must be a W-hole.

DEFINITION 18 (Good contexts). *A good context relative to s is a context C relative to s which satisfies the condition that: for every occurrence γ of a composition in C, if there exists a hole at an occurrence of the form $\gamma 1 \alpha$, then $C/\gamma 2$ is a W-hole.*

Now we introduce the second restriction on contexts which says that a context C for s is very good if it is good, and whenever C/γ is a W-hole, s/δ is an L-term for any proper-prefix δ of γ.

DEFINITION 19 (Very good contexts). *A very good context C relative to s is a good context relative to s such that if $s = C[u]$ and u_k is a W-term, then for every occurrence γ such that $C/\gamma = \Box_k$ and for every δ proper prefix of γ, s/δ is not a W-term.*

DEFINITION 20 (Good/Very good inflation). *An inflation of s is a good/very good inflation if its context relative to s is good/very good.*

EXAMPLE 21. *The inflation $(\Box_1, \uparrow)$ is very good and for s an L-term, $s \circ \uparrow$ is the result of this very good inflation of s*

[12] gives two important lemmas (22 and 23) which say that good inflations behave nicely and that one can pass from good inflations to very good ones. Lemma 22 (resp. Lemma 23) is encoded in ALF as in Figure 13 (resp. Figure 14) and proved in Lemma 51 (resp. Lemma 52).

LEMMA 22 (Reduction of contexts preserves good inflations). *Let s' be the result of the good inflation (C, w) of $s = C[u]$. Let D be a context such that $C \to D$, and let $t = D[u]$ (hence, $s \to t$ and $s' \to t'$). Then there exists a good inflation (D', w') of t whose result is t'.*

LEMMA 23 (Very good inflations with same results as good inflations). *Let $\mathrm{K_c}(C)$ be the number of the holes in C. If (C, u, w) is a good inflation of s with result s', then there exists a very good inflation (C', u', w') of s with result s' that $\mathrm{K_c}(C') \leq \mathrm{K_c}(C)$.*

THEOREM 24 (Preservation of SN). *Let $s \in \mathrm{SN}$ and let s' be the result of a very good inflation (C, w) of $s = C[u]$. Then $s' \in \mathrm{SN}$.*

See Page 34 for a proof of Theorem 24 and Section 8 for the ALF proof of Corollary 25.

COROLLARY 25 (σ and σ_0 are strongly normalising). *The calculi σ and σ_0 are strongly normalising (i.e. Theorems 2 and 7 are now proved).*

Proof: First, we prove this corollary for σ_0. By Example 21 and Theorem 24, if s is an L-term, and $s \in$ SN, then $s \circ \uparrow \in$ SN. But, if s is a W-term, then $s \circ \uparrow$ is also a W-term and hence by Lemma 9, $s \circ \uparrow \in$ SN. Hence, in all cases, if $s \in$ SN, then $s \circ \uparrow \in$ SN. Hence, property $(*)$ is proved. Now, one can easily prove that if $s, t \in$ SN, then $s \circ t \in$ SN. Finally, use Lemma 9 to show that if $s \in \Lambda\sigma_0$ then $s \in$ SN. Now, for σ, simply use the strong normalisation of σ_0 and Lemma 6. □

3. The calculus λs and the termination proofs of s

We present the calculus λs [21] in this section and give three strong normalisation proofs of the s-calculus, each using a different method. The proof of Section 3.4 was given in [21] and is by a strict translation from s to σ (and the strong normalisation of the calculus σ). In Sections 3.2 and 3.3 we give two new proofs. In this paper, we formalise all these proofs in ALF. For comparison between λs and $\lambda\sigma$, see [23].

3.1. The calculus λs

DEFINITION 26 (The λs-calculus). *The terms Λs of the λs-calculus are given by: $a, b ::= N \mid ab \mid \lambda a \mid a\sigma^i b \mid \varphi^i_k a$ where $i \geq 1, k \geq 0$.*

The set, denoted λs, of rules of the calculus is given in Figure 2. The calculus of substitutions associated with the λs-calculus, called the s-calculus, is the rewriting system whose rules are $\lambda s \setminus \{\sigma$-generation$\}$.

THEOREM 27 (SN of s). *The s-calculus is strongly normalizing.*

3.2. Interpretations for the termination of the calculus s

The interpretation method can be used to prove the termination of s.

DEFINITION 28 (Polynomial interpretation of s). *The polynomial interpretations for s are defined by induction on the structure of the terms in Λs:* $[\![n]\!] = 2$, $[\![ab]\!] = [\![a]\!] + [\![b]\!] + 1$, $[\![\varphi^i_k a]\!] = 2[\![a]\!]$,
$[\![\lambda a]\!] = [\![a]\!] + 1$, $[\![a\sigma^i b]\!] = [\![a]\!]([\![b]\!] + 1)$

Now we can prove Theorem 29 which gives another termination proof of s. It was checked in ALF by some trivial inequalities like: "For any $a \in \Lambda s$, $[\![a]\!] \geq 2$" (proved by induction on the structure of terms of s).

(σ-generation)	$(\lambda a)b \to a\sigma^1 b$
(σ-λ-transition)	$(\lambda a)\sigma^i b \to \lambda(a\sigma^{i+1}b)$
(φ-λ-transition)	$\varphi^i_k(\lambda a) \to \lambda(\varphi^i_{k+1}a)$
(σ-app-transition)	$(a_1a_2)\sigma^i b \to (a_1\sigma^i b)(a_2\sigma^i b)$
(φ-app-transition)	$\varphi^i_k(a_1a_2) \to (\varphi^i_k a_1)(\varphi^i_k a_2)$
(σ-destruction)	$n\sigma^i b \to \begin{cases} n-1 & \text{if } n>i \\ \varphi^i_0 b & \text{if } n=i \\ n & \text{if } n<i \end{cases}$
(φ-destruction)	$\varphi^i_k n \to \begin{cases} n+i-1 & \text{if } n>k \\ n & \text{if } n\leq k \end{cases}$

Figure 2. Rules of λs

THEOREM 29. *For any $a, b \in \Lambda s$, if $a \to_s b$ then $[\![a]\!] > [\![b]\!]$.*

3.3. Another termination proof of s by induction

Now we give the direct proof that s is strongly normalising. This proof is reminiscent of the method of Reducibility Candidates. It is done by structural induction, the method used to prove strong normalisation of σ_0 (the proof is easier for s). Let SN be the set of all strongly normalising terms. For $t \in$ SN, $dpth(t)$ is the length of the longest derivation[4], $lgth(t)$ is the number of variables and operations defined as follows:

$lgth(n)$	$=1$		
$lgth(\lambda a)$	$= lgth(a)+1$	$lgth(ab)$	$= lgth(a)+lgth(b)+1$
$lgth(\varphi^i_k a)$	$= lgth(a)+1$	$lgth(a\sigma^i b)$	$= lgth(a)+lgth(b)+1$

Since there are no rules of the calculus s which contain "λ" or "apply" as head symbol, in order to prove that all terms are terminating, we need only to check that if $a, b \in$ SN, then $\varphi^i_k a \in$ SN and $a\sigma^i b \in$ SN which we prove in Lemma 30 by lexicographic induction on $(dpth(a), lgth(a)))$ and $(dpth(a), lgth(a), dpth(b), lgth(b))$ respectively.

LEMMA 30. *Let $a, b \in \Lambda s$. We have (iff stands for if and only if):*

1. *$ab \in$ SN iff $a \in$ SN and $b \in$ SN. Also, $\lambda a \in$ SN iff $a \in$ SN.*
2. *If $a \in$ SN, then $\varphi^i_k a \in$ SN for all $i \geq 1$, $k \geq 0$.*
3. *If $a, b \in$ SN, then $a\sigma^i b \in$ SN for all $i \geq 1$, $k \geq 0$.*

[4] $dpth(t)$ is well defined for $t \in$ SN by König Lemma. Note also that when a, b are terminating and $a \longrightarrow^+ b$, then $dpth(a) > dpth(b)$. However, we do not need the notion of "depth" when formalising the lexicographic induction principle in ALF.

3.4. A termination proof of s via termination of σ

[21] shows strong normalisation of s by giving a strict translation from s to σ (Theorem 32 below) which we formalise in ALF in Section 6. The termination proof of σ will be given in Section 8.

DEFINITION 31. *We introduce some notations:*

- *Let $k \geq 0$, $i \geq 1$. Define $s_{ki} = 1 \cdot 2 \cdot \ldots \cdot k \cdot \uparrow^{k+i-1}$ (write $s_{0i} = \uparrow^{i-1}$).*
- *Let $b \in \Lambda\sigma^t$. We define a family of substitutions $(b_k)_{k\geq 1}$ as follows:*
  $b_1 = b[id] \cdot id \quad b_2 = 1 \cdot b[\uparrow] \cdot \uparrow \quad \ldots \quad b_{i+1} = 1 \cdot 2 \cdot \ldots \cdot i \cdot b[\uparrow^i] \cdot \uparrow^i$
- *Let $\Lambda\sigma = \Lambda\sigma^t \cup \Lambda\sigma^s$. Define the function $T : \Lambda s \to \Lambda\sigma$ by:*
  $T(n) = n \quad T(ab) = T(a)T(b) \quad T(\varphi^i_k a) = T(a)[s_{ki}]$
  $T(\lambda(a) = \lambda(T(a)) \quad T(a\sigma^i b) = T(a)[T(b)_i]$

THEOREM 32. *If $a \to_s b$ then $T(a) \to^+_\sigma T(b)$.*

4. The proof assistant ALF

4.1. About Martin-Löf's Type Theory

In Martin-Löf's type theory [25] predicate logic is interpreted within type theory through Pat, the Curry-Howard-de Bruijn interpretation of propositions as types (sets). A proposition is interpreted as a set whose elements represent the proofs of the proposition. Hence, a false proposition is interpreted as the empty set and a true proposition as a non-empty set. To prove a proposition is to prove the set is inhabited.

There are two ways of introducing types in Martin-Löf's type theory: function types and inductively defined sets of the type **Set**. The function types make it possible to express rules in a natural deduction style and logic can then be introduced by the Pat principle.

For every inductively defined set, one finds *one formation rule* (expressing how to form a set), *introduction rules* (expressing how to form the elements of the set), and *one elimination rule* (giving the induction principle for this set, i.e. how to prove all the elements of the set satisfy some property). Basically one states in the elimination rule that if for every constructor one can show the property holds, then the property holds for all the elements of the set. Another way to look at the elimination rule is that it says there are no other objects in this set except those given by the introduction rules. There is a general scheme to derive the elimination rule from the introduction rules of a set.

As an example, we give the formation, elimination and introduction rules of the set of natural numbers N. First, we give some notations:

NOTATION 33. *A function f which takes arguments $x_1 \in A_1, \ldots, x_n \in A_n$ and returns $f(x_1, \ldots x_n) \in B$, is written as $f \in (x_1 \in A_1, \ldots, x_n \in A_n)B$. E.g., the successor function on natural numbers is written as* $\mathrm{succ} \in (n \in \mathrm{N})\mathrm{N}$. *I.e., it takes $n \in \mathrm{N}$ and returns* $\mathrm{succ}(n) \in \mathrm{N}$.

Now, the set N is formed by the *formation rule*: $\dfrac{}{\mathrm{N} \in \mathbf{Set}}$

The elements of the set N are defined by two *introduction rules*:[5]

$$\frac{}{\mathrm{zero} \in \mathrm{N}} \qquad\qquad \frac{a \in \mathrm{N}}{\mathrm{succ}(a) \in \mathrm{N}}$$

The *elimination rule* of N is just the induction principle:

$$\frac{\begin{array}{l} C(v)\,set[v \in \mathrm{N}] \\ a \in \mathrm{N} \\ d \in C(\mathrm{zero}) \\ e(x,y) \in C(\mathrm{succ}(x))[x \in \mathrm{N}, y \in C(x)] \end{array}}{\mathrm{natrec}(a,d,e) \in C(a)}$$

In ALF (see Section 4.2), the introduction rules of N look like:

$$\begin{array}{l} \mathrm{N} \in \mathbf{Set} \\ \quad \mathrm{zero} \in \mathrm{N} \\ \quad \mathrm{succ} \in (n \in \mathrm{N})\mathrm{N} \end{array}$$

We present rules in a natural deduction style or in ALF style above. We use $a \in A$ or $a : A$ to denote a is an element (object) of the set (type) A. A proposition is proved by constructing a proof object, or an element of the set in ALF. Objects of a type are formed from constants and variables using application (app) and abstraction (abs) given by:

$$\frac{c \in (x \in A)B \qquad a \in A}{c(a) \in B[x := a]}\ (\mathrm{app}) \qquad \frac{b \in B[x \in A]}{[x]b \in (x \in A)B}\ (\mathrm{abs})$$

4.2. The proof assistant ALF

ALF [24, 32] implements a *monomorphic* version of type theory where all type information is in the term. As a result there is a lot of redundant or uninteresting type information, and the size of the proofs can be very large. However, the user can instruct ALF to suppress unwanted type information when displaying proofs. ALF emphasizes the interactive development of type-theoretic constructions, i.e. proof objects and programs, using a window-based user interface. Thus ALF supports an arbitrary mixture of top-down and bottom-up development.

