Proof of termination of the rewriting system SUBST on CCL

Thérèse Hardin, A. Laville

CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.(1) 39 63 55 11

Rapports de Recherche

N° 560

PROOF OF TERMINATION OF THE REWRITING SYSTEM SUBST ON CCL

Thérèse HARDIN
Alain LAVILLE

Août 1986

Proof of Termination of the Rewriting System SUBST on CCL

Thérèse Hardin

Université de Reims et L.I.T.P.

Alain Laville

Université de Reims et I.N.R.I.A.

Résumé

Le système de réécriture SUBST de la Logique Combinatoire Catégorique permet la simulation de la substitution du λ-calcul avec couples explicites. Ce système est localement confluent mais les méthodes classiques pour s'assurer de la terminaison échouent sur ce système.

Dans ce rapport, nous indiquons une nouvelle méthode permettant de démontrer la confluence de SUBST et d'obtenir ainsi la confluence de ce système.

Abstract

The rewriting system SUBST of the Combinatory Categorical Logiç allows the simulation of substitution of the λ-calculus with explicit couples. This system is locally confluent but classical methods used to show termination cannot conclude here.

In this report, we indicate a new method which is able to prove termination of SUBST and so to get Church-Rosser property for this system.

Proof of Termination of the Rewriting System SUBST on CCL

Thérèse Hardin

Université de Reims et L.I.T.P.

Alain Laville

Université de Reims et I.N.R.I.A.

In [4], P.L. Curien defines a translation of the λc-calculus in the Pure Combinatory Categorical Logic and etablishes an equivalence theorem between these two theories. The rewriting system SUBST simulates in particular the substitution of the λc-calculus. This system is locally confluent. We show here that it is also noetherian.

1. Introduction, definitions, notations

CCL, the Pure Combinatory Categorical Logic, is the algebra of terms built over the following signature :

App, F, S and I	of arity zero, respectively called application, first projection, second projection and identity.
Λ	of arity one, called currying.
<,> and o	of arity two, which are the operations of pairing and of composition (with infix notation).

The rewriting system SUBST, on CCL, is defined by the rules :

$$(s \circ t) \circ u \to s \circ (t \circ u) \qquad \text{(Ass)}$$

$$I \circ t \to t \qquad \text{(IdL)}$$

$$t \circ I \to t \qquad \text{(IdR)}$$

$$F \circ \langle s,t\rangle \to s \qquad \text{(Fst)}$$

$$S \circ \langle s,t\rangle \to t \qquad \text{(Snd)}$$

$$\langle s,t\rangle \circ u \to \langle s \circ u , t \circ u\rangle \qquad \text{(DPair)}$$

$$\langle F \circ t , S \circ t\rangle \to t \qquad \text{(SPair)}$$

$$\langle F,S\rangle \to I \qquad \text{(FSI)}$$

$$\Lambda(s) \circ t \to \Lambda(s \circ \langle t \circ F , S \rangle) \qquad \text{(DΛ)}$$

The system PROD obtained by removing from SUBST the rule (DΛ) can easily be showed terminating with, for example, a Recursive Path Ordering [1]. But the classical orderings used to show termination : R.P.O, R.D.O., Knuth-Bendix, multi-set and the polynomial interpretations ([3], [5], [6], [7], [9], [10]) cannot orientate the rule (DΛ). As far as we know, the recently developped methods in [2] and [11] do not seem to be suitable to prove the termination. The orderings currently implemented in rewriting laboratories fail to show termination of the SUBST system.

In order to prove this termination, we define on CCL a fonction $P_{D\Lambda}$ such that, for any t, and for any derivation D (using the rules of SUBST) of t, the number of applications of the rule (DΛ) in D is bounded by $P_{D\Lambda}(t)$.

2. The terms describing functions

Notations

A derivation of a term t is a sequence of reductions of t. The graph of t is the set of terms derived from t. It is noted G(t).

A symbol is said potential in a term t if it appears in one of the elements of G(t).

2.1. Definition of $P_{D\Lambda}$

The function $P_{D\Lambda}(t)$ is defined by induction on the structure of t as follows:

[1] $P_{D\Lambda}(t) = 0$ if t is App, F, S or I

[2] $P_{D\Lambda}(\Lambda(s)) = P_{D\Lambda}(s)$

[3] $P_{D\Lambda}(< s , u >) = P_{D\Lambda}(s) + P_{D\Lambda}(u)$

The principal problem is to define $P_{D\Lambda}(s \circ u)$. To give an upper bound to the number of (DΛ)-redexes in a such term, we have to take into account :

1) those which are contained in s and in u. Their number are respectively $P_{D\Lambda}(s)$ and $P_{D\Lambda}(u)$. Furthermore, the (DΛ)-redexes of u can be duplicated by reduction of the (Dpair)-redexes created by the potential pairs of s with the symbol "o" at the top of s o u. So we have to estimate the maximal number of potential pairs in a term. We shall do that with a new function P_p ($P_p(t)$ will be the maximal number of potential pairs in t, increased with 1) .

2) those created by the symbol of composition at the top of term and the potential Λ's in s. Therefore, we shall define another function P_Λ to compute the number of maximal potential Λ's in a term.

3) Moreover reductions of those (DΛ)-redexes pointed out in 2) give subterms u o F. Thus any potential Λ in u can create a (DΛ)-redex with this context "o F" and these redexes can also be duplicated by the potential pairs in s.

How can we estimate the number of duplications owed to the potential pairs in s ? If we are looking only at the pairs, s is like a binary tree. Composition with u is only lifting down u to the leaves of the tree, distributing u along every node. Thus the number of duplications by s is equal to the number of leaves of this tree : it is the number of nodes increased by 1.

With these two functions P_Λ and P_p (which we define later), we complete the definition of $P_{D\Lambda}$ as follows :

$$[4] \quad P_{D\Lambda}(s \circ u) = \begin{array}{l} P_{D\Lambda}(s) \\ + P_{D\Lambda}(u) \times P_p(s) \\ + P_\Lambda(s) \\ + P_\Lambda(s) \times P_\Lambda(u) \times P_p(s) \end{array}$$

Example

Let s, t, u be three terms $\in$ CCL. Let $A \equiv (\Lambda(\langle t,s\rangle) \circ u) \circ D$, where D is $\Lambda(F)$ for example.

$A \dashrightarrow (\Lambda(\langle t,s\rangle \circ \langle u \circ F,S\rangle) \circ D \dashrightarrow \Lambda((\langle t,s\rangle \circ \langle u \circ F,S\rangle) \circ \langle D \circ F,S\rangle)$

$\dashrightarrow \Lambda(\langle t \circ (\langle u \circ F,S\rangle \circ \langle D \circ F,S\rangle), s \circ (\langle u \circ F,S\rangle \circ \langle D \circ F,S\rangle)\rangle)$

The reduction of these Dpair-redexes creates four copies of the sub-term D o F, and so, four $D\Lambda$-redexes.

2.2. Definition of P_Λ

Defining the function P_Λ is not very difficult : the only way to create a symbol Λ is to duplicate an already present Λ with the rule (DPair).

P_Λ is defined by induction on the structure of terms as follows :

[1] $P_\Lambda(t) = 0$ if t is App, F, S or I
[2] $P_\Lambda(\Lambda(s)) = 1 + P_\Lambda(s)$
[3] $P_\Lambda(\langle s,u\rangle) = P_\Lambda(s) + P_\Lambda(u)$
[4] $P_\Lambda(s \circ u) = P_\Lambda(s) + P_\Lambda(u) \times P_p(s)$

The function P_p will be defined later. We study before some properties of functions P_Λ and $P_{D\Lambda}$. To do that, we will suppose that P_p verifies some properties, exhibited during the proofs.

2.3. Properties of $P_{D\Lambda}$ and P_Λ

A function f on CCL, into an ordered set, is said to be *compatible* with the structure of terms if:

For any t, t', for any context C[], if $f(t) \geq f(t')$ then $f(C[t]) \geq f(C[t'])$.

f has the *sub-term property* if:

for any sub-term s of t, $f(s) \leq f(t)$.