The basic metaphor of ALF is the refinement of an incomplete proof object which is displayed in a window (scratch area). Using the mouse,

[5] Here N is a set having two *constructors*: the nullary zero and the unary succ, which is a function from N to N.

the user can fill in placeholders by pointing at them and then selecting a previously constructed object from a menu. In ALF place-holders are used to represent those parts of objects which are to be filled in. The expression $? \in A$ expresses a state of an ongoing process of finding an object in the type A. There are four ways of refining a placeholder:

- The placeholder is replaced by a constant c. This is correct if the type of c is equal to A.
- The placeholder is replaced by a variable x, where x must be in the local scope of the placeholder.
- The placeholder is replaced by an abstraction $[x]? \in A$ if A is equal to function type $(y \in B)C$. We are constructing a solution to the problem C under the assumption that we have a solution to B.
- The placeholder can be replaced by an application $c(?_1, \ldots, ?_n)$. In this case we can divide the problem to several subproblems.

We have used Window ALF which was implemented by Magnusson [24]. By the Pat principle, to prove a theorem in ALF is the same as writing a program "witnessing" the truth of the theorem. This is a fundamental difference between ALF and HOL (and many other proof-assistants), where the proof is presented as a sequence of tactics.

5. Formalising explicit substitution calculi in ALF

In this section we give the implementations of σ_0, σ and s, and present the notion of termination and the well-founded induction principle.

5.1. Explicit substitution calculi in ALF

An explicit substitution calculus is defined by a set of terms (substitutions) and a set of rules. Each set is inductively defined by its introduction rules in ALF. There is an elimination rule for each inductively defined set, which gives the induction principle on the set.

For σ_0, the set of terms $\Lambda\sigma_0$ and the reduction rules $R\sigma_0$ (cf. Definition 3) are given below. Note that in ALF, we should use something like Sig0Term instead of $\Lambda\sigma_0$, but for readability, we write the latter in this paper and follow the same abuse of notation for all other names:

$\Lambda\sigma_0 \in \mathbf{Set}$	$R\sigma_0 \in (s,t \in \Lambda\sigma_0)\mathbf{Set}$
$1 \in \Lambda\sigma_0$	$VrId \in R\sigma_0(1 \circ id, 1)$
$id \in \Lambda\sigma_0$	$VrCons \in R\sigma_0(1 \circ (s \cdot t), s)$
$\uparrow \in \Lambda\sigma_0$	$Abs \in (s,t \in \Lambda\sigma_0)R\sigma_0((\lambda s) \circ t, \lambda(s \circ (1 \cdot (t \circ \uparrow))))$
$\lambda \in (a \in \Lambda\sigma_0)\Lambda\sigma_0$	$IdL \in R\sigma_0(id \circ s, s)$
$\circ \in (s,t \in \Lambda\sigma_0)\Lambda\sigma_0$	$ShId \in R\sigma_0(\uparrow \circ id, \uparrow)$
$\cdot \in (s,t \in \Lambda\sigma_0)\Lambda\sigma_0$	$ShCons \in R\sigma_0(\uparrow \circ (s \cdot t), t)$
	$Map \in R\sigma_0((s \cdot t) \circ u, (s \circ u) \cdot (t \circ u))$
	$Ass \in R\sigma_0((s \circ t) \circ u, s \circ (t \circ u))$

For readability, we will not be using the real syntax of ALF in this paper. The reader can refer to [1] for all the code in ALF. We abuse the notation of ALF and write some operators as infix rather than prefix, we use the same names of the original calculus rather than names allowed by the syntax of ALF, and we suppress some arguments of the constructors. The real implementation in ALF looks as follows:

$R\sigma_0 \in (s,t \in \Lambda\sigma_0)\mathbf{Set}$
$\quad VrId \in R\sigma_0(Com(V_1, Id), V_1)$
$\quad VrCons \in (s,t \in \Lambda\sigma_0)R\sigma_0(Com(V_1, App(s,t)), s)$
$\quad Abs \in (s,t \in \Lambda\sigma_0)R\sigma_0(Com(Lam(s),t), Lam(Com(s, App(V_1, Com(t, Shift)))))$
$\quad \ldots$

One step reduction $\rightarrow$ is formalised as follows:

$$\begin{array}{l}
\rightarrow \in (s,t \in \Lambda\sigma_0)\mathbf{Set} \\
\quad Direct \in (R\sigma_0(s,t))s \rightarrow t \\
\quad \lambda Compa \in (s \rightarrow t)\lambda s \rightarrow \lambda t \\
\quad \circ CompL \in (s_1 \rightarrow s_2)s_1 \circ t \rightarrow s_2 \circ t \\
\quad \circ CompR \in (s_1 \rightarrow s_2)t \circ s_1 \rightarrow t \circ s_2 \\
\quad \cdot CompL \in (s_1 \rightarrow s_2)s_1 \cdot t \rightarrow s_2 \cdot t \\
\quad \cdot CompR \in (s_1 \rightarrow s_2)t \cdot s_1 \rightarrow t \cdot s_2
\end{array}$$

The calculi σ and s are formalised in the same way.

5.2. Termination of explicit substitutions in ALF

Let (A, R) be an explicit substitution calculus, where R is the one step reduction relation. Termination is defined as a family of sets inductively defined in ALF by the *Formation*, *Introduction*[6] and *elimination* rules given in Figure 3. E.g., we give the definition of SN for σ_0 in Figure 4.

The introduction rule says that an element a is strongly normalising if whenever it is one step reduced to a term b, b is also strongly normalising. This is a recursive definition. SNintr is the constructor. We will write SN(a) or $a \in$ SN when the term a is strongly normalising.

Let C be a proposition on SN(a) for $a \in A$. The elimination rule says that to prove C holds for $sn \in$ SN(t) and $t \in A$, we need to show:

[6] This is a typical constructive way to describe infinite objects. We have hidden the arguments A and R in the introduction rule.

$$\frac{a : A}{\mathrm{SN}(a) : Set} \text{ (Formation)}$$

$$\frac{a : A}{\mathrm{SNintr} : (a : A; h : (b : A; R(a,b))\mathrm{SN}(b))\mathrm{SN}(a)} \text{ (Introduction)}$$

$$\frac{\begin{array}{l} C : (t : A; \mathrm{SN}(t))Set \\ d : (x : A; b : (y : A; R(x,y))\mathrm{SN}(y); b_1 : T_1)T_2{}^{\dagger} \\ t : A \\ sn : \mathrm{SN}(t) \end{array}}{\mathrm{SNelim}(C, b, t, sn) : C(t, sn)} \text{ (Elimination)}$$

†: where $T_1 \equiv (y : A; a : R(x,y))C(y, b(y,a))$ and $T_2 \equiv C(x, \mathrm{SNintr}(x,b))$

Figure 3. Termination

$$\begin{array}{l} SN \in (a \in \Lambda\sigma_0)\mathbf{Set} \\ \quad \mathrm{SNintr} \in (a \in \Lambda\sigma_0; (b \in \Lambda\sigma_0; a \to b)SN(b))SN(a) \end{array}$$

Figure 4. Definition of SN for σ_0

- if whenever x is one step reduced to y then y is strongly normalising and there is a proof of $C(y, b(y,a))$, then we can get a proof of $C(x, \mathrm{SNintr}(x,b))$ (note that b is the induction hypothesis).

We will use the non-dependent version of the recursor RecSN given in Figure 5 to simulate induction over the length of the longest reduction of a strongly normalising term. RecSN as an induction principle says that SN is the smallest set of terms closed under one step reduction.

In later sections we will need to prove propositions like $\mathrm{SN}(a)$ implies $\mathrm{SN}(a')$. To prove such propositions using the induction principle of Figure 5 we can try to find a predicate P such that $\mathrm{SN}(a)$ implies P(a), and P(a) implies $\mathrm{SN}(a')$. To prove $\mathrm{SN}(a)$ implies P(a), by the induction principle we need only to prove P is closed under one step reduction. We will use this technique to prove some lemmas in Section 6. Alternatively, we define $a \prec b$ if $b \to a$. Then a reduction $\to$ is strongly normalising if and only if the order $\prec$ is well founded. Hence the induction principle RecSN is just the well founded induction principle.

$$\frac{\begin{array}{l} P : (a : A)Set \\ h : (m : A; (n : A; R(m,n))\mathrm{SN}; (n : A; R(m,n))P(n))P(m) \\ m_1 : A \\ sn : \mathrm{SN}(m_1) \end{array}}{\mathrm{RecSN}(P, h, m_1, sn) : P(m_1)}$$

Figure 5. RecSN

$\Lambda s \in \mathbf{S}et$	$[\![]\!] \in (\Lambda s)\mathrm{N}$
$Var \in (n \in \mathrm{N}; n > \text{zero}))\Lambda s$	$[\![Var(n)]\!] \equiv succ(succ(\text{zero}))$
$\text{ap} \in (s, t \in \Lambda s)\Lambda s$	$[\![st]\!] \equiv succ(plus([\![s]\!], [\![t]\!]))$
$\lambda \in (s \in \Lambda s)\Lambda s$	$[\![\lambda t]\!] \equiv succ([\![t]\!])$
$\sigma \in (s \in \Lambda s; j \in \mathrm{N}; j > \text{zero}); t \in \Lambda s)\Lambda s$	$[\![s\sigma^j t]\!] \equiv Multiply([\![s]\!], succ([\![t]\!]))$
$\varphi \in (i, k \in \mathrm{N}; i > \text{zero}); s \in \Lambda s)\Lambda s$	$[\![\varphi^i_k s]\!] \equiv Multiply(succ(succ(\text{zero})), [\![s]\!])$

Figure 6. The (sugared) Implemenation of the calculus s

6. Formalising the termination proof of s in ALF

6.1. Formalising the Interpretations for termination of s

The implementation of the interpretations for the termination of s is not difficult. All we need is some inequalities about the addition and multiplication of natural numbers. See Section 3.2.

The (sugared) implementation of the calculus s (Definition 26) and of its interpretation (Definition 28) is given in Figure 6 (again, by abuse of notation, we write ab instead of $\text{ap}(a, b)$).
We formalise Theorem 29 as: $Prop1 \in (a, b \in \Lambda s; a \to b)[\![a]\!] > [\![b]\!]$.

All the details of this proof of termination of s using interpretation (see Section 3.2) have been fully formalised in ALF. See [1].

6.2. Formalising the induction termination proof of s

Now let us see how to implement the strong normalisation proof of s given in Section 3.3. We use $\mathrm{SN_s}$ for strong normalisation in λs.

Theorem 27 is proved by induction on the structure of terms, or using the elimination rule:

$$
\begin{array}{l}
\Lambda s\text{\_elim} : (C : (\Lambda s)Set) \\
\quad (e_1 : C(Var(n))) \\
\quad (e_2 : (a, b : \Lambda s; C(a); C(b))C(ab)) \\
\quad (e_3 : (a : \Lambda s; C(a))C(\lambda a) \\
\quad (e_4 : (a : \Lambda s; i, k : N; C(a))C(\varphi^i_k a) \\
\quad (e_5 : (a, b : \Lambda s; i : N; C(a); C(b))C(a\sigma^i b) \\
\quad (a : \Lambda s) \\
\quad C(a)
\end{array}
$$

To this end we need Lemma 36 below which gives the proof objects of e_1, e_2, e_3, e_4 and e_5 when C is $\mathrm{SN_s}$. Intuitively, $\mathrm{SN_s}$ holds for all normal forms because for them the premise of the introduction rule for $\mathrm{SN_s}$ is vacuously true. Since every variable is strongly normalising, e_1 is easy to get. After proving Lemma 36, we finish the proof of Theorem 27 by induction on the structure of terms.

REMARK 34. *By using pattern matching, we do not need to write the elimination rule of Λs. Theorem 27 reads in ALF:* $\mathrm{SN_s} : (a : \Lambda s)\mathrm{SN_s}(a)$. *By pattern matching on the argument a, we get the following equations:*

$$\begin{aligned}&\mathrm{SN_s} : (a : \Lambda s)\mathrm{SN_s}(a)\\ &\quad \mathrm{SN_s}(Var(n)) = ?_{e_1}\\ &\quad \mathrm{SN_s}(ab) = ?_{e_2}\\ &\quad \mathrm{SN_s}(\lambda(a)) = ?_{e_3}\\ &\quad \mathrm{SN_s}(\varphi^i_k a) = ?_{e_4}\\ &\quad \mathrm{SN_s}(a\sigma^i b) = ?_{e_5}\end{aligned}$$

Here we have the same tasks to give those proof objects $e_1, \cdots, e_5$.

REMARK 35. *It is easy to prove Lemma 36 in classical logic using the definition "there are no infinite derivations". But when proving it in ALF, we can't use the classical law of refutation. In ALF, we must use introduction and elimination rules to give a constructive proof.*

LEMMA 36. *The following hold:*
1. $ab \in \mathrm{SN_s}$ *if and only if* $a \in \mathrm{SN_s}$ *and* $b \in \mathrm{SN_s}$.
2. $\lambda a \in \mathrm{SN_s}$ *if and only if* $a \in \mathrm{SN_s}$.
3. *For any* $i \geq 1, k \geq 0$, $\varphi^i_k a \in \mathrm{SN_s}$ *if and only if* $a \in \mathrm{SN_s}$.
4. *For any* $i \geq 1$, $a\sigma^i b \in \mathrm{SN_s}$ *if and only if* $a, b \in \mathrm{SN_s}$.