The functions $P_{D\Lambda}$, P_Λ and P_p have the sub-term property but are not compatible with the structure of terms as showing by the following examples.

Examples

Let $t = <\Lambda(F),\Lambda(F)>$ and $s = \Lambda^n(F)$. s contains no pair and t contains one. Moreover these terms are in normal form. The term (t o F) contains only one potential pair but the term (s o F) contains n such pairs.

Let $t = \Lambda(F)oF$ and s be the same as above. t contains a (DΛ)-redex and s contains none. The term (t o F) contains 2 and the term (s o F) contains n such redexes.

We shall assume in the following that the function P_p verifies conditions, which we call (P1), (P2), (P3), (P4). These conditions will be stated as they are needed.

Let (P1) be the following condition:

P_p is a function into **N** verifying:

1) the sub-term property
2) If $t \in G(s)$ then $P_p(s) \geq P_p(t)$ (and thus for any context C[], $P_p(C[s]) \geq P_p(C[t])$).

Kamin and Lévy [8] pointed out that, in order to verify the monotonicity of P_p, it suffices to test it only on the rewrite rules when this kind of condition is satisfied (instead of full compatibility with the structure of terms).

Proposition 1

Let $t \in G(s)$. If the property (P1) is verified and furthermore, we have :

$P_\Lambda(s) \geq P_\Lambda(t) \quad P_{D\Lambda}(s) \geq P_{D\Lambda}(t)$,

then for any context C, we have:

$P_\Lambda(C[s]) \geq P_\Lambda(C[t]) \quad P_{D\Lambda}(C[s]) \geq P_{D\Lambda}(C[t])$

Proof

By induction on the structure of the terms, first for P_Λ next for $P_{D\Lambda}$.

Proposition 2

For any term t derived from the term s, we have the inequality:

$$P_\Lambda(t) \leq P_\Lambda(s)$$

Proof

According to the previous proposition, we only have to compute the respective values of the left and right members of every redex.

1) (Spair), (Fst), (Snd), (IdL), (IdR), (FSI)

Straightforward by using the sub-term property.

2) (DPair)

Then $s = < u , v > o\ w$ and $t = < u\ o\ w , v\ o\ w >$. By definition of P_Λ, we have :

$$P_\Lambda(s) = P_\Lambda(u) + P_\Lambda(v) + P_\Lambda(w) \times P_p(< u , v >)$$

So we ask P_p to verify the following condition :

$$P_p(<u,v>) = P_p(u) + P_p(v) \qquad \text{(P2)}$$

($P_p(t)$ is intended to be the number of potential pairs in t increased with 1)

We now compute $P_\Lambda(t)$:

$$P_\Lambda(t) = P_\Lambda(u) + P_\Lambda(w) \times P_p(u) + P_\Lambda(v) + P_\Lambda(w) \times P_p(v)$$

whence :

$$P_\Lambda (s) = P_\Lambda (t)$$

3) (Ass)

We have $s = (u\ o\ v)\ o\ w$ and $t = u\ o\ (v\ o\ w)$. We get :

$P_\Lambda(s) = P_\Lambda(u) + P_p(u) \times P_\Lambda(v) + P_p(uov) \times P_\Lambda(w)$
$P_\Lambda(t) = P_\Lambda(u) + P_p(u) \times P_\Lambda(v) + P_p(u) \times P_p(v) \times P_\Lambda(w)$

So we ask P_p to verify the following condition :

$$P_p\ (u\ o\ v) \geq P_p(u) \times P_p(v) \qquad \text{(P3)}$$

With (P3), we obtain the result.

4) (DΛ)

We have $s = \Lambda(u)\ o\ v$ and $t = \Lambda\ (\ u\ o < v\ o\ F , S>)$.

If P_p verifies the condition:

$$P_p(\Lambda(s)) = P_p(s) \qquad \text{(P4)}$$

we get the equality of $P_\Lambda(s)$ and of $P_\Lambda(t)$. ▪

Theorem

Let D be a derivation of the term s to the term t . Then :

$$P_{D\Lambda}(t) \leq P_{D\Lambda}(s)$$

Furthermore if D contains one application of the rule $(D\Lambda)$, then this inequality is stict.

Proof

We prove this proposition rule by rule.

1) (Spair), (Fst), (Snd), (IdL), (IdR), (FSI)
Straightforward by using proposition 1

2) (DPair)
We use property (P2) to show: $P_{D\Lambda}(s) \geq P_{D\Lambda}(t)$.

3) (Ass)
Using definitions of $P_{D\Lambda}$ and of P_Λ and the property (P3), we get :

$$P_{D\Lambda}((s \circ t) \circ u) \geq P_{D\Lambda}(s \circ (t \circ u))$$

4) $(D\Lambda)$
We compute the values of $P_{D\Lambda}$ on the two members .

$$\begin{aligned} P_{D\Lambda}(\Lambda(s) \circ t) = {} & P_{D\Lambda}(\Lambda(s)) \\ & + P_{D\Lambda}(t) \times P_p(\Lambda(s)) \\ & + P_\Lambda(s) + 1 \\ & + (P_\Lambda(s) + 1) \times P_\Lambda(t) \times P_p(\Lambda(s)) \end{aligned}$$

$$\begin{aligned} P_{D\Lambda}(\Lambda(s \circ < t \circ F, S >)) = {} & P_{D\Lambda}(s) \\ & + [P_{D\Lambda}(t) + P_\Lambda(t)] \times P_p(s) \\ & + P_\Lambda(s) \\ & + P_\Lambda(s) \times P_\Lambda(t) \times P_p(s) \end{aligned}$$

Using property (P4), we deduce:

$$P_{D\Lambda}(\Lambda(s) \circ t) = P_{D\Lambda}(\Lambda(s \circ < t \circ F, S >)) + 1$$

So this function really computes the maximal number of applications of rule $(D\Lambda)$. ▪

So we still have to define the function P_p , intended to compute the number of potential pairs in a term and verifying the previous four properties.

3. Definition and properties of the function P_p

The difficulty is the definition of P_p (s o u). If we look at the left member of the rule (DΛ), we get the feeling that the left son Λ(s) of the composition should be much more heavy that the right son t. But this right son becomes a left son in the right member. Moreover we have to take the simplification rules into account (derive Λ(F) o t for example). The following example suggests how symbols Λ can create pairs.

Example

Let $M = \Lambda^n(F) \circ \Lambda^p(F)$ where Λ^n denotes a sequence of n Λ. We construct a derivation of M. We use n times the rule (DΛ) and we get :

$\Lambda^n(F \circ <..<\Lambda^p(F) \circ F,S> \circ F,S>,..> \circ F,S>)$

containing n pairs embedded in each other. After n (Dpair)-reductions, we get:

$\Lambda^n(F \circ <..<(..(\Lambda^p(F) \circ F) \circ F) \circ F)..) \circ F,(S \circ F)..) \circ F>,(S \circ F) \circ F).. \circ F>,..>,S>)$

where the term $\Lambda^p(F)$ is topped by n symbols o. After the reductions of (n×p)(DΛ)-redexes created in this way, we get:

$\Lambda^n(F \circ <..<(..(\Lambda^p((F \circ H), \ldots ,) \circ H),(S \circ F)..) \circ F>,(S \circ F) \circ F).. \circ F>,..>,S>)$

where H is a term containing only symbols F, S, o and p pairs. After some dressing with the rules (Ass) and (Dpair), we get a term under Λ^p containing $((1+p)^n - 1)$ pairs.

This example suggests to build a function P_p looking like:

$P_p(s \circ u) = P_p(s) + (1+P_p(s)) (P_p(u)+P_\Lambda(s) + (1+P_\Lambda(u))^{P_\Lambda(s)})$

But this function can be strictly increased by application of rule (Ass). In fact, this sort of formula supposes that all pairs in the left son act together on the right son. We can see that this is false on the following example:

$(<\Lambda^n(F) , \Lambda^m(F) > \circ \Lambda^p(F))$

where the factor $(1+p)^{n+m}$ is really bigger than the number of potential pairs and should be replaced by $(1+p)^n + (1+p)^m$.