Proof: Cases 1 and 2 are by induction on the derivation sequences, that is by $\mathrm{SN_s}$ elimination. Cases 3 and 4 need to combine the $\mathrm{SN_s}$ elimination and term elimination, which correspond to induction on (dpth(a), lgth(a)).[7] Below, we only give the ALF proof of case 2.
Case 2: We prove first the "only if" part. We define a predicate $P_1(a) \equiv \forall x \in \Lambda s((a = \lambda(x)) \Rightarrow \mathrm{SN_s}(x))$. We will prove the following facts:
1. $\forall x \in \Lambda s(P_1(\lambda(x)) \Rightarrow \mathrm{SN_s}(x))$. 2. $\forall x \in \Lambda s(\mathrm{SN_s}(x) \Rightarrow P_1(x))$.

1. It is easy to see that $P_1(\lambda(a))$ implies $\mathrm{SN_s}(a)$ by definition of P_1. This is proved in ALF by giving a function which for any proof of $P_1(\lambda(a))$ gives a proof of $\mathrm{SN_s}(a)$:

   Suppose we have a proof $h : P_1(\lambda(a))$; by the elimination rule of $\forall$, we have a proof: $\mathrm{Forall\_elim}(h, \lambda(a)) : (\lambda(a) = \lambda(a)) \Rightarrow \mathrm{SN_s}(a))$. By the elimination rule of $\Rightarrow$ and a proof $r : \lambda(a) = \lambda(a)$, we get a proof of $\mathrm{SN_s}(a)$. The final proof of $P_1(\lambda(a)) \Rightarrow \mathrm{SN_s}(a)$ for any $a \in \Lambda s$ looks like: $\mathrm{Imply\_intro}([h]\mathrm{Imply\_elim}(\mathrm{Forall\_elim}(h, \lambda(a)), r)$. Using introduction rule of $\forall$, we get the proof of 2.

[7] Recall that $dpth(a)$ is the number of reductions in the longest derivation path starting from term a, however we don't need to formalise $dpth(a)$, which is a partial function defined only on strongly normalising terms.

2. We must prove that $\forall x \in \Lambda s(\mathrm{SN_s}(x) \Rightarrow P_1(x))$. This is by the induction principle $\mathrm{RecSN_s}$, which amounts to induction on derivations. We should prove that $P_1(x)$ is closed under one step reduction, i.e.: $\dfrac{m : \Lambda s; h : (n : \Lambda s; m \to n)P_1(n)}{? : P_1(m)}$

   Note that we denote statemens of the form: "under assumption A, find a proof of type B" by: $\dfrac{A}{? : B}$

   $P_1(m)$ is solved by introduction rules of universal quantifier, imply and by finding a proof object of type $\mathrm{SN_s}(x)$: $\dfrac{x : \Lambda s; m \equiv \lambda(x)}{? : \mathrm{SN_s}(x)}$

   The problem $\mathrm{SN_s}(x)$ is solved by the introduction rule of $\mathrm{SN_s}$ and finding a proof object of type $\mathrm{SN_s}(b)$: $\dfrac{b : \Lambda s; h_2 : x \to b}{? : \mathrm{SN_s}(b)}$

   This is proved using the proofs of $P_1(\lambda(b)) \Rightarrow \mathrm{SN_s}(b)$ and $P_1(\lambda(b))$. The proof of $P_1(\lambda(b))$ comes from the proof $h : (n : \Lambda s; m \to n)P_1(n)$, where $m = \lambda(x), n = \lambda(b)$ and $m \to n$ because $h_2 : x \to b$.

The "if" direction is proved in the same way: Suppose $P_2(x) \equiv \mathrm{SN_s}(\lambda(x))$, then we can prove that $P_2(x)$ is closed under one step reduction. By reduction on derivations, we prove that $\forall x \in \Lambda s(\mathrm{SN_s}(x) \Rightarrow P_2(x))$. Finally, it is easy to see that $P_2(x)$ implies $\mathrm{SN_s}(\lambda(x))$. □

Again, all the details of this induction termination proof of s (see Section 3.3) have been fully formalised in ALF. See [1].

6.3. Formalising the termination proof of s via σ

In this section, we formalise in ALF, the strong normalisation of s by giving a translation from s to σ (see Section 3.4). The translation T of Definition 31 is formalised in ALF as follows (in sugared notation):

$$\begin{array}{l} T \in (a \in \Lambda s)\Lambda\sigma \\ T(Var(n)) \equiv code(n) \\ T(st) \equiv T(s)T(t) \\ T(\lambda t) \equiv \lambda T(t) \\ T(s\sigma^j t) \equiv T(s)[T(t_j] \\ T(\varphi^i_k s) \equiv T(s)[s_{ki}] \end{array}$$

Theorem 32 is checked, which is implemented in ALF (again in sugared notation) as: $SigSimulateS \in (a, b \in \Lambda s; p \in a \to b)T(a) \to^+_\sigma T(b)$.

The ALF proof of this theorem is by case analysis on the proof object p. We should check when any of the seven $\to$ reduction rules for p, the theorem is correct. One of the main tasks is, when coming to the rule *Direct* for one step reduction in s (see page 15), to prove the theorem holds for any of the reduction rules of s. We can use induction to prove other cases, i.e. the compatible rules. For instance, we should prove the following propositions hold when we are at σ-destruction:

Projection$_1$: $(n : N; b : \Lambda\sigma^t)n[b_n] \to_\sigma^+ b[\uparrow^n]$
Projection$_2$: $(n, i : N; i > n; b : \Lambda\sigma^t)n[b_i] \to_\sigma^+ n$
Projection$_3$: $(n, i : N; n > i; b : \Lambda\sigma^t)n[b_i] \to_\sigma^+ n - 1$

It is easy to see that they are intuitively true. However when proving them in ALF, there are a lot of details which we need to check. Let us take these projections to see some details of the ALF proof.

First of all, we should have a denotation for b_i for any $i \geq 1$. When we write b_i, we actually refer to a function $b_i : \Lambda\sigma^t \times N \to \Lambda\sigma^s$. In the following we will feel free to use the convention notation b_i defined by: $b_i = \text{ConcaFinite}(i, b[\uparrow^{i-1}] \cdot \uparrow^{i-1})$ where:

$$\begin{aligned} &\text{ConcaFinite} : (n : N; s : \Lambda\sigma^s)\Lambda\sigma^s \\ &\quad \text{ConcaFinite}(0, s) = s; \\ &\quad \text{ConcaFinite}(n+1, s) = \text{ConcaFinite}(n, n \cdot s) \end{aligned}$$

Alternatively we could give b_i by two simultaneously defined functions $\uparrow$ Subs and $\uparrow$ term: $b'_1 = b[id] \cdot id$ and $b'_{i+1} = 1 \cdot \uparrow \text{Subs}(b'_i)$ where

$\uparrow \text{Subs} : (s : \Lambda\sigma^s)\Lambda\sigma^s$	$\uparrow \text{term} : (a : \Lambda\sigma^t)\Lambda\sigma^t$
$\uparrow \text{Subs(id)} = \uparrow$	$\uparrow \text{term}(1) = 1[\uparrow]$
$\uparrow \text{Subs}(\uparrow) = \uparrow \circ \uparrow$	$\uparrow \text{term}(ab) = \uparrow \text{term}(a) \uparrow \text{term}(b)$
$\uparrow \text{Subs}(a \cdot s) = \uparrow \text{term}(a) \cdot \uparrow \text{Subs}(s)$	$\uparrow \text{term}(\lambda a) = \lambda(a[1 \cdot (\uparrow \circ \uparrow)])$
$\uparrow \text{Subs}(s \circ t) = s \circ \uparrow \text{Subs}(t)$	$\uparrow \text{term}(a[s]) = a[\uparrow \text{Subs}(s)]$

LEMMA 37. *For any $a \in \Lambda\sigma^t$, $s \in \Lambda\sigma^s$ and $n \in$ N, the following hold:*

1. $a[\uparrow] \to_\sigma^\star \uparrow \text{term}(a)$.
2. $s \circ \uparrow \to_\sigma^\star \uparrow \text{Subs}(s)$.
3. $\uparrow \text{Subs}(\uparrow^n) = \uparrow^{n+1}$.
4. $\uparrow \text{term}(n) = n + 1$.

Projection$_1$ is proved by Lemma 38, which is shown by induction on n:

LEMMA 38. $\uparrow^n \circ \text{ConcaFinite}(n, s) \to_\sigma^+ s$ *for any $s \in \Lambda\sigma^s$ and $n > 0$.*

LEMMA 39. Projection$_1$: $(n : N; n > 0; b : \Lambda\sigma^t)n[b_n] \to_\sigma^+ b[\uparrow^n]$.

Proof: By induction on n.

– For n = 1, $1[b_1] = 1[b[id] \cdot id] \to b[id] = b[\uparrow^0]$ by the rule (VrCons).

– For n+1, $(n+1)[b_n] = 1[\uparrow^n][b_n] \to 1[\uparrow^n \circ b_n] \to 1[b[\uparrow^n] \cdot \uparrow^n] \to b[\uparrow^n]$ by the rule (Clos) and Lemma 38. □

Similarly, Projection$_2$ is proved by induction on n.

– For n = 1 we need to prove: $1[\text{ConcaFinite}(b, i)] \to 1$

This is proved by the rule (VrCons) in one step. This is because $b_i = 1 \cdot s$ for some substitution s when $i > 1$. However, b_i is defined by the function ConcaFinite, we need to prove this fact. It is immediate if we use the notation b'_i.

- For n+1, we should prove: $1[\uparrow^n][b_i] \to_\sigma^+ 1[\uparrow^n]$.

  By the closure rule, we have: $1[\uparrow^n][b_i] \to_\sigma 1[\uparrow^n \circ b_i]$.

  Intuitively $\uparrow^n \circ b_i = (n+1) \cdot \ldots \cdot b[\uparrow^{i-1}] \cdot \uparrow^{i-1}$, and Projection$_2$ is solved by the rule VarCons.

  Therefore we should prove that $\uparrow^n \circ b_i = (n+1) \cdot \ldots \cdot b[\uparrow^{i-1}] \cdot \uparrow^{i-1}$

  Now there should be a notation for $(n+1) \cdot \ldots \cdot b[\uparrow^{i-1}] \cdot \uparrow^{i-1}$

  So we define another notation ConcaFinite3:

  $$\begin{array}{ll} \text{ConcaFinite3} & : (n, i : N; s : \Lambda\sigma^s)\Lambda\sigma^s \\ & \text{ConcaFinite3}(n, 0, s) = n \cdot s \\ & \text{ConcaFinite3}(n, i+1, s) = \text{ConcaFinite3}(n, i, (n+i+1) \cdot s) \end{array}$$

LEMMA 40. *Let $n \in \mathrm{N}$, $s \in \Lambda\sigma^s$ and $i > n$.*
- ConcaFinite3$(0, n, s)$ = ConcaFinite$(n+1, s)$
- $\uparrow^n \circ$ConcaFinite$(i, s) \to_\sigma^+$ ConcaFinite3$(n, i-n-1, s)$.

We have to prove that ConcaFinite3(n, i, s) has the form $n \cdot s$ for some substitution s explicitly.

LEMMA 41. *Let $i, n \in \mathrm{N}$, $s \in \Lambda\sigma^s$, $LS1(n, i+1, s) =$* ConcaFinite3$(n+1, i, s)$ *and $LS1(n, 0, s) = s$.* ConcaFinite3$(n, i, s) = n \cdot LS1(n, i, s)$.

LEMMA 42. Projection$_2$: $(n, i : N; i > n; b : \Lambda\sigma^t)n[b_i] \to_\sigma^+ n$.

Projection$_3$ is proved by induction on the proof object $p : n > i$ based on proving the following lemma:

LEMMA 43. $\uparrow^{n+1+i} \circ b_i \to_\sigma^+ \uparrow^{n+i}$ *for any $i, n \in \mathrm{N}$.*

This is because $n[b_i] = 1[\uparrow^{n-1}][b_i] \to 1[\uparrow^{n-1} \circ b_i] \to 1[\uparrow^{n-2}]$.

LEMMA 44. Projection$_3$: $(n, i : N; n > i; b : \Lambda\sigma^t)n[b_i] \to_\sigma^+ n-1$.

Having proved T is a strict interpretation, we can conclude Theorem 27 using of course the proof of termination of σ (Section 8).

Again, all the details of this termination proof of s via termination of σ (see Section 3.3) have been fully formalised in ALF. See [1].

7. Formalising the context calculus of σ_0 in ALF

In this section we formalise in ALF all the notions informally given in [12, 30]. In [12], many notions were taken for granted and not introduced, and many lemmas were left unproven. To formalise the proofs of [12] we had to rewrite all the intuitions and informal notions, and to

$/ : (C \in \mathrm{Cont}; \gamma \in \mathcal{L})\mathrm{Cont}$	$/ : (s \in \Lambda\sigma_0; \gamma \in \mathcal{L})\Lambda\sigma_0$
$C/nil = C$	$t/nil = t$
$\Box_m/\{1,2\}\gamma = 1_c$	
$1_c/\{1,2\}\gamma = 1_c$	$1/\{1,2\}\gamma = 1$
$\mathrm{id}_c/\{1,2\}\gamma = \mathrm{id}_c$	$\mathrm{id}/\{1,2\}\gamma = \mathrm{id}$
$\uparrow_c /\{1,2\}\gamma = \uparrow_c$	$\uparrow /\{1,2\}\gamma = \uparrow$
$\lambda_c(C)/\{1,2\}\gamma = C/\gamma$	$\lambda(s)/\{1,2\}\gamma = s/\gamma$
$C \cdot_c D/1\gamma = C/\gamma$	$s \cdot t/1\gamma = s/\gamma$
$C \cdot_c D/2\gamma = D/\gamma$	$s \cdot t/2\gamma = t/\gamma$
$C \circ_c D/1\gamma = C/\gamma$	$s \circ t/1\gamma = s/\gamma$
$C \circ_c D/2\gamma = D/\gamma$	$s \circ t/2\gamma = t/\gamma$

Figure 7. Formulation of Notation 11

check a lot of details. Sometimes we had to change the implementation to make the proofs go through. We shall discuss some of the implementations during the process of the formalisation in this section.[8] When formalising on a machine, nothing can be left to the intuition.

The contexts of Definition 10 are formalised in ALF as follows:

$$\begin{array}{l} \mathrm{Cont} \in \mathbf{Set} \\ \quad \Box \in (n \in \mathrm{N}; n \geq 1)\mathrm{Cont} \\ \quad id_c, 1_c, \uparrow_c \in \mathrm{Cont} \\ \quad \lambda_c \in (C \in \mathrm{Cont})\mathrm{Cont} \\ \quad \cdot_c, \circ_c \in (C, D \in \mathrm{Cont})\mathrm{Cont} \end{array}$$

Let $\{1,2\}$ mean 1 or 2. Let $\mathcal{L}$ be the set of lists of $\{1,2\}$. $\mathcal{L}$ is also called the set of occurrences. Figure 7 formalises Notation 11. This is done by the position of the sub-context, which is a list of $\{1,2\}$ (also called an occurrence). In Figure 7 which implements the notion of sub-context, we write C/γ instead of $/(C,\gamma)$, which denotes the sub-context of C at γ. Similarly we define the sub-terms of a term by its occurrences of the sub-terms. $\gamma' \prec \gamma$ denotes γ' is a proper prefix of γ.

7.1. Substitutions of the context

There are several ways to think of a term t as a context filled with its sub-terms. We first define the substitutions of contexts with tuples of terms $\mathrm{Subst} \in (C \in \mathrm{Cont}; n \in \mathrm{N}; u \in \Lambda\sigma_0^n)\Lambda\sigma_0$ and implement C_q by $\mathrm{LiftCont} : (C \in \mathrm{Cont}; q \in \mathrm{N})\mathrm{Cont}$. We use $C[u]$ for $Subst(C, Lg(u), u)$

[8] Pages 33–53 of [30] are simply the French version of Sections 2–7 of [12].

$T_1 \equiv (C \in \text{Cont}; \gamma \in \mathcal{L}; B \in \text{Cont})\text{Cont}$	$T_2 \equiv (s \in \Lambda\sigma_0; \gamma \in \mathcal{L}; t \in \Lambda\sigma_0)\Lambda\sigma_0$
SubSubcon : T_1	SubSubtm : T_2
$C\{nil \leftarrow B\} = B$	$s\{nil \leftarrow t\} = t$
$\Box_m\{\{1,2\}\gamma \leftarrow B\} = \Box_m$	
$\text{id}_c\{\{1,2\}\gamma \leftarrow B\} = \text{id}_c$	$\text{id}\{\{1,2\}\gamma \leftarrow t\} = \text{id}$
$1_c\{\{1,2\}\gamma \leftarrow B\} = 1_c$	$1\{\{1,2\}\gamma \leftarrow t\} = 1$
$\uparrow_c \{\{1,2\}\gamma \leftarrow B\} =\uparrow_c$	$\uparrow \{\{1,2\}\gamma \leftarrow t\} =\uparrow$
$\lambda_c(C)\{\{1,2\}\gamma \leftarrow B\} = \lambda_c C\{\gamma \leftarrow B\})$	$\lambda(s)\{\{1,2\}\gamma \leftarrow t\} = \lambda(C\{\gamma \leftarrow t\}$
$C \cdot_c D\{1\gamma \leftarrow B\} = (C\{\gamma \leftarrow B\}) \cdot_c D$	$s_1 \cdot s_2\{1\gamma \leftarrow t\} = s_1\{\gamma \leftarrow t\} \cdot s_2$
$C \cdot_c D\{2\gamma \leftarrow B\} = C \cdot_c (D\{\gamma \leftarrow B\})$	$s_1 \cdot s_2\{2\gamma \leftarrow t\} = s_1 \cdot s_2\{\gamma \leftarrow t\}$
$C \circ_c D\{1\gamma \leftarrow B\} = (C\{\gamma \leftarrow B\}) \circ_c D$	$s_1 \circ s_2\{1\gamma \leftarrow t\} = s_1\{\gamma \leftarrow t\} \circ s_2$
$C \circ_c D\{2\gamma \leftarrow B\} = C \circ_c (D\{\gamma \leftarrow B\})$	$s_1 \circ s_2\{2\gamma \leftarrow t\} = s_1 \circ s_2\{\gamma \leftarrow t\}$

Figure 8. Formulation of substitutions in contexts and terms

(cf. Definition 12 and Notation 13)). We use u_k to denote $\text{Proj}(n, k, u)$, the *kth*-projection of the n-tuple u. We take $\text{K}_\text{c}(C)$ to be the number of the holes in C, and $\text{N}_\text{c}(C)$ to be the largest hole index in context C.

The next lemma states some basic facts about the substitution $C[u]$:

LEMMA 45. *Let* $u = (u_1, \cdots, u_m)$ *and* $v = (v_1, \cdots, v_n)$. *We have:*

1. $\text{Proj}(m+n, m+k, u@v) = \text{Proj}(n, k, v)$.
2. $C[u@v] = C[u]$ *if* $\text{N}_\text{c}(C) \leq Lg(u)$.
3. $C_m[u@v] = C[v]$.
4. *If* $\text{K}_\text{c}(C) = 0$, *then* $C[u] = C[v]$ *for any* u *and* v.

Intuitively they are all true, but to formally prove them, we need to check a lot of cases by induction. For all the ALF proofs see [1].

Substitutions in contexts is a basic operation in the context calculus. We use $C\{\gamma \leftarrow D\}$ for the context obtained by replacing in C the sub-context C/γ by the context D, and $s\{\gamma \leftarrow t\}$ for the term obtained by replacing in s the sub-term s/γ by the term t. We again abuse notation and write in the ALF definition, $C\{\gamma \leftarrow B\}$ instead of $\text{SubSubcon}(C, \gamma, B)$.[9] Substitution is given in Figure 8.

We consider both substitutions on variables (holes) and on "positions", where we only substitute some occurrences of a variable.

[9] $\text{TmtoCont}(s)$ will denote the context when thinking of term s as a context without any hole. We shall use $C\{\gamma \leftarrow t\}$ instead of $C\{\gamma \leftarrow \text{TmtoCont}(t)\}$.

7.2. Relations on contexts

In this section we formalise in ALF many of the notions of Section 2.3 (e.g., relative contexts, good (very good) contexts, inflations, etc.) and we prove in ALF many of the properties of these notions that are necessary for our main proof in ALF (in particular of Theorems 2 and 24). We will write all these formalisations in sugared notation and refer the reader to [1] for the full details and non sugared code.[10]

Let N_1 be the singleton set and $\mathbf{I}$ is the intensional equality

$$\begin{aligned} &\mathbf{I} \in (a, b \in A)\mathbf{Set} \\ &\quad r \in (A \in \mathbf{Set}; a \in A)I(a, a) \end{aligned}$$

The constructive definition of the relation relative of Definition 15, is defined inductively on the structure of C as follows:

$$\begin{aligned} &\mathrm{Relative} : (C : \mathrm{Cont}; s : \Lambda\sigma_0)\mathbf{Set} \\ &\quad \mathrm{Relative}(\Box_m, s) = \mathrm{N}_1 \\ &\quad \mathrm{Relative}(\mathrm{id}_c, s) = \mathbf{I}(s, \mathrm{id}) \\ &\quad \mathrm{Relative}(1_c, s) = \mathbf{I}(s, 1) \\ &\quad \mathrm{Relative}(\uparrow_c, s) = \mathbf{I}(s, \uparrow) \\ &\quad \mathrm{Relative}(\lambda_c(C), s) = \exists h(\mathbf{I}(s, \lambda(h)) \wedge \mathrm{Relative}(C, h)) \\ &\quad \mathrm{Relative}(C \cdot_c D, s) = \exists a, b(\mathbf{I}(s, a \cdot b) \wedge \mathrm{Relative}(C, a) \wedge \mathrm{Relative}(D, b)) \\ &\quad \mathrm{Relative}(C \circ_c D, s) = \exists a, b(\mathbf{I}(s, a \circ b) \wedge \mathrm{Relative}(C, a) \wedge \mathrm{Relative}(D, b)) \end{aligned}$$

Let $\mathcal{R}(C, s)$ stand for C is a context relative to s. Let $\mathcal{G}(C, s)$ (resp. $\mathcal{V}(C, s)$) stand for C is a good (resp. very good) context relative to s. The next lemma relates sub-contexts and sub-terms:

LEMMA 46 (Preservation of relative contexts in terms). *Let C, D be contexts, s, t be terms, u be a tuple of terms and γ be an occurrence.*

- *$\mathcal{R}(C, s)$ if and only if $\mathcal{R}(\lambda_c(C), \lambda(s))$.*
- *$\mathcal{R}(C \cdot_c D, s \cdot t)$ if and only if $\mathcal{R}(C, s)$ and $\mathcal{R}(D, t)$.*
- *$\mathcal{R}(C \circ_c D, s \circ t)$ if and only if $\mathcal{R}(C, s)$ and $\mathcal{R}(D, t)$.*
- *If $\mathcal{R}(C, s \text{ with } u)$ then $\mathcal{R}(C_q, s \text{ with } v@u)$ for any $v = (v_1, \cdots, v_q)$.*
- *If $D[u] = E[u]$ then $C\{\gamma \leftarrow D\}[u] = C\{\gamma \leftarrow E\}[u]$.*
- *If $\mathcal{R}(C, s)$ then $\mathcal{R}(C\{\gamma \leftarrow t\}, s\{\gamma \leftarrow t\})$.*

The notion of inflations of Definition 16 is defined as a relation in ALF in Figure 9 (where $\mathrm{WtmTuple}(w)$ means w is a W-Tuple, i.e. a tuple of W-terms). Figure 9 also defines a function in ALF to express the prime operation of Definition 16.

The next lemma relates context operations to inflations:

LEMMA 47 (Preservation of inflations in the structure of terms). *Let $\underline{C, D \text{ be}}$ contexts, s, t be terms, u, u_1, u_2, w, w_1, w_2 be tuples of terms.*

[10] Many lemmas of the context calculus were proved in ALF by analysing if a term is a W-term or L-term.

Inflation $\in (s \in \Lambda\sigma_0; C \in \text{Cont}; n \in \text{N}; u, w \in \Lambda\sigma_0^n))\mathbf{Set}$

$\quad$ Inflation$(s, C, n, u, w) \equiv (n \geq \text{NumofHoles}(C)) \wedge \mathbf{I}(s, C[u]) \wedge WtmTuple(w)$

$\tilde{\circ} : (u, w : \Lambda\sigma_0^n)\Lambda\sigma_0^n$

$\quad u\tilde{\circ}w = one$

$\quad < a, b > \tilde{\circ} < a_1, b_1 > = < a\tilde{\circ}a_1, IfThEl(b, b_1, b \circ b_1) >$

where $\quad IfThEl : (s \in \Lambda\sigma_0; t_1, t_2 : \Lambda\sigma_0)\Lambda\sigma_0$

$\quad\quad IfThEl(s, t_1, t_2) = t_1$ if s is a W-term

$\quad\quad IfThEl(s, t_1, t_2) = t_2$ otherwise

Figure 9. Inflations and the prime operation of Definition 16

GoodCont : $(C : \text{Cont}; s : \Lambda\sigma_0)Set$

$\quad$ GoodCont$(\Box_m, s) = \text{N}_1$

$\quad$ GoodCont$(\text{id}_c, s) = \mathbf{I}(s, \text{id})$

$\quad$ GoodCont$(1_c, s) = \mathbf{I}(s, 1)$

$\quad$ GoodCont$(\uparrow_c, s) = \mathbf{I}(s, \uparrow)$

$\quad$ GoodCont$(\lambda_c(C), s) = \exists h(\mathbf{I}(s, \lambda(h)) \wedge \text{GoodCont}(C, h))$

$\quad$ GoodCont$(C \cdot_c D, s) = \exists a, b(\mathbf{I}(s, a \cdot b) \wedge \text{GoodCont}(C, a) \wedge \text{GoodCont}(D, b))$

$\quad$ GoodCont$(C \circ_c D, s) = \exists a, b(\mathbf{I}(s, a \circ b) \wedge \text{GoodCont}(C, a) \wedge \text{GoodCont}(D, b) \wedge$

$\quad\quad\quad (((\exists l)\text{IsHole}(C, l)) \rightarrow$ W-hole(D)))

Figure 10. Good contexts Definition 18

- Inflation(s, C, u, w) *if and only if* Inflation$(\lambda(s), \lambda(C), u, w)$.
- *If* Inflation(s, C, u_1, w_1) *and* Inflation(t, D, u_2, w_2), *then* Inflation$(s \cdot t, C \cdot_c D_m, u_1@u_2, w_1@w_2)$.
- *If* Inflation(s, C, u_1, w_1) *and* Inflation(t, D, u_2, w_2), *then* Inflation$(s \circ t, C \circ_c D_m, u_1@u_2, w_1@w_2)$.
- *If* $\text{K}_c(C) = 0$ *then* Inflation$(C[u], C, n, u, w)$ *for any* u, w.

A good context (Definition 18) is defined as a set on C and s (see Figure 10) where IsHole(C, l) denotes C/l is a hole and W-hole(D) denotes that D is a W-hole. We shall also say a context C is good for a term s where we mean that C is a good context for s. A good Inflation Definition 20 will be defined in ALF as in Figure 11. A very good context (Definition 19) is defined in ALF as in Figure 12.

Many facts about good contexts are needed when proving some important lemmas in Section 7.3:

$$\text{GoodInflation} \in\in (s \in \Lambda\sigma_0; C \in \text{Cont}; n \in \mathrm{N}; u, w \in \Lambda\sigma_0^n))\mathbf{Set}$$
$$\text{GoodInflation}(s, C, n, u, w) \equiv \text{GoodCont}(C, s) \wedge \text{Inflation}(s, C, n, u, w)$$

Figure 11. Good Inflations Definition 20

$$\text{VeryGoodCont}(C, s) \text{ iff } \text{GoodCont}(C, s) \wedge ((\forall\gamma(\text{IsHole}(C, \gamma)) \to \forall(\delta \prec \gamma)\text{Lterm}(s/\delta))$$

Figure 12. Very Good Contexts of Definition 19

LEMMA 48. *Suppose that C, D are contexts, and s, t, c, d, e are terms.*

- *If $\mathcal{R}(C, s)$ and $\mathrm{K_c}(C) = 0$ then $\mathcal{G}(C, s)$.*
- *$\mathcal{G}(C, s)$ if and only if $\mathcal{G}(\lambda_c(C), \lambda(s))$.*
- *$\mathcal{G}(C \cdot_c D, s \cdot t)$ if and only if $\mathcal{G}(C, s)$ and $\mathcal{G}(D, t)$.*
- *If $\mathcal{G}(C \circ_c D, s \circ t)$ then $\mathcal{G}(C, s)$ and $\mathcal{G}(D, t)$.*
- *If $\mathcal{G}(C, s)$, $\mathrm{K_c}(C) = 0$, and $\mathcal{G}(D, t)$ then $\mathcal{G}(C \circ_c D, s \circ t)$.*
- *If $\mathcal{G}(C, s)$ and t is a W-term, then $\mathcal{G}(C \circ_c \Box_m, s \circ t)$.*
- *If $\mathcal{G}(C \circ_c D, a \circ b)$, $\mathrm{K_c}(C) \geq 1$, and $\mathcal{G}(E, e)$; then $\mathcal{G}(E \circ_c D, e \circ b)$.*
- *If $\mathcal{G}(C, s)$, then $\mathcal{G}(C_m, s)$ for any $m \in \mathrm{N}$.*
- *If $\mathcal{G}((C \cdot_c D) \circ_c E, (c \cdot d) \circ e)$, then $\mathcal{G}((C \circ_c E) \cdot (D \circ_c E), (c \circ e) \cdot (d \circ e))$.*
- *If $\mathcal{G}((C \circ_c D) \circ_c E, (c \circ d) \circ e)$ then $\mathcal{G}(D \circ_c E, d \circ e)$.*
- *If $\mathcal{G}((C \circ_c D) \circ_c E, (c \circ d) \circ e)$ and $\mathrm{K_c}(C) = 0$, then $\mathcal{G}(C \circ_c (D \circ_c E), c \circ (d \circ e))$.*

The following lemma will be needed when the main case of the Preservation Theorem 56 is checked:

LEMMA 49. *Let C, D be contexts, s, t be terms , γ be an occurrence and $m \in \mathrm{N}$.*

1. *If $\mathcal{G}(C, s)$, C/γ is a W-hole, t is a W-term, then $\mathcal{G}(C\{\gamma \leftarrow \Box_m\}, s\{\gamma \leftarrow t\})$. Moreover, $\mathcal{G}(C\{\gamma \leftarrow \Box_m\}, s)$.*
2. *If $\mathcal{G}(C, s)$, C/γ is λ-hole and $\mathcal{G}(D, t)$ then $\mathcal{G}(C\{\gamma \leftarrow D\}, s\{\gamma \leftarrow t\})$.*
3. *If $\mathcal{V}(C, s)$ then $\mathcal{V}(C/\gamma, s/\gamma)$.*
4. *If $\mathcal{V}(C, s)$, C/γ is a λ-hole, and t is an L-term, then $\mathcal{V}(C\{\gamma \leftarrow t\}, s\{\gamma \leftarrow t\})$.*
5. *Let $\mathcal{R}(C, s)$. If $\mathrm{K_c}(C) = 0$ then $\mathcal{V}(C, s)$.*
6. *If $\mathrm{K_c}(C) = 0$, $\mathcal{V}(C, s)$ and $\mathcal{V}(D, t)$. If $s \cdot t$ is an L-term then $\mathcal{V}(C \cdot_c D, s \cdot t)$. If $s \circ t$ is an L-term then $\mathcal{V}(C \circ_c D, s \circ t)$.*
7. *If $\mathcal{V}(C, s)$ and $C/\gamma(\gamma \neq nil)$ is a hole, then s is an L-term.*
8. *If $\mathcal{V}(C, s)$ then $\mathcal{V}(C_m, s)$ for any $m \in \mathrm{N}$.*
9. *If s is an L-term, t is a W-term and $\mathcal{V}(C, s)$ then $\mathcal{V}(C \circ_c \Box_m, s \circ t)$.*
10. *If $\mathcal{V}(C, s)$ then $\mathcal{V}(C\{\gamma \leftarrow \Box_m\}, s)$.*
11. *If $\mathcal{V}(C, s)$, C/γ is a $\lambda -$ hole, t is an L-term and $\mathcal{V}(D, t)$ then $\mathcal{V}(C\{\gamma \leftarrow D\}, s\{\gamma \leftarrow t\})$.*

7.3. Reduction of contexts

The main theorem in this section, the Preservation Theorem, basically says, the reduct of an increment of a strongly normalising term is still an increment of a strongly normalising term, and is smaller in some sense. Hence we are interested in the properties of context reduction. We define a notion of context reduction such that $C \to D$ if $C[u] \to D[u]$. This context reduction is defined as ($C \to D$ denotes $\mathrm{ContOneStep}(C, D)$):

$$\begin{array}{l}
\mathrm{ContOneStep} : (\mathrm{Cont}; \mathrm{Cont})Set \\
\quad \mathrm{id}_c \circ_c D \to D \\
\quad 1_c \circ_c \mathrm{id}_c \to 1_c \\
\quad 1_c \circ_c (C \cdot_c D) \to C \\
\quad \uparrow_c \circ_c \mathrm{id}_c \to \uparrow_c \\
\quad \uparrow_c \circ_c (C \cdot_c D) \to D \\
\quad \lambda_c(C) \circ_c D \to \lambda_c(C \circ_c (1_c \cdot_c (D \circ_c \uparrow_c))) \\
\quad (C \cdot_c D) \circ_c E \to (C \cdot_c E) \circ_c (D \cdot E) \\
\quad (C \circ_c D) \circ_c E \to C \circ_c (D \circ_c E) \\
\quad \lambda_c(C) \to \lambda_c(D) \text{ if } C \to D \\
\quad C \cdot_c D \to C' \cdot_c D \text{ if } C \to C' \\
\quad C \cdot_c D \to C \cdot_c D' \text{ if } D \to D' \\
\quad C \circ_c D \to C' \circ_c D \text{ if } C \to C' \\
\quad C \circ_c D \to C \circ_c D' \text{ if } D \to D'
\end{array}$$

LEMMA 50. *Suppose that C, D are contexts, γ is an occurrence.*

1. *f $C \to D$ and D/γ is a hole, then there is δ such that C/δ is a hole.*
2. *If $C \to D$ and $\mathrm{K_c}(C) = 0$, then $\mathrm{K_c}(D) = 0$.*

The next lemma is the formalisation of lemma 22. Figure 13 gives its explicit version in ALF.

LEMMA 51. *Let s' be the result of the good inflation (C, w) of $s = C[u]$, D be a context where $C \to D$, and $t = D[u]$ (hence, $s \to t$ and $s' \to t'$). There is a good inflation (D', w') of t whose result is t'.*

Proof: This is proved by induction on $Lg(C)$. Let us see how the case $C = A \circ B$ is proved. Let $s = C[u] = A[u] \circ B[u] = a \circ b$, and $m = |u|$. When $C = A \circ B \to D$, there are three possibilities according to the position of the redex. (We drop $_c$ when no confusion arises.)

1. The redex is within A. Suppose that $A \to E$, by I.H. there exists (E', u', w') which is the good inflation of $E[u]$, and $E'[u' \tilde{\circ} w'] = E[u \tilde{\circ} w]$. There are two cases: $\mathrm{K_c}(A) = 0$ or $\mathrm{K_c}(A) \geq 1$.

   a) $\mathrm{K_c}(A) = 0$. We have the following facts:

   – If $C \to D$ and $\mathrm{K_c}(C) = 0$, then $\mathrm{K_c}(D) = 0$. So $\mathrm{K_c}(E) = 0$.
   – $(E \circ B_q, u'@u, w'@w)$ is a good inflation $E[u] \circ B[u]$ by Lemmas 48 and 47, where $q = |u'|$.
   – The equality is checked by Lemma 45: $(E \circ B_q)[(u'@u)\tilde{\circ}(w'@w)] = (E \circ B_q)[(u'\tilde{\circ}w')@(u\tilde{\circ}w)] = (E[u'\tilde{\circ}w']) \circ (B[u\tilde{\circ}w]) = (E[u\tilde{\circ}w]) \circ (B[u\tilde{\circ}w]) = (E \circ B)[u\tilde{\circ}w]$.

b) $\mathrm{K_c}(A) \geq 1$. We have the following facts:

- B is a hole and $B[u]$ is a W-term by definition.
- $(E' \circ \Box_{m+1}, u'@B[u], w'@B[w])$ is a good inflation of $E[u] \circ B[u]$ by Lemmas 48 and 47.
- The equality is checked: $(E' \circ \Box_{m+1})[(u'@B[u])\tilde{\circ}(w'@B[w])] = (E'[u'\tilde{\circ}w'])\circ(B[u]\tilde{\circ}B[w]) = (E[u\tilde{\circ}w])\circ(B[u\tilde{\circ}w]) = (E\circ B)[u\tilde{\circ}w]$.

2. The redex is within B. Suppose $B \to E$. We have these facts:
   - If $C \to D$, then C is not a hole. Therefore B is not a hole.
   - If $C \circ D$ is good for $a \circ b$ and D is not a hole, then $\mathrm{K_c}(C) = 0$. Hence $\mathrm{K_c}(A) = 0$ here as $A \circ B$ is good and B is not a hole.

   By I.H. there is a good inflation (E', u', w') of b such that $b' = E'[u'\tilde{\circ}w'] = E[u\tilde{\circ}w]$. From Lemma 48 $A \circ E'_m$ is good for $s = A[u] \circ E[u]$. Now we have: $(A \circ E'_m)[(u@u')\tilde{\circ}(w@w')] = (A[u\tilde{\circ}w]) \circ (E'[u'\tilde{\circ}w']) = (A[u\tilde{\circ}w]) \circ (E[u\tilde{\circ}w]) = (A \circ E)[u\tilde{\circ}w]$. Hence $(A \circ E'_m, u@u', w@w')$ is the good inflation of $(A \circ E)[u]$ in the lemma.

3. The redex is $A \circ B$. We argue according to the rule:

   (Ass): $C = (E \circ F) \circ B$. Two cases arise:
   a) $\mathrm{K_c}(E) = 0$. We can prove that $(E \circ F) \circ B$ is good for $(a \circ b) \circ c$ implies that $E \circ (F \circ B)$ is good for $a \circ (b \circ c)$ (see Lemma 48). Thus $(E \circ (F \circ B), u, w)$ is the good inflation.

   b) $\mathrm{K_c}(E) \geq 1$. Then $F[u] \circ B[u]$ is a W-term, and the good inflation is $(E \circ \Box_{m+1}, u@(F[u] \circ B[u]), w@(F[w] \circ B[w]))$.

   (Abs): $C = (\lambda(E)) \circ B$ and $D = \lambda(E \circ (1 \cdot (B \circ \uparrow)))$. Two cases:
   a) If $\mathrm{K_c}(E) = 0$, $(\lambda(E \circ (1 \cdot B \circ \Box_{m+1})), u@\uparrow, w@\uparrow)$ is the good inflation.
   b) If $\mathrm{K_c}(E) \geq 1$, $(\lambda(E \circ \Box_{m+1}), u@(1 \cdot (B[u]\circ \uparrow)), w@(1 \cdot (B[w]\circ \uparrow))$ is the good inflation.

   (Map): $C = (E \cdot F) \circ B$ and $D = (E \circ B) \cdot (F \circ B)$. $((E \circ B) \cdot (F \circ B), u, w)$ is the good inflation, as by Lemma 48 if $(E \cdot F) \circ B$ is good for $(a \cdot b) \circ c$ then $(E \circ B) \cdot (F \circ B)$. $((E \circ B)$ is good for $(a \circ c) \cdot (b \circ c)$ and the equality is checked easily. $\Box$

The next lemma is the formalisation of Lemma 23. Figure 14 gives its explicit ALF version.

LEMMA 52. *If (C, u, w) is a good inflation of s with result s', then there exists a very good inflation (C', u', w') of s with result s' that* $\mathrm{K_c}(C') \leq \mathrm{K_c}(C)$.

Proof: Induction on the structure of C. $\mathrm{K_c}(C') \leq \mathrm{K_c}(C)$ ensures that induction can be done. (Note that $_c$ is dropped if no confusion arises.)

1. $C = \lambda(A)$. By I.H. there exists a very good inflation (A', u', w') for $s = A[u]$. Then we simply choose $(\lambda(A'), u', w')$.
2. $C = A \circ B$. Suppose that (A', u_A, w_B) and (B', u_B, w_B) are the very good inflations for $A[u]$ and $B[u]$ respectively, and $A'[u_A \tilde{\circ} w_A] = A[u \tilde{\circ} w]$ and $B'[u_B \tilde{\circ} w_B] = B[u \tilde{\circ} w]$. Two cases arise:

   a) $(A \circ B)[u]$ is a W-term, if there is no hole in $A \circ B$, then($A \circ B, u, w)$ is the very good inflation, otherwise $(\square_1, (A \circ B)[u], (A \circ B)[u \tilde{\circ} w])$ is the very good inflation. By Lemma 49, for the W-term $C[u]$, we always have $C[u] = C[u \tilde{\circ} w]$.

   b) $(A \circ B)[u]$ is an L-term.

   – If $\mathrm{K_c}(A') = 0$, then $A' \circ B'_q$ is very good for $A[u] \circ B[u]$ by 49, and $(A' \circ B'_q, u_A @ u_B, w_A @ w_B)$ is a good inflation for $A[u] \circ B[u]$, where $q = |u_A|$. The equality can be checked: $(A' \circ B'_q)[(u_A @ u_B) \tilde{\circ} (w_A @ w_B)] = (A' \circ B'_q)[(u_A \tilde{\circ} w_A) @ (u_B \tilde{\circ} w_B)]$ $= A'[u_A \tilde{\circ} w_A] \circ B'[u_B \tilde{\circ} w_B] = A[u \tilde{\circ} w] \circ B[u \tilde{\circ} w]$. $\mathrm{K_c}(A' \circ B'_q) \leq \mathrm{K_c}(A \circ B)$ because $\mathrm{K_c}(B'_q) = \mathrm{K_c}(B')$.

   – $\mathrm{K_c}(A') \geq 1$. Then $\mathrm{K_c}(A) \geq 1$, and B is a w-hole, this also implies that $A[u]$ is an L-term because $A \circ B$ is very good. $A \circ \square_{q+1}$ is a very good context for $A[u] \circ B[u]$ by Lemma 49. $(A \circ \square_{q+1}, u_A @ B[u], w_A @ B[w])$ is the very good inflation. The equality is checked easily. $\mathrm{K_c}(A' \circ \square_{q+1}) = \mathrm{K_c}(A') + 1 \leq \mathrm{K_c}(A \circ B)$ as $\mathrm{K_c}(B) = 1$.
3. $C = A \cdot B$. Similarly to the above case. □

Combining the last two lemmas, we get the following lemma which is needed in Lemma 56. See Figure 15 for the explicit ALF representation.

LEMMA 53. *Suppose that s' is the result of a very good inflation (C, u, w) of $s = C[u]$, $C \to D$, and $t' = D[u \tilde{\circ} w]$, then there exists a very good inflation (D', u', w') of $t = D[u]$ such that $D'[u' \tilde{\circ} w'] = D[u \tilde{\circ} w]$.*

Proof: This follows directly from Lemmas 51 and 52, and the ALF proof is not big, but it took some time to check. □
(Note: VeryGoodInflation$(C[u], A, u, w')$ = GoodInflation$(C[u], A, u, w')$ $\wedge VeryGood(A, C[u])$ and EX_4 denotes four existential quantifiers.)

Now let us see what is happening when a term $s = C[u]$ is reduced. Intuitively there are three possibilities: the redex is in C, the redex is in some u_k or there is an interaction between C and u. For our purpose and simplicity, we only consider the reduction $C[u \tilde{\circ} w] \to t$. In this case, three cases arise according to the occurrence of the redex:

1. The reduction is in the context, i.e. $C \to D$ and $t = D[u \tilde{\circ} w]$.

$Prop6 \in (C, D \in \mathrm{Cont}; n\ Nat; u, w \in \Lambda\sigma_0^n; C \to D; \mathrm{GoodInflation}(C[u], C, n, u, w))$

$$Aux_{Prop6}(C, D, n, u, w)$$

where

$Aux_{Prop6} \in (C, D \in \mathrm{Cont}; n\ Nat; u, w \in \Lambda\sigma_0^n)\mathbf{S}et$

$Aux_{Prop6} \equiv [C, D, n, u, w]EX4(\mathrm{Cont}, [h]\mathrm{N}, [h, h_1]\Lambda\sigma_0^{h_1}, [h, h_1, h_2]\Lambda\sigma_0^{h_1},$

$[h, h_1, h_2, h_3]\mathrm{GoodInflation}(D[u], h, h_1, h_2, h_3) \wedge \mathbf{I}(D[u\tilde{\circ}w], h[h_2\tilde{\circ}h_3])$

$EX4 \in (A \in \mathbf{S}et;$

$B \in (A)\mathbf{S}et;$

$C \in (h \in A; B(h))\mathbf{S}et;$

$D \in (h \in A; h_1 \in B(h); h_2 \in C(h, h_1))\mathbf{S}et;$

$D \in (h \in A; h_1 \in B(h); h_2 \in C(h, h_1); h_3 \in D(h, h_1, h_2))\mathbf{S}et$

$)\mathbf{S}et$

$$\frac{\mathrm{GoodInflation}(C, u, w); C \to D}{\mathrm{EX}_4(A, m, u', w')(\mathrm{GoodInflation}(A, u', w') \wedge \mathbf{I}(D[u\tilde{\circ}w], A[u'\tilde{\circ}w']))}$$

Figure 13. Encoding Lemma 51 (or Lemma 22) in ALF

$Prop7 \in (C, D \in \mathrm{Cont}; n\ Nat; u, w \in \Lambda\sigma_0^n; \mathrm{GoodInflation}(C[u], C, n, u, w))$

$$Aux_{Prop7}(C, n, u, w)$$

where $Aux_{Prop7} \in (C \in \mathrm{Cont}; n\ Nat; u, w \in \Lambda\sigma_0^n)\mathbf{S}et$

$Aux_{Prop7} \equiv [C, n, u, w]EX4(\mathrm{Cont}, [h]\mathrm{N}, [h, h_1]\Lambda\sigma_0^{h_1}, [h, h_1, h_2]\Lambda\sigma_0^{h_1}, [h, h_1, h_2, h_3])$

$\mathrm{GoodInflation}(C[u], h, h_1, h_2, h_3) \wedge \mathrm{VeryGoodCont}(h, C[u]) \wedge$

$\mathbf{I}(C[u\tilde{\circ}w], h[h_2\tilde{\circ}h_3]) \wedge \mathrm{K_c}(C) \leq \mathrm{K_c}(h)$

$$\frac{\mathrm{GoodInflation}(C, u, w)}{\mathrm{EX}_4(A, m, u', w')T}$$

where $T \equiv \mathrm{VeryGoodInflation}(C[u], A, u, w') \wedge \mathbf{I}(C[u\tilde{\circ}w], A[u'\tilde{\circ}w']) \wedge \mathrm{K_c}(C) \geq \mathrm{K_c}(A)$

Figure 14. Encoding Lemma 52 (or Lemma 23) in ALF

$Prop67 \in (C, D \in \mathrm{Cont}; n\ Nat; u, w \in \Lambda\sigma_0^n; C \to D; \mathrm{GoodInflation}(C[u], C, n, u, w))$

$$Aux_{Prop7}(D, n, u, w)$$

$$\frac{\mathrm{GoodInflation}(C, u, w); C \to D}{\mathrm{EX}_4(A, m, u', w')T}$$

where $T \equiv \mathrm{VeryGoodInflation}(D[u], A, u, w') \wedge \mathbf{I}(D[u\tilde{\circ}w], A[u'\tilde{\circ}w']) \wedge \mathrm{K_c}(D) \geq \mathrm{K_c}(A)$

Figure 15. Encoding Lemma 53 in ALF

2. The redex is in some hole $\square_k$, $(u \circ w)_k \to r$ and $t = C'[(u\tilde{\circ}w)@r]$ where C' is the context by replacing the square $\square_k$, where u_k is reduced to r, by $\square_{\mathrm{N_c}(C)+1}$;
3. There is an interaction between C and $u\tilde{\circ}w$. It is not easy to state this case clearly. There are three cases which may cause interaction:

   $\bullet(\square_m \circ D)[u\tilde{\circ}w] \to b.$ $\bullet(1 \circ \square_m)[u\tilde{\circ}w] \to b.$ $\bullet(\uparrow \circ\square_m)[u\tilde{\circ}w] \to b$

   If C is a very good context of $C[u]$, 3.b) and 3.c) turn to case 2. The case 3.a) happens only when the rule (Ass.) is applied, and $(u\tilde{\circ}w)_m = u_m \circ w_m$. This means there exists a very good inflation for $C[u]$ and the result is still $C[u\tilde{\circ}w]$, which is what we want for proving the Preservation Theorem.

LEMMA 54. *Let (C, u, w) is a very good inflation and $C[u\tilde{\circ}w] \to b$. One of the following holds:*
1. *There exists a context D such that $C \to D$ and $b = D[u\tilde{\circ}w]$.*
2. *There is $\gamma \in \mathcal{L}, c \in \Lambda\sigma_0$ such that C/γ is a hole, $C[u\tilde{\circ}w]/\gamma \to c$ and $b = (C\{\gamma \leftarrow \square_q\})[u@c]$.*
3. *There exist $D \in$ Cont$, n \in$ N$, u', w' \in \Lambda\sigma_0^n$ such that (D, u', w') is a very good inflation of $C[u]$, $b = D[u'\tilde{\circ}w']$, $Lg(D) > Lg(C)$ and $\mathrm{K_c}(D) \leq \mathrm{K_c}(C)$.*

Proof: By induction on the structure of C. Let us see how the lemma is proved when $C = A \circ B$. We prove this case by analysing the rule $(A \circ B)[u\tilde{\circ}w]$. We shall use the notation $C(n, \gamma) = C\{\gamma \leftarrow \square\}$.

1. The redex is in $A[u\tilde{\circ}w]$ and $A[u\tilde{\circ}w] \to b$. Three cases by I.H.:
   a) There exists a context A' such that $A \to A'$. Therefore $A \circ B \to A' \circ B$ and $b = (A' \circ B)[u\tilde{\circ}w]$.
   b) A/γ is a hole and $A[u]/\gamma \to a'$ and $b = A(n, \gamma)[(u\tilde{\circ}w)@a']$. Then $A \circ B/1\gamma$ is a hole, $(A \circ B)[u]/1\gamma \to a' \circ B[u]$ and $b \circ B[u\tilde{\circ}w] = ((A \circ B)\{1\gamma \leftarrow \square_k\})[(u\tilde{\circ}w)@a']$.
   c) There is a very good inflation $(A[u], A', u', w')$ such that $b = A'[u'\tilde{\circ}w']$. Two cases arise:
      i) $\mathrm{K_c}(A') = 0$. If $\mathrm{K_c}(B) = 0$, then we take the inflation $(A' \circ B, u, w)$. The result of the inflation is $(A' \circ B)[u'\tilde{\circ}w'] = b \circ B[u\tilde{\circ}w]$ because $\mathrm{K_c}(A') = 0$, $A'[u'\tilde{\circ}w'] = A'[u\tilde{\circ}w]$. Otherwise, we take the inflation $(A' \circ B_q, u'@u, w'@w)$. The result is $b \circ B[u\tilde{\circ}w]$. In both cases they are very good inflations by Lemma 49 with the same result $b \circ B[u\tilde{\circ}w]$.
      ii) $\mathrm{K_c}(A') \geq 1$. We take the inflation $(A' \circ \square_q, u'@B[u], w'@B[w])$. In this case, $\mathrm{K_c}(A) \geq 1$, hence B is W-hole. By Lemma 49, it is a very good inflation. The result is $b \circ B[u\tilde{\circ}w]$.

   For both cases the two equalities are true.

2. The redex is in $B[u\tilde{\circ}w]$, it is similar to the above case.
3. The redex is $(A \circ B)[u\tilde{\circ}w]$.

(IdL_c): $C = \mathrm{id} \circ B \to B$, $(\mathrm{id} \circ B)[u\tilde{\circ}w] \to B[u\tilde{\circ}w]$. Hence the first case of the lemma holds.

($VrId_c$): $C = 1 \circ \mathrm{id} \to 1$. The first case of the lemma holds.

($VrCons_c$): $C = 1 \circ (A \cdot B)$ and $1 \circ (A \cdot B) \to A$. The first case of the lemma holds.

($1 \circ \Box_k$): $C = 1 \circ \Box_k$. In this case we have the following facts:

– If $C \circ D$ is very good for $a \circ b$ and $\mathrm{K_c}(C \circ D) \geq 1$, then $a \circ b$ is an L-term; hence u_k is an L-term because $1 \circ \Box_k$ is very good for $1 \circ u_k$;

– $(1 \circ \Box_k)[u\tilde{\circ}w] = 1 \circ (u_k \circ w_k)$ because u_k is an L-term and hence $(u\tilde{\circ}w)_k = u_k \circ w_k$

– If $1 \circ (a \circ b) \to c$, then $c = 1 \circ d$ where $a \circ b \to d$. This concludes that the redex is in $(u\tilde{\circ}w)_k = u_k \circ w_k$.

Therefore the second case of the lemma holds, $\gamma = 2$ and $1 \circ (u_k \circ w_k)/2 \to c$.

($\uparrow \circ \Box_k$): $C = \uparrow \circ \Box_k$, it is the same as the case above.

($ShId_c$): $C = \uparrow \circ \mathrm{id} \to \uparrow$. The first case of the lemma holds.

(ShCons): $C = \uparrow \circ (A \cdot B) \to B$. The first case of the lemma holds.

(Interaction): $C = \Box_k \circ B$. The following facts were proved:

– B is a W-hole and u_k is an L-term as $\Box_k \circ B$ is very good.

– $b = u_k \circ (w_k \circ B[u\tilde{\circ}w])$

– $(u_k \circ \Box_{n+1}, u@B[u], w@(w_k \circ B[w]))$ is a good inflation.

– $u_k \circ \Box_{n+1}$ is a very good context for $(u_k \circ \Box_{n+1})[u]$.

– $b = (u_k \circ \Box_{n+1})[u\tilde{\circ}w]$.

– $Lg(u_k \circ \Box_{n+1}) = Lg(u_k) + 1 > 1 = Lg(\Box_k \circ B)$.

– $\mathrm{K_c}(u_k \circ \Box_{n+1}) = 1 < 2 = \Box_k \circ B$.

All these facts mean that the third case of the lemma holds. Now, each of the cases below imply the first case of the lemma.

(Abs): $C = \lambda(A) \circ B \to \lambda(A \circ (1 \cdot (B \circ \uparrow)))$.

(Map): $C = (A \cdot B) \circ E \to (A \circ E) \cdot (B \circ E)$.

(Ass): $C = (A \circ B) \circ E \to A \circ (B \circ E)$. $\Box$

REMARK 55. *The second statement was changed many times in order to present all the information when it is applied. What we should present is the most original information which can derive other information when it is needed. In this case, the most original information is presented in terms of the "position". The first statement is:*

There exist $\gamma \in \mathcal{L}, k \in \mathrm{N}, c \in \Lambda\sigma_0$ such that $(u\tilde{\circ}w)_k \to c$ and $b = (C\{\gamma \leftarrow \Box_q\})[u@c]$.

Then one need the information that C/γ is a hole, and $k \leq n = |u|$ and $C[u\tilde{\circ}w]/\gamma = (u\tilde{\circ}w)_k$, which we have when the lemma is proved. However, it is not enough still. One need to say that $C[u]/\gamma = u_k$ and $C[w]/\gamma = w_k$, which we can not get from the revised statement. In fact all the information is stored in the following statement:

There exist $\gamma \in \mathcal{L}, c \in \Lambda\sigma_0$ such that C/γ is a hole, and $C[u\tilde{\circ}w]/\gamma \to c$ and $b = (C\{\gamma \leftarrow \Box_q\})[u@c]$.

The following lemma, which is proved by analysing the position of the redex of s' based on Lemma 54, enables us to use induction on the triple $\theta_{s,s'} = (dp(s), lg(s) - lg(C), \Sigma\mu_k dp(w_k))$ to prove Theorem 24.

LEMMA 56. *If s' is the result of a very good inflation (C, u, w) of some term s, and $s' \to t'$, then:*

- *there exists a term t such that $s \to t$ and t' is the result of a very good inflation of t; or*
- *t' is the result of a very good inflation (D, a, b) of s and $lg(D) > lg(C)$; or*
- *there exists some term r such that $w_k \to r$ and t' is the result of the very good inflation $(C', u@u_k, w@r)$ of s where $C' = C\{\gamma \leftarrow \Box_n\}$ and $n = Lg(u) + 1$.*

Proof: This is proved by analysing the position of the redex of s' based on Lemma 54. We will prove that for any t' such that $s' = C[u\tilde{\circ}w] \to t'$, t' can be the result of a very good inflation of some t such that $\theta_{t,t'} < \theta_{s,s'}$, so the I.H. can be applied. There are three cases by Lemma 54:

1. The reduction is in the context. By Lemma 53, $t = D[u]$, $s \to t$ and there is a very good inflation (D', u', w') of t. The result remains t'.
2. The reduction is in some hole of context C, i.e. there is $\gamma \in \mathcal{L}, k \in \mathrm{N}$ such that $C[u\tilde{\circ}w]/\gamma = (u\tilde{\circ}w)_k$ and $(u\tilde{\circ}w)_k \to t'$. Let $C' = C\{\gamma \leftarrow \Box_n\}$. Two cases arise: $(u\tilde{\circ}w)_k$ is either an L- or a W-term.
   a) $(u\tilde{\circ}w)_k$ is an L-term. In this case, $(u\tilde{\circ}w)_k = u_k \circ w_k$ and u_k must be an L-term. We argue according to the redex. Five cases arise:
      (Ass): Let $u_k = a \circ b$, and so, $s = C'[u@(a \circ b)]$ and $s' = C'[(u\tilde{\circ}w)@((a \circ b) \circ w_k)]$. Then $t' = C'[(u\tilde{\circ}w)@(a \circ (b \circ w_k))]$, where $C' = C\{\gamma \leftarrow \Box_n\}$. Take $C'' = C\{\gamma \leftarrow a \circ \Box_n\}$ and the inflation $(C[u], C'', u@a, w@w_n)$, where $w_n = \mathrm{IfThEl}(b, b \circ w_k, w_k)$, which is a W-term. The next facts (which give the 2nd case of Lemma 56) are proved:
      - $C[u]\{\gamma \leftarrow (a \circ \Box_n)[u@b]\} = C[u]$
      - C'' is a very good context of $C[u]$ by Lemma 49;

- $(C[u], C'', u@b, w@w_k)$ is a very good inflation;
- $t' = C'[(u\tilde{\circ}w)@(a \circ (b \circ w_k))] = C''[((u@b)\tilde{\circ}(w@w_n))]$;
- $Lg(C'') > Lg(C)$.

(Abs): Take $u_k = \lambda(a)$, $C'' = C\{\gamma \leftarrow \lambda\Box_n\}$ and the inflation $(C[u], C'', u@a, w@w_n)$ where $w_n = \text{IfThEl}(a, a \circ (1 \cdot (w_k \circ \uparrow)), 1 \cdot (w_k \circ \uparrow))$, which is a W-term. The next facts (which give the 2nd case of Lemma 56) are proved:
- $C[u]\{\gamma \leftarrow (\lambda\Box_n)[u@a]\} = C[u]$.
- C'' is good for $C[u]$ by Lemma 49.
- C'' is very good for $C[u]$ by lemma 49.
- $Lg(C'') > Lg(C)$.

(Map): Take $u_k = a{\cdot}b$, $C'' = \{\gamma \leftarrow \Box_n{\cdot}\Box_{n+1}\}$ and the inflation $I = (C[u], C'', u@ < a, b >, w@ < w_n, w_{n+1} >)$, where $w_n = \text{IfThEl}(a, a{\circ}w_k, w_k)$, $w_{n+1} = \text{IfThEl}(b, b{\circ}w_k, w_k)$. We have the next facts (which give the 2nd case of Lemma 56):
- $C[u]\{\gamma \leftarrow a \circ b\} = C[u]$.
- C'' is good for $C[u]$ by Lemma 49.
- C'' is very good for $C[u]$ by Lemma 49.
- $(C[u], C'', u@ < a, b >, w@ < w_n, w_{n+1} >)$ is a very good inflation.
- $C''[(u@ < a, b >)\tilde{\circ}(w@ < w_n, w_{n+1} >)] = C'[(u\tilde{\circ}w)@((a \circ w_k) \cdot (b \circ w_k))] = t'$.
- $Lg(C'') > Lg(C)$.

(ComL): The redex is in u_k, and $u_k \to a$. Let $t = C'[u@a]$. Then $s \to t$. By Lemma 49 C' is a good context for t.

i) If a is an L-term. $(C\{\gamma \leftarrow \Box_n\}, u@a, w@w_k)$ is a very good inflation with the result $(C\{\gamma \leftarrow \Box_n\})[(u\tilde{\circ}w)@(a \circ w_k)]$.

ii) If a is a W-term, $C\{\gamma \leftarrow \Box_n\}$ is a good context for $t = (C\{\gamma \leftarrow \Box_n\})[u@a]$, then $(C\{\gamma \leftarrow \Box_n\}, u@a, w@(a \circ w_k)$ is a good inflation. By Lemma 52 there is a very good inflation with the same result. Therefore the first case of the Lemma holds.

(ComR): In this case, $w_k \to b$, and $(C[u], C', u@u_k, w@b)$ is the very good inflation with the result $C'[(u\tilde{\circ}w)@(u_k \circ b)]$. The third case of the lemma holds.

b) $(u\tilde{\circ}w)_k$ is a W-term, hence u_k is also a W-term. So $(u\tilde{\circ}w)_k = w_k$, $w_k \to b$, and b is a W-term. By Lemma 49 $C\{\gamma \leftarrow \Box_n\}$ is very good for $C[u](\gamma, b)$. Then the inflation $(C\{\gamma \leftarrow \Box_n\}, u@u_k, w@b)$ is very good, and the result is $(C\{\gamma \leftarrow \Box_n\})[(u\tilde{\circ}w)@w_k]$.

3. For the 3rd case of Lemma 54, use the 2nd case of the lemma. $\Box$

Now we are in the position to prove Theorem 24. See Figure 16 for its ALF representation.

Proof: By induction over a triple $\theta_{s,s'} = (dp(s), lg(s) - lg(C), \Sigma\mu_k dp(w_k))$ where $\mu_k = card\{\gamma : C/\gamma = \Box_k\}$, i.e. the number of occurrences of $\Box_k$

$$\begin{array}{l}
\text{Preservation-Th} \;\in (u, w \in \Lambda\sigma_0^n; \text{GoodInflation}(C[u], C, n, u, w); \\
\qquad\qquad\qquad\qquad \text{VeryGoodCont}(C, C[u]); \text{SN}(C[u]))\text{SN}(C[u\tilde{\circ}w]) \\
\quad \text{Preservation-Th}(u, w, h, h_1, \text{SN-intr}(-, h_3)) \equiv \\
\qquad \text{SN-intr}(C[u\tilde{\circ}w][b, h_2] \\
\qquad \text{Or-elim}([h_4]\text{SN}(b), \\
\qquad\quad [x]\text{Preservation-Th1}(u, w, b, h, h_1, \text{SN-intr}(C[u], h_3)h_2, x) \\
\qquad\quad [h_4]\text{Preservation-Th2}(u, w, b, h, h_1, \text{SN-intr}(C[u], h_3)h_2, h_4) \\
\qquad\quad\; \text{IH-Preservation}(C, n, u, w, b, h, h_1, h_2)))
\end{array}$$

Figure 16. Preservation Theorem

in C. By Lemma 56, there are three cases. 1) b is a very good inflation of t and the first component decreases and t is strongly normalising,hence I.H. applies. 2) b is the result of a very good inflation of s and the second component decreases, hence I.H. applies. 3) b is a result of a very good inflation of s and the third component decreases, so I.H. applies. $\square$

8. Formalising the termination proof of σ in ALF

Now we give the strong normalisation proofs of σ_0 and σ in ALF.

8.1. Termination of σ_0 in ALF

The strong normalisation of σ_0 is proved by the elimination rule of $\Lambda\sigma_0$. The difficulty is in the proof $e_5 : (a, b : \Lambda\sigma_0; \text{SN}(a); \text{SN}(b))\text{SN}(a \circ b)$, or more precisely when coming to the rule: $(Abs) : \quad (\lambda s) \circ t \to \lambda(s \circ (1 \cdot (t\circ \uparrow)))$. If we prove $(*)$: If $s \in \text{SN}$ then $s\circ \uparrow \in \text{SN}$, then induction on $(dpth(s), lgth(s), dpth(t), lgth(t))$ gives the proof object e_5. To solve $(*)$, we introduced a context calculus in Section 7. Now we give the proof of $(*)$ by the Preservation Theorem. It is easy to prove that reduction which does not involve the rule (Abs) is strongly normalising.

LEMMA 57. *Let $s, t \in \Lambda\sigma_0$.*

1. *$\lambda(t)$ are strongly normalising if and only if t is strongly normalising.*
2. *$s \cdot t$ is strongly normalising if and only if s and t are strongly normalising.*
3. *If $s\circ t$ is strongly normalising, then s and t are strongly normalising.*
4. *Let s, t be W-terms. If s and t are strongly normalising, then $s \circ t$ is strongly normalising.*
5. *If $t\circ \uparrow$ is strongly normalising whenever t is, then $s \circ t$ is strongly normalising for any strongly normalising terms s and t.*

6. Any W-term is strongly normalising.

In ALF, this lemma was proved by induction on depth and length. We gave the details of the induction proof in Section 6 when we proved the termination of the calculus s. Now we come to the lemma where the Preservation Theorem is used and the problem $(*)$ is solved:

LEMMA 58. *Let s be a L-term, then $s\circ\uparrow$ is the result of the very good inflation $(\Box_1,\uparrow)$ of s. So if s is strongly normalising then $s\circ\uparrow$ is strongly normalising. In ALF:* $\mathit{Com\_SN3} \in (s \in \Lambda\sigma_0; \mathit{Lterm}(s); \mathrm{SN}(s)))\mathrm{SN}(s\circ\uparrow)$

It is easy to see that $s\circ\uparrow$ is the result of a very good inflation $(\Box_k,\uparrow)$ of s when s is not a W-term, and it is strongly normalising by the preservation theorem. If s is a W-term, then $s\circ\uparrow$ is also a W-term, and it is strongly normalising. Hence we have the following lemma:

LEMMA 59. *If s is strongly normalising, then $s\circ\uparrow$ is strongly normalising. In ALF:* $\mathit{Com\_SN4} \in (s \in \Lambda\sigma_0; \mathrm{SN}(s))\mathrm{SN}(s\circ\uparrow)$

The next lemma gives the proof object e_5. It is proved by induction on $(dpth(s), lgth(s), dpth(t), lgth(t))$

LEMMA 60. *If s, t are strongly normalising, then $s \circ t$ is strongly normalising. In ALF:*

$$\begin{array}{l}
\mathit{Com\_SN5} \in (s,t \in \Lambda\sigma_0; \mathrm{SN}(s); SN(t)))SN(s \circ t)\\
\mathit{Com\_SN5}(s,t,h,h_1) \equiv\\
\qquad \mathit{Com\_SN1}(\mathit{Forall\_intr}([x]\mathit{Imply\_intr}([h_2]\mathit{Com\_SN4}(x,h_2)), s,t,h,h_1)\\
\mathit{where}\ \mathit{Com\_SN1} \in\\
(\mathit{Forall}(\Lambda\sigma_0,[h]\mathit{Imply}(SN(h), SN(h\circ\uparrow))); s,t \in \Lambda\sigma_0; SN(s); SN(t))SN(s \circ t)
\end{array}$$

Having got all the proof objects $e_1, \cdots, e_6$, we prove Theorem 7, the strong normalisation of σ_0 by induction on the structure of σ_0-terms:

$$\begin{array}{l}
SN \in (s \in \Lambda\sigma_0)SN(s)\\
\quad SN(1) \equiv L4W(1, c1)\\
\quad SN(id) \equiv L4W(id, c1)\\
\quad SN(\uparrow) \equiv L4W(\uparrow, c1)\\
\quad SN(\lambda(a)) \equiv L4\lambda(a, SN(a))\\
\quad SN(s_1 \circ t) \equiv \mathrm{ComSN5}(s_1, t, SN(s_1), SN(t))\\
\quad SN(App(s_1,t)) \equiv L4App(s_1, t, SN(s_1), SN(t))\\
\text{where}\\
\quad L4W \in (a \in \Lambda\sigma_0; Wterm(a))SN(a)\\
\quad L4\lambda \in (a \in \Lambda\sigma_0; SNs0(a))SN(\lambda(a))\\
\quad L4App \in (a,b \in \Lambda\sigma_0; SN(a); SNs0(b))SN(App(a,b))
\end{array}$$

8.2. Termination of σ in ALF

Having proved that the calculus σ_0 is terminating, now we can prove that the calculus σ is terminating. Let F' and F'' be the translations in ALF of the function F from σ to σ_0 in Definition 5, where $F' \in (\Lambda\sigma^t)\Lambda\sigma_0$ $F'' \in (\Lambda\sigma^s)\Lambda\sigma_0$ are defined in the obvious way following Definition 5. Then, we get the strong normalisation of σ by checking that the translation is really a strict interpretation from σ to σ_0, i.e. by proving Lemma 6 in ALF:

$$S_1 \in (a, b \in \Lambda\sigma^t, a \to b)F'(a) \to F'(b)$$
$$S_2 \in (a, b \in \Lambda\sigma^s, a \to b)F''(a) \to F''(b)$$

Hence, finally, we have:

THEOREM 61. *The calculus σ is strongly normalising.*

9. Conclusion and future work

In this paper we gave formal proofs of the strong normalisation of σ and s in ALF. To prove σ is strongly normalising, we formalised the notions and checked all the proofs of [12]. Some of these proofs were informal and needed to be checked formally, e.g. Lemma 54. For the calculus s, we gave three formal proofs of strong normalisation, which follow the usual ways of proving strong normalisation of explicit substitutions. Two of these proofs are given for the first time in this paper.

The formalisation of this paper lead us to remark that:

- During formalisations, one has to explain how to move from the classical logic used in informal proofs to a constructive logic. For example, Lemma 36 is easy to prove using classical logic because one can resort to the definition of the absence of infinite derivations. However, in a proof checker based on the Pat principle, it is not possible to use the classical laws of refutation and hence, proofs are constructive and are done via introduction and elimination rules. See the ALF proof of Lemma 36 discussed in Section 6.3.

- Many of the intuitively true statements required checking a lot of details in ALF. For example, Projection$_1$ – Projection$_3$ in Section 6.3) needed much details some of which are given in Section 6.3. Similarly, Lemma 45 lists some basic facts about substitution that are intuitively true but we needed to check a lot of cases by induction to be able to formally prove them.

- The context calculus of [12] demanded much work during the formalisation. In [12, 30], many notions were not introduced, and many lemmas were left unproven. To formalise the proofs in [12] we had to rewrite all the intuitions and informal notions, and to check a lot of details. We often had to change the implementation to make the proofs go through. For example, the second statement of Lemma 54 was changed several times in order to present all the details when this lemma is applied (see Remark 55).

A lot of work has been done on proof checking in various proof checkers (e.g., ALF, Coq, Lego). Advantages of this work include:

1. Helping people prove theorems whose proofs are cumbersome. In the proof process, one only gives some orders to the prover and the prover carries out the detailed computations and reasoning. For instance, when filling a hole, the user can give only the name of the lemma and the prover itself will fill in all the parameters by unifications. However, one needs to do more work to give a formal proof. In the strong normalisation proof of σ, many lemmas are true intuitively, but involve much work to prove formally. For instance, in Lemma 54, the position of a redex can occur in three cases. In Lemma 49, when replacing a λ-hole in a context C, which is very good for s, with a L-term t, the resulting context is very good for the term replacing the sub-term at the same position with t.

2. Investigating the processes of mathematical proofs, to help people understand mathematical reasoning and to build automatic theorem provers. In fact, it is during the implementation of theorem provers and during the checking of proofs that one understands more about mathematical proofs. In the case of termination of explicit substitutions, we hope to understand why decreasing measures can be found for some calculi but not for others.

It is interesting to find a general way to prove properties of explicit substitutions such as strong normalisation, confluence and preserving β-strong normalisation and to develop a package of special tools to deal with calculi of explicit substitutions, e.g. to help researchers to prove the above properties. In our formalisation, we tried to remain as general as possible with the intention that our proof should be adapted to other existing calculi of substitutions. The fact that during our formalisation of the four different proofs of termination we shared a lot of implemented proofs, means that our work can well be adapted to formalising other proofs of terminations of other substitution calculi.

Of course, there is the question of portability of our proofs of this paper to other theorem provers. Although we did not attempt to run our proofs on any of these other provers, we believe that the top level (that is filling in the intuition by formal details) can be used by any other prover, given the right translation between the formalism of that other prover and ALF. And, looking at those lemmas whose proofs in ALF depart from the proofs on paper (like those mentioned above, Lemmas 36, 45, 54 and 49), it seems that they can be dealt with similarly in a prover based on PAT such as Coq.

References

1. ftp://ftp.cs.chalmers.se/users/cs/qiao/Sigma/.
2. M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.
3. B. Barras. *Auto-validation d'un système de preuves avec familles inductives.* Thèse de doctorat, Université Paris 7, November 1999.
4. Z. Benaissa, D. Briaud, P. Lescanne, and J. Rouyer-Degli. $\lambda\nu$, a calculus of explicit substitutions which preserves strong normalisation. *Journal of Functional Programming*, 6(5):699–722, September 1996.
5. R. Bloo and K. Rose. Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. In *CSN-95: Computer Science in the Netherlands*, November 1995.
6. M. Bognar and R. de Vrijer. The context calculus lambda-c. *Workshop on Logical Frameworks and Meta-languages*, 1999.
7. R. L. Constable, S. Allen, H. Bromely, W. Cleveland, et al. *Implementing Mathematics with the Nuprl Development System.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1986.
8. Project Coq. The Coq proof assistant reference manual, version 6.1. Technical report, INRIA, 1996.
9. C. Coquand. From semantics to rules: A machine assisted analysis. In E. Börger, Y. Gurevich, and K. Meinke, editors, *Proceedings of the 7th Workshop on Computer Science Logic*, pages 91–105. Springer-Verlag LNCS 832, 1993.
10. P.-L. Curien. *Categorical Combinators, Sequential Algorithms, and Functional Programming.* Progress in Theoretical Computer Science. Birkhäuser, Boston, 2nd edition, 1993. (1st ed., Pitman Publishing, London, and J. Wiley and Sons, New York).
11. P-L Curien, T. Hardin, and J-J. Lévy. Confluence properties of weak and strong calculi of explicit substitutions. Technical Report RR 1617, INRIA, Rocquencourt, 1992.
12. P-L Curien, T. Hardin, and A. Ríos. Strong normalisation of substitutions. *Logic and Computation*, 6:799–817, 1996.
13. R. David and B. Guillaume. The lambda l calculus. *Second International Workshop on Explicit Substitutions, Theory and Applications*, 1999.
14. N. de Bruijn. A namefree lambda calculus with facilities for internal definition of expressions and segments. Technical report, Department of Mathematics , University of Eindhoven, Netherlands, 1978.
15. M. J. C. Gordon and T. F. Melham. *Introduction to HOL: A theorem proving environment for higher order logic.* Cambridge University Press, 1993.

16. T. Hardin and A. Laville. Proof of termination of the rewriting system SUBST on CCL. *Theoretical Computer Science*, 46(2-3):305–312, 1986.
17. T. Hardin and J.-J. Lévy. A confluent calculus of substitutions. *France-Japan Artificial Intelligence and Computer Science Symposium*, December 1989.
18. T. Hardin, L. Maranget, and B. Pagano. Functional runtime systems within the lambda-sigma calculus. *Functional Programming*, 8(2):131–176, 1998.
19. M. Hashimoto and A. Ohori. A typed context calculus. *Type theory and its applications to computer systems (Japanese)*, 1023:76–91, 1998.
20. F. Kamareddine and R. P. Nederpelt. On stepwise explicit substitution. *International Journal of Foundations of Computer Science*, 4(3):197–240, 1993.
21. F. Kamareddine and A. Ríos. A λ-calculus a la de Bruijn with explicit substitutions. In *PLILP95, Lecture Notes in Computer Science*, volume 982, pages 45–62. Springer-Verlag, 1995.
22. F. Kamareddine and A. Ríos. Extending a λ-calculus with explicit substitution which preserves strong normalisation into a confluent calculus on open terms. *Journal of Functional Programming*, 7(4):395–420, July 1997.
23. F. Kamareddine and A. Ríos. Relating the $\lambda\sigma$- and λs-styles of explicit substitutions. *Logic and Computation*, 10(3):349–380, 2000.
24. L. Magnusson. *The Implementation of ALF—A Proof Editor Based on Martin-Löf's Monomorphic Type Theory with Explicit Substitution.* PhD thesis, Chalmers University of Technology and Göteborg University, January 1995.
25. P. Martin-Löf. An intuitionistic theory of types. *Logic Colloquium '73*, 1975.
26. Yukiyoshi Kameyama Masahiko Sato, Takafumi Sakurai. A simply typed context calculus with first-class environments. *Fifth International Symposium on Functional and Logic Programming, FLOPS'01*, 2001.
27. J. McKinna and R. Pollack. Pure type systems formalised. In M. Bezem and J. F. Groote, editors, *Proceedings 1st Intl. Conf. on Typed Lambda Calculi and Applications, TLCA'93, Utrecht, The Netherlands, 16–18 March 1993*, volume 664 of *Lecture Notes in Computer Science*, pages 289–305. Springer-Verlag, Berlin, 1993.
28. C. Muñoz. Confluence and preservation of strong normalisation in an explicit substitutions calculus (extended abstract). In *Proceedings of the Eleven Annual IEEE Symposium on Logic in Computer Science*, New Brunswick, New Jersey, July 1996. IEEE Computer Society Press.
29. L. C. Paulson. Isabelle: The next 700 theorem provers. In Piergiorgio Odifreddi, editor, *Logic and Computer Science*, pages 361–386. Academic Press, 1990.
30. A. Ríos. *Contribution à l'étude des λ-calculs avec substitutions explicites.* PhD thesis, Université de Paris 7, 1993.
31. A. Saïbi. Formalisation of a λ-calculus with explicit substitutions in Coq. In P. Dybjer, B. Nordström, and J. Smith, editors, *Proceedings of the International Workshop on Types for Proofs and Programs*, pages 183–202, Båstad, Sweden, June 1994. Springer-Verlag LNCS 996.
32. B. Nordström T. Altenkirch, V. Gaspes. A user's guide to alf. Technical report, University of Göteborg, 1994.
33. H. Zantema. Termination of term rewriting: interpretation and type elimination. *Journal of Symbolic Computation*, 17(1):23–50, January 1994. Conditional term rewriting systems (Pont-à-Mousson, 1992).
34. H. Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24:89–105, 1995.