We are led to introduce an auxiliary function L, defining a list intended to represent the potential pairing structure of the term by the binding depths of its potential leaves.

3.1. Definition of the auxiliary function L

This function associates with a term, a list of integers. We define it by induction on the structure of terms (the lists of integers are noted between brackets : [1,2,3] or $[s_1, \ldots, s_n]$ for example).

[1] $L(t) = [0]$ if t is App, F, S or I

With the notations $L(s) = [s_1, \ldots, s_n]$ and $L(t) = [t_1, \ldots, t_p]$) :

[2] $L(\Lambda(t)) = [1+t_1, \ldots, 1+t_p]$

[3] $L(\langle s, t \rangle) = [s_1, \ldots, s_n, t_1, \ldots, t_p]$

[4] $L(s \circ t)$ is the list composed with the following elements :

- any s_i repeated s_i times
- for any possible value of index i and j, s_i+t_j repeated $(1+t_j)^{s_i}$ times.

We shall write $|L(t)|$ the length of the list $L(t)$. We write $L(t) \subset L(t')$ if every element of L appears at least as many times in L' .i.e the lists are representations of multisets ordered with inclusion.

<u>Example</u>

Let $N = (\Lambda^2(F) \circ \Lambda^2(F)) \circ \Lambda^2(F)$. We compute $L(M)$:

$L(\Lambda^2(F)) = [2]$

$M = L(\Lambda^2(F) \circ \Lambda^2(F)) = [2, 2, 4$ repeated $(1+2)^2$ times]

In the precedent example, we have shown that this term M can be rewritten to a term containing 10 pairs.

$L(N) =$ [2 repeated 2 times, 2 repeated 2 times, 4 repeated (4x9) times,
4 repeated $(1+2)^2$ times, 4 repeated $(1+2)^2$ times,
6 repeated $(1+2)^{4 \times 9}$ times]

So, $|L(N)| = 787$. Effectively, there exists a term in the graph of N, which possesses 786 pairs !

3.2. Properties of L

Proposition

With respect to the ordering $\subset$ on lists, L is compatible with the structure of terms but does not verify the sub-term property. However if $t' \in G(t)$, L verifies:

$$L(t') \subset L(t)$$

<u>Proof</u>

Compatibility of L is proved by induction on the structure of the context. Looking at the term $\Lambda(t)$, we notice that L does not verify the subterm property. L being compatible, we only have to look at a reduction at the top of the term.

1) (Fst), (Snd), (IdL), (IdR), (Dpair), (SPair), (FSI)

Straightforward by using the definition of list.

2) (DΛ)
We construct the lists associated with the two members of the rule:
Let $L(s) = [s_1, \ldots, s_n]$ $L(t) = [t_1, \ldots, t_p]$.

L (Λ(s) o t) is a list which contains exactly, for any possible value of index i and j :

- the element $1+s_i$ repeated $1+s_i$ times

- the element $1+s_i+t_j$ repeated $(1+t_j)^{1+s_i}$ times

We construct L (Λ (s o < t o F , S >) in several steps.

The list L (t o F) contains exactly the elements t_j each of them repeated $1+t_j$ times.

L (< t o F , S >) is deduced from the previous by adding a 0.

Therefore L (s o < t o F , S >) contains (for any possible i and j) :

- s_i repeated s_i times

- s_i+t_j repeated $(1+t_j)^{s_i}$ times for any element t_j of the previous list. Now t_j is repeated $(1+t_j)$ times. Therefore s_i+t_j is repeated $(1+t_j)^{1+s_i}$ times.

- again a copy of any s_i because of the element 0

We now get L (Λ (s o < t o F , S >) by adding 1 to any element of the previous list and we obtain exactly L (Λ(s) o t)

4) (Ass)
We use 3 terms s, t and u such that: $L(s) = [s_1, \ldots, s_n]$, $L(t) = [t_1, \ldots, t_p]$ and $L(u) = [u_1, \ldots, u_q]$.

We compute L ((s o t) o u) :

L (s o t) contains any s_i repeated s_i times and any s_i+t_j repeated $(1+t_j)^{s_i}$ times. So L ((s o t) o u) contains the following elements :

- any s_i repeated $s_i \times s_i$ times

- any s_i+t_j repeated $(1+t_j)^{s_i} \times (s_i+t_j)$ times

- any s_i+u_k repeated $s_i \times (1+u_k)^{s_i}$ times

- any $s_i+t_j+u_k$ repeated $(1+u_k)^{s_i+t_j} \times (1+t_j)^{s_i}$ times.

Now we compute L (s o (t o u)). This list contains the following elements:

- any s_i repeated s_i times

- any s_i+t_j repeated $(1+t_j)^{s_i} \times t_j$ times

- any $s_i+t_j+u_k$ repeated $(1+u_k)^{t_j} \times (1+t_j+u_k)^{s_i}$ times.

As $(1+t_j+u_k)$ is less than $(1+t_j) \times (1+u_k)$, this list is extracted from the previous one. ▪

3.3. Definition and properties of the function P_p

For any term t, we define : $P_p(t) = |L(t)|$

Proposition

P_p verifies the properties (P1) to (P4)

Proof

One checks easily that:

$P_p(t) = 0$ if t is App, F, S ou I

(P2) $P_p(<s,t>) = P_p(s) + P_p(t)$

(P3) $P_p(sot) \geq P_p(s) \times P_p(t)$
since this property can be rewritten $|L(sot)| \geq |L(s)| \times |L(t)|$. Now the number $(1+t_j)^{s_i}$ is strictly positive thus L(sot) contains at least one time each element s_i+t_j.

(P4) $P_p(\Lambda(t)) = P_p(t)$

Therefore P_p takes its values in **N**. Thanks to inclusion of lists, the property (P1) is then completely verified. We can remark that P_p verifies the sub-term property but it is not compatible with the structure of terms.

Acknowledgements

We would like to thank the members of the project FORMEL at INRIA for warmly welcoming us during our sabbatical year. We enjoyed very much the fructful discussions we had with them.

Bibliography

[1] BELLEGARDE F. Utilisation des Systèmes de Réécriture d'Expressions Fonctionnelles comme Outils de Transformation de Programmes Itératifs. Thèse d'Etat. Université de Nancy I (1985).

[2] BELLEGARDE F. LESCANNE P. Termination Proofs Based On Transformation Techniques. Centre de Recherche en Informatique de Nancy (to appear).

[3] BEN CHERIFA A. LESCANNE P. Termination of Rewriting Systems by Polynomial Interpretations and Its Implementation. Centre de Recherche en Informatique de Nancy (to appear).

[4] CURIEN P.L. Categorical Combinators, Sequential Algorithms and Functional Programming, Pitman Ed. London (1985)

[5] DERSHOWITZ N. Orderings for Term-Rewriting Systems. Theoretical Computer Science 17 (1980)

[6] DERSHOWITZ N. Termination. in Proc. First Conf. Rewriting Techniques and Applications Dijon (France) (1985). L.N.C.S. vol 202 Springer-Verlag

[7] DERSHOWITZ N. MANNA Z. Proving Termination with Multi-Set Orderings CACM 22 (1979)

[8] KAMIN S. and LEVY J.J. Two Generalisations of Recursive Path Ordering. Unpublished note (1980). See also : HUET G. Computation and Deduction (Course notes Carnegie Mellon University 1986)

[9] HUET G. and OPPEN D. Equations and Rewrite Rules: A Survey in Formal Languages: Perspectives and Open Problems Ed. Book R. (1980)

[10] LESCANNE P. Some properties of Decomposition Ordering, A Simplification Ordering to Prove Termination of Rewriting Systems. R.A.I.R.O Theoretical Informatics vol 16 no 4 1982

[11] PUEL L. Using Unavoidable Sets of Trees to Generalize Kruskal's Theorem (to appear)

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique