Institute of Informatics, Warsaw University

Linear interpolation for the higher order matching problem

Aleksy Schubert
alx@mimuw.edu.pl

TR 96-16 (237)
December, 1996

Institute of Informatics
Warsaw University
ul. Banacha 2
02-097 Warsaw, Poland
phone: (48-22) 658-31-65
fax: (48-22) 658-31-64

Druk: Zakład Graficzny UW zam.nr. .

Contents

Linear interpolation for the higher order matching problem[1]

Aleksy Schubert
alx@mimuw.edu.pl

December, 1996

Abstract

We present here a particular case of the higher order matching problem — the linear interpolation problem. The problem consists in solving a collection of higher order matching equations of the shape $xM_1 \ldots M_k = N$, where x is the only unknown quantity. We prove recursive equivalence of the higher order matching problem and the linear interpolation problem. We also investigate decidability of a special case of the fifth order linear interpolation problem. The restriction we consider consists in that arguments of variables from the main abstraction in terms $M_1, \ldots, M_k$ cannot contain variables from the main abstraction.

1 Preface

The higher-order matching problem for simply typed λ-calculus has been considered since 1976 ([5]). There were proposed several partial solutions of the problem (second order matching — [4]; correct, but without a proof of completeness, algorithm — [7]; third order matching — [3]; fourth order matching — [6]).

In this paper, we present a special case of the higher order matching problem — the linear interpolation problem. The problem consists in solving a collection of higher order matching equations of the shape $xM_1 \ldots M_k = N$ where x is the only unknown quantity. We prove here recursive equivalence of the higher order matching problem and the linear interpolation problem.

This problem is interesting since to construct a solution for such a problem we deal with a single object, not a set of objects as in the matching problem in general formulation of is taken into account. Moreover, V. Padovani investigates a similar problem in his paper [6]. The Padovani's problem consists in solving the pair of sets $\{\Phi, \Psi\}$ of interpolation equations. A solution of such a problem is a concretisation of unknown quantities which satisfies each equation in the set

[1]This work has been partly supported by ESPRIT BRA 7232 GENTZEN, and KBN 8 T11C 034 10 grants.

Φ and does not satisfy any equation in the set Ψ. Decidability of the problem implies decidability of the matching problem as proven in [6].

In the second part of the paper, we look into decidability of a special case of the fifth order linear interpolation problem. The restriction we consider is that arguments of variables from the main abstraction in terms $M_1, \ldots, M_k$ cannot contain occurrences of variables from the main abstraction.

This issue is interesting, since it gives constructors of proof-checkers and proof-assistants possibility of solving some fifth order matching equations.

This paper is organised as follows — in Section 2 we present some basic definitions and define some useful notation, in Section 3 we prove recursive equivalence of the higher-order matching problem and the interpolation problem, and in Section 4 we prove our decidability result.

Acknowledgements. I would like to thank prof. J. Tiuryn for encouragement to deal with the higher-order matching problem and for discussions on the topic we had. I also thank prof. P. Urzyczyn, Robert Maron, Grzegorz Grudziński for many prolific debates on the λ-calculus.

2 Basic definitions

2.1 Types and terms

We assume the reader is familiar with the notions of λ-term, β and η reduction, type systems. Corresponding definitions can be found in [1] or [2].

Additionally, we understand the simply typed λ-calculus in the formulation, where we have a type indexed set $\mathcal{E}$ of unknown quantities symbols, and terms may contain these symbols in unbound positions. The set of all simply-typed λ-terms is denoted by $\Lambda_{\rightarrow}$. The set of free variables in a term t is denoted by $FV(t)$ and the set of constants by $\mathrm{Const}(t)$.

We assume, except when stated explicitly, all terms are in β-normal, η-long form. When necessary the normal form of a term t is denoted by $\mathrm{NF}(t)$.

Moreover, we mean by *a closed term* a term defined as usual but we impose one additional condition — the term cannot contain unknown quantities. We denote by $\mathrm{Closed}(A)$ the restriction of the set A of terms to closed terms.

Terms are denoted by capital Latin letters (for instance $A, D, M \ldots$) and by small Latin letters starting from s ($s, t, \ldots$). Types are denoted by small Greek letters starting from σ ($\sigma, \tau, \ldots$). We denote by $\mathrm{Types}(t)$ the set of all types of subterms of the term t. $\mathrm{SubTypes}(A)$ denotes the set of all the subtypes of types from A.

Definition 1 `(order of a type)`
The order of a type *is an integer computed according to the following rules:*

- $\mathrm{ord}(\alpha) = 1$, *where α is a base type,*
- $\mathrm{ord}(\sigma \rightarrow \tau) = \max(\mathrm{ord}(\sigma) + 1, \mathrm{ord}(\tau))$.

2.2 Böhm trees

Definition 2 `(Böhm trees for` λ`-terms)`
Let

$$t \;=\; \lambda x_1 \ldots x_n . v M_1 \ldots M_k$$

be a term in the normal form. The Böhm tree of the term t *is a tree* T *with its root labelled by a pair* (l, v)*, where* l *is a sequence of variables* $\{x_i\}_{i \in \{1,\ldots,n\}}$*, and with edges leading from the root to Böhm trees* T_i *for* $i \in \{1, \ldots, k\}$ *of terms* M_i*.*

A path in such a Böhm tree *is a sequence of numbers of subsequent descendants in the tree.*

Definition 3 `(a subterm at the path)`
Subterm at the path γ in a term t *is the term*

$$v M_1 \ldots M_n$$

where v *is the second component of the label at the last position in the sequence* γ *and* M_i *for* $i \in \{1, \ldots, n\}$ *are terms corresponding to sons of the node to which* γ *leads.*

Definition 4 `(an occurrence)`
The path γ *is* an occurrence of the symbol v *iff the last label in* γ *has the form*

$$(l, \;\; v)$$

for a certain sequence l*.*

The number of all the occurrences in the term t *is denoted by* $|t|$ *and the set of all the occurrences —* Occurences(t)*.*

A path γ *is* an occurrence of the subterm s *iff the subtree of the Böhm tree of* t *beginning at the last label of* γ *is a Böhm tree of* s *after replacing the sequence in the first node by the empty sequence.*

From now on, except when stated explicitly, we use the name "term" to refer to the Böhm tree of the term in question.

Definition 5 `(a graft)`
A graft of the term s in the term t at the path γ *is the name of the term* t' *(denoted by* $t[s \to \gamma]$*) that is equal to* t *except for the ending of* γ *and all the continuations of the path. The subtree at the ending of* γ *is a modified Böhm tree of* s*. This modification consist in changing the label at the root of* s *so that if the ending of* γ *has the form* (l, v) *and the root of* s *—* (l', v') *then the modified root has the label* $(l@l', v')$*, where @ is a concatenation operator.*

Now, we present a fact that will be used in Section 4 of the paper.

Fact 2.1 `(recursively enumerable set of terms)`
For each collection $\langle \mathcal{T}, C, g, \sigma \rangle$*, where*

- *$\mathcal{T}$ is a finite set of types,*
- *C is a type indexed set of constants with finite set C_σ for each type σ,*
- *g is a natural number,*
- *σ is a type*

the set of closed λ-terms of the type σ in β-normal, η-long form such that

- *each subterm (not subtree) has its type in $\mathcal{T}$,*
- *each constant is in C,*
- *each path (in the Böhm tree) has length bounded by g*

is finite and there exists an algorithm that given a collection of the form

$$\langle \mathcal{T}, C, g, \sigma \rangle$$

computes the set of such terms.

Proof:
We give an algorithm that generates the described above set of terms. Termination of the algorithm will imply the finiteness of the set.

Let us assume

$$\sigma = \ \sigma_1 \to \ldots \to \sigma_n \to \iota$$

where ι is a base type.

Algorithm A($\mathcal{T}$, C, g, σ):

1. If $g = 1$ then return as a result the constants from C that are of type σ.
2. If $g > 1$ then the term t has the form

$$\lambda x_1 \ldots x_n . v M_1 \ldots M_{k(v)} \tag{1}$$

   where x_i for $i \in \{1, \ldots n\}$ are local variables, v is either a constant or one of x_i for $i \in \{1, \ldots n\}$, and M_i for $i \in \{1, \ldots k(v)\}$ are terms of suitable types. We have to generate all the terms of such a shape

   (a) For each constant D that has its type in $\mathcal{T}$ and the form of the type is

   $$\tau = \ \tau_1 \to \ldots \to \tau_{k(v)} \to \iota,$$

   call A($\mathcal{T}$, $C \cup \{x_1, \ldots x_n\}$, $g-1$, τ_i) for $i \in \{1, \ldots k(v)\}$. Use the resulting sets A_{τ_i} for $i \in \{1, \ldots k(v)\}$ to get

   $$T_c \ = \ \{\lambda x_1 \ldots x_n . v M_1 \ldots M_{k(v)} \mid M_i \in A_{\tau_i} \text{ for } i \in \{1, \ldots k(v)\}\}$$

   (b) For other constants put $T_c = \emptyset$.

(c) For x_i with the type τ in $\mathcal{T}$, where

$$\tau = \tau_1 \to \ldots \to \tau_{k(v)} \to \iota,$$

call $\mathrm{A}(\mathcal{T},\ C \cup \{x_1, \ldots x_n\},\ g-1,\ \tau_j)$ for $j \in \{1, \ldots k(x_i)\}$. Use the resulting sets A_{τ_j} for $j \in \{1, \ldots k(v)\}$ to get

$$T_{x_i} = \{\lambda x_1 \ldots x_n . v M_1 \ldots M_{k(x_i)} \mid M_j \in A_{\tau_j} \text{ for } j \in \{1, \ldots k(x_i)\}\}$$

The result is

$$\bigcup_{c \in C} T_c \cup \bigcup_{i=1}^{n} T_{x_i}.$$

Obviously, all the generated sets are finite (a proof by induction on g). The correctness of the algorithm is a consequence of the fact that if the depth of a closed term is 1 then it must be a constant and if the depth is greater than 1, then it must have the form (1). □

2.3 The matching problem

Now we introduce the definition of the *higher-order matching problem.*

Definition 6 `(a higher-order matching equation)`
Let $M : \tau$ and $N : \tau$ be closed λ-terms where $N : \tau$ does not contain any unknown quantity. The equation

$$M : \alpha = N : \alpha,$$

where α is a base type, is called higher-order matching equation.

Definition 7 `(a concretisation of unknown quantities)`
Any type-respecting function $\rho : \mathcal{E} \to \mathrm{Closed}(\mathrm{NF}(\Lambda_\to))$ is called a concretisation of unknown quantities. *For any λ-term M the* result of the concretisation of its unknown quantities *is a term $\rho(M)$ in which every unknown quantity x is substituted for by the term $\rho(x)$.*

Please note that in the definition of the result of concretisation, no variable from concretisation gets bounded during the process of substitution.

Definition 8 `(the matching problem, a solution of an equation)`
The higher-order matching problem *is a decision problem to ascertain whether for a given higher-order matching equation $M = N$ exists a concretisation of unknown quantities $\rho : \mathcal{E} \to \mathrm{Closed}(\mathrm{NF}(\Lambda_\to))$ such that*

$$\mathrm{NF}(\rho(M)) = N.$$

Such a concretisation is called a solution of the equation *$M = N$.*

Throughout the rest of the text, we use the term *matching problem* to denote the *higher-order matching problem*

Definition 9 `(an order of a higher-order matching problem)`
The matching problem of the order *n is a higher-order matching problem where instances may have unknown quantities symbols of the order at most n.*

2.4 The linear interpolation problem

Now we introduce the problem that is equivalent to the matching problem as we show later.

Definition 10 (an interpolation equation)
We say the matching equation $M = N$ is an interpolation equation iff

$$M = xM_1 \ldots M_n$$

where $x \in \mathcal{E}$ and for each i term M_i is closed (in particular it has no occurrence of an unknown quantity).

Apparently, interpolation equations are much simpler than arbitrary matching equations. Each such an equation contains exactly one unknown quantity and this leads to the situation that the solution we are looking for consists of a single term and it is not a set of terms anymore. It is easy seen, though, that a single interpolation equation has always a solution and in the case of the equation in Definition 10 it looks like this

$$\rho(x) \;=\; \lambda x : \sigma_1 \ldots x : \sigma_n . N$$

where σ_i is the type of M_i.

This leads to the conclusion that *the linear interpolation problem* has to be formulated as a search for a solution of a set of interpolation equations.

Definition 11 (the linear interpolation problem)
The linear interpolation problem *is a problem to decide whether there exists for a finite set E of interpolation equations of the shape*

$$e \;=\; [xM_1 \ldots M_n \;=\; N],$$

where x appears in all equations and in each such an equation is the only unknown quantity, concretisation of unknown quantities

$$\rho : \{x\} \;\rightarrow \mathrm{Closed}(\mathrm{NF}(\Lambda_{\rightarrow}))$$

that is a solution for each $e \in E$.

We call such a concretisation a solution of the interpolation set E.

The maximum in E of the number of occurrences in terms N is denoted

$$\mathrm{MaxResult}(E).$$

As only one value is relevant in such a concretisation, we sometimes use the name *solution of an interpolation set* in order to refer the one value.

Definition 12 (an order of an interpolation problem)
The interpolation problem of the order n *is an interpolation problem where instances may have unknown quantities symbols of the order at most n.*

Please note that such formulation leads to a problem that need not to be trivial. The following example illustrates the claim. The set of interpolation equations $E = \{e_1,\ e_2\}$, where e_1 i e_2 are

$$e_1 = [xZ = Z_1]$$
$$e_2 = [xZ = Z_2]$$

and $Z,\ Z_1,\ Z_2$ are pairwise distinct constants of the same type, does not have any solution.

Definition 11 could be formulated in such a way that different interpolation equations from one instance might contain many unknown quantities. This is superfluous, though. Single equation may contain only one unknown quantity so solution for one unknown quantity does not affect other ones and such a problem may be divided into subproblems of the defined above form. Solutions of all the subproblems generate a solution of the primary instance.

In the next section, we show the relation between the just formulated problems.

3 The matching problem and the interpolation problem

We start with the simpler reduction. We show that interpolation problems may be solved using algorithm for the higher-order matching problem.

Fact 3.1 (interpolation $\leq_m$ matching)
Assume that there exists an algorithm $\mathcal{A}$ solving the higher-order matching problem. Then, there exists an algorithm $\mathcal{B}$ that solves the interpolation problem.

Proof:
Let $E = \{e_1, \ldots, e_m\}$ be an instance of the interpolation problem that for each i has $e_i = [x^i M_1^i \ldots M_{n_i}^i = N^i]$. If we have an algorithm $\mathcal{A}$ for the higher-order matching problem we can solve such an instance by introducing a new constant

$$Z : \tau_1 \to \ldots \to \tau_m \to \alpha,$$

where τ_i is a type of term N^i in the interpolation equation e_i, and α is a type constant, and then applying the algorithm $\mathcal{A}$ to the following instance of the matching problem

$$Z(x^1 M_1^1 \ldots M_{n_1}^1) \ldots (x^m M_1^m \ldots M_{n_m}^m) = ZN^1 \ldots N^m$$

It is straightforward that each solution of the instance gives a solution of the collection E and that each solution of E is a solution of the instance, too. $\Box$

It is worth mentioning that in this construction the order of the problem to be solved does not change.

Now we show the reverse reduction. To achieve the result, we show a simple fact that helps to reason about the shape of the solution when we know the result of normalisation.

Fact 3.2 `(the shape of a term and a result of normalisation)`
Let t be a term in the normal form and Z a constant of a base type. If

$$\mathrm{NF}(t\lambda y_1 \dots y_m.Z) = Z$$

then $t = \lambda y.Z$ or $t = \lambda y.yM_1 \dots M_m$ where M_i for $i \in \{1, \dots, m\}$ are in the normal form.

Proof:
The term t is a closed term in normal form so it is a constant or a λ-abstraction. The first case is impossible otherwise, assuming the constant is C, we get

$$\mathrm{NF}(t\lambda y_1 \dots y_m.Z) = C\lambda y_1 \dots y_m.Z \neq Z.$$

Let t be a λ-abstraction. This implies $t = \lambda y.\ vM_1 \dots M_n$ where v is a symbol of a suitable type, each M_i is a term of a suitable type in normal form, $n \geq 0$. If v is the constant Z we get the first shape from the thesis of our fact. If v is different from Z then v is equal to y or equal to some constant D. The first case corresponds to the second shape from the thesis of our fact. The second case cannot happen otherwise D would have to occur in the term $\mathrm{NF}(t\lambda y_1 \dots y_m.Z)$ and this is impossible by assumptions of our fact. $\square$

Fact 3.3 `(matching` $\leq_m$ `interpolation)`
Given an algorithm $\mathcal{A}$ that solves the interpolation problem, we can construct an algorithm $\mathcal{B}$ that solves the higher-order matching problem.

Proof:
Let $M = N$ be a matching problem instance. We present an interpolation problem instance which has a solution iff the instance $M = N$ has one.

The constructed set E of interpolation equations contains elements

$$\begin{aligned} x\lambda y_1 \dots y_m.M' &= N \\ x\lambda y_1 \dots y_m.Z &= Z \end{aligned} \tag{2}$$

where Z is a fresh constant of a suitable type and

$$M' = M[y_1/x_1, \dots, y_m/x_m].$$

Additionally, $\{x_1, \dots, x_m\} = FV(M) \cap \mathcal{E}$ and $\{y_1, \dots, y_m\}$ is a set of fresh local variables of suitable types.
($\Rightarrow$) Given a solution $\rho : \mathcal{E} \to \Lambda_\to$ of the equation $M = N$ such that

$$\rho(x_i) = t_i$$

the solution $\rho' : \{x\} \to \mathrm{Closed}(\mathrm{NF}(\Lambda_\to))$ of the constructed interpolation problem instance may be described by the pattern

$$\rho'(x) = \lambda y.\ yt_1 \dots t_m$$

The proof that ρ' is a solution of the collection (2) is straightforward but tedious.
($\Leftarrow$) Let ρ' be a solution of our instance of the interpolation problem and

$$\rho'(x) = t$$

Assume further, t is in normal form. The shape of the second equation implies that $\mathrm{NF}(t\lambda y_1 \ldots y_m.Z) = Z$. Now we can use Fact 3.2 and get $t = \lambda y.\ Z$ or $t = \lambda y.\ yu_1 \ldots u_m$. The first case is impossible, because the result of the first interpolation equation cannot be Z (Z is fresh).

Now it is easy to see that each $\rho : \mathcal{E} \to \Lambda_{\to}$ such that

$$\rho(x_i) \;=\; \mathrm{NF}((\lambda y.\ u_i)\lambda y_1 \ldots y_m.\ M')$$

is a solution of the instance $M = N$ of the matching problem. $\square$

Remark 3.4 *Please note that the constructed in the proof of Fact 3.3 instance of the interpolation problem has the order greater by two than the order of the primary matching problem instance.*

We end this part of the paper with the following theorem

Theorem 3.5 `(equivalence of problems)`
The problem of linear interpolation is recursively equivalent to the higher-order matching problem.

Proof:
It immediately follows from Fact 3.1 and Fact 3.3. $\square$

4 Decidability of a fragment of the fifth order interpolation problem

At the beginning, we introduce a definition that shall help us to present the fragment of the interpolation problem we are dealing with.

Definition 13 `(unsophisticated` λ`-terms)`
We say a λ-term $\lambda x_1 \ldots x_m.M$ is unsophisticated *iff for each occurrence in M of a term of the form*

$$x_i\ N_1 \ldots N_k$$

and for each $j \in \{1, \ldots, k\}$ none of x_l, where $l \in \{1, \ldots, m\}$, appears in N_j.

Each variable, constant, or application is an unsophisticated *term.*

Definition 14 `(the interpolation with unsophisticated arguments)`
The linear interpolation problem with unsophisticated arguments *is a linear interpolation problem, where we impose additional restriction on the form of an instance — for each $e \in E$ of the form*

$$e \;=\; [xM_1 \ldots M_n \;=\; N]$$

where M_i for $i \in \{1, \ldots, n\}$ is an unsophisticated term.

A solution of a set E of such equations is called a solution of the interpolation with unsophisticated arguments set E.

We restrict additionally our attention to the instances of the fifth order.

We show further that if for a given set E, there exists any solution then there exists one in some recursively dependent on E set of λ-terms. Fact 2.1 plays here the crucial role.

We fix a set E of equations and analyse the set of its solutions. Our construction consists of two steps. In the first one, we restrict the set of solutions in such a way that we know the set of types of their subterms and the set of constants they are built up of. In the second one, we narrow the already got class so that we know the depth of solutions.

Before we present next results, we introduce some notation. For an interpolation equations set

$$E = \{e_1, \ldots, e_n\}$$

where

$$e_i = [xM_1^i \ldots M_k^i = N^i]$$

we put

$$\mathrm{MaxResult}(E) = \max_{i \in \{1,\ldots,n\}} |N^i|$$

$$\mathrm{SumResult}(E) = \sum_{i=1}^{n} |N^i|.$$

4.1 Accessible terms

We define here a class of solutions the elements of what intuitively do not have unnecessary subtrees in their Böhm trees. The idea of the notions is taken form [3].

Definition 15 `(an occurrence accessible wrt. an equation)`
Let $e = [xM_1 \ldots M_n = N]$ is an interpolation equation, t its solution, and γ an occurrence in the term t. We say the occurrence γ in the term t is accessible wrt. equation e *iff the term*

$$\mathrm{NF}((t[Z \to \gamma])M_1 \ldots M_n),$$

where Z is a fresh constant of a suitable type, has an occurrence of the constant Z.

The set of all accessible wrt. equation e occurrences is denoted by

$$\mathrm{Accessible}(t, e).$$

Intuitively, an accessible subterm cannot be lost during the reduction of the term $tM_1 \ldots M_n$.

Definition 16 `(an occurrence accessible wrt. a set of equations)`
We say that an occurrence γ is accessible wrt. the set E of equations *iff it is accessible wrt. at least one of $e \in E$.*

If γ is not accessible we say it is inaccessible.

We denote the set of all occurrences accessible wrt. the set E of equations by $\mathrm{Accessible}(t, E)$.

Definition 17 `(an accessible path)`
A path γ is accessible *iff the occurrence that corresponds to its end is accessible. If such a path is not accessible we call it* inaccessible.

Definition 18 `(an accessible solution)`
The solution t of a set E is accessible *iff each occurrence γ in the solution either is accessible or is an occurrence of a constant Z of a suitable type.*

We fix a simple types indexed set $\mathcal{P}$ of fresh constants. Additionally, we assume each $\mathcal{P}_\sigma$ has exactly one element. On next pages, we are considering the lambda calculus extended by the set. The set we call *the set of fillings* and each its element *a filling*.

Definition 19 `(a solution with a good filling)`
We say the solution t of the set E of equations has a good filling *iff each inaccessible occurrence in t is an occurrence of a filling.*

Remark 4.1 *Obviously, solutions with a good filling are accessible.*

Theorem 4.2 `(existence of solutions with a good filling)`
If there exists a solution of a set E of interpolation equations then there exists a solution of the set which has a good filling.

Proof:
Let E be a set of interpolation equations and t its solution. We construct the term $\hat{t}$ that has a good filling.

Let γ be an inaccessible wrt. the set E occurrence in t. We graft the filling Z of a suitable type at γ. The resulting term t' is a solution, because for each equation

$$e \;=\; [xM_1 \dots M_n \;=\; N]$$

in E we have correspondence between the reduction of the term

$$(t[Z \to \gamma])M_1 \dots M_n$$

and the term

$$tM_1 \dots M_n.$$

The corresponding terms in the reductions are the same but the residuals of respectively Z and the term at γ in t. When a residuum of Z disappears then disappears a residuum of the term at γ in t. At last, we get the term

$$\mathrm{NF}((t[Z \to \gamma])M_1 \dots M_n)$$

which does not contain Z (by accessibility). Obviously, the corresponding term in the other reduction does not contain any residuum of the term at γ in t. This proves the terms are equal so $\mathrm{NF}((t[Z \to \gamma])M_1 \dots M_n) = N$.

Using this argumentation, we eliminate one by one inaccessible occurrences of non-trivial terms. At last, we get a solution that has a good filling. $\square$

Fact 4.3 `(number of constants in the solution with a good filling)`
The set of constants occurring in the solution with a good filling is contained in the set

$$C = \bigcup_{i=1}^{n} \mathrm{Const}(N^i) \cup \mathcal{P}.$$

Proof:
We have two possibilities

1. an occurrence γ of a constant D is accessible wrt. some equation e_i

2. an occurrence γ of D is inaccessible.

In the first case, we show by the induction on the sum of the length of all the reductions from the term $t[Z \to \gamma]M_1^i \dots M_k^i$ to its normal form that occurrences of the constant Z correspond to suitable occurrences of the symbol D. As γ is an accessible path, $\mathrm{NF}(t[Z \to \gamma]M_1^i \dots M_k^i)$ contains an occurrence of Z, so $\mathrm{NF}(tM_1^i \dots M_k^i)$ contains D and then D is a constant in N^i.

In the second case, we get by the definition of an accessible solution that $D \in \mathcal{P}$. □

Fact 4.4 `(types of symbols in an accessible solution)`
For each interpolation set E, there exists a closed on subtypes, finite and recursively dependent on E set of types $\mathcal{T}$ such that for each accessible solution t of E the set of types of symbols occurring in t is included in $\mathcal{T}$.

Proof:
The set of types is

$$\mathcal{T} = \bigcup_{i=1}^{n} \mathrm{SubTypes}(\mathrm{Types}(N^i)) \cup \mathrm{SubTypes}(\{\sigma\})$$

This set is finite, recursively dependent on E and closed on subtypes. We shall show that the type of each symbol v occurring in t is in $\mathcal{T}$.

Induction wrt. depth of occurrence of v.

- When the depth is zero the symbol v cannot occur.

- When the depth is greater than zero we have two cases

  1. The symbol v is a variable. If v is from the main abstraction then its type is in $\mathrm{SubTypes}(\sigma)$ else v is declared in t inside one of terms D_i from the expression

     $$wD_1 \dots D_m$$

     and by inductive assumption the type of w is in T. The type of v is a subtype of D_i and the type of D_i is a subtype of w so as $\mathcal{T}$ is closed on subtypes the type of v is in $\mathcal{T}$.

2. The symbol v is a constant. If v is an accessible constant then it must occur in one of N^i so the type of v is in $\mathrm{Types}(N^i)$ else by the definition of the accessible solution it must occur in the expression like
$$wD_1 \dots v \dots D_m$$
so the type of v is a subtype of w. By the inductive assumption, the type of w is in $\mathcal{T}$.

□

Remark 4.5 *It is worth mentioning that the above reasoning concerns arbitrary interpolation problem.*

In the following sections, we achieve the third needed property — boundedness of the length of a path in a solution. We make the assumption that if we have a solution in hand then the solution is accessible.

4.2 Active and passive symbols

Here we introduce the notion of active and passive symbols and show a bound on the number of occurrences of the latter ones. The next section is concerned with active symbols.

Definition 20 `(active and passive symbols)`
Let $t = \lambda\vec{x}.t'$ be a λ-term. We define the set Active(t) *of the* active occurrences

- *each occurrence of x_i from $\vec{x}$ is in* Active(t)
- *for each subterm*
$$zt_1 \dots t_k$$
such that z is in Active(t), *if $t_i = \lambda\vec{y}.\ t_i'$ then each occurrence of y_j from $\vec{y}$ in t_i' belongs to* Active(t).

We say an occurrence γ in a term t is passive, *when it does not belong to* Active(t). *The set of all passive occurrences is denoted by* Passive(t).

If an occurrence of a symbol z is in Active(t) *then we say the symbol is* active *else we say the symbol is* passive.

Fact 4.6 `(passive variables and substitutions)`
For each interpolation equations set E and its accessible solution t, if an occurrence $\gamma \in$ Passive(t) *is an occurrence of a symbol v then for any equation*
$$e = [xM_1 \dots M_k = N] \in E$$
during normalisation of
$$tM_1 \dots M_k$$
v is not substituted for.

Proof:
We show our claim by the induction on the depth of an occurrence of a passive variable on the path γ.

- For the depth 0, there are no occurrences so the claim is true.
- For occurrences of the depth $n > 0$, we assume by the induction hypothesis our claim for all $m < n$. Let us take the passive symbol v occurring at the depth n. There are three cases
  - Symbol v is a constant — the thesis holds by the constant case of Definition 20.
  - Symbol v occurs earlier on the path — the thesis holds by applying the induction hypothesis to the earlier occurrence and by the fact that all the occurrences of the same variable in a given context are substituted for by the same term.
  - Symbol v does not occur earlier on the path and is a variable — the proof is more sophisticated.

As v is not earlier on the path and is not a constant, it must have been declared in a subterm of the form

$$zt_1 \dots (\lambda \vec{u} v \vec{w}.t'_i) \dots t_k$$

where z is passive. By the induction hypothesis, z is not substituted for during the reduction. This implies v is not substituted for, either. This completes the proof. □

Fact 4.7 `(passive variables and occurrences in results)`
For each interpolation equations set E, its accessible solution t, and equation $e \in E$ of the form

$$e \;=\; [xM_1 \dots M_k \;=\; N],$$

there exists one-to-one map

$$f_e : \mathrm{Passive}(t) \cap \mathrm{Accessible}(t, e) \to \mathrm{Occurences}(N).$$

Proof:
Let $\gamma \in \mathrm{Passive}(t) \cap \mathrm{Accessible}(t, e)$ be a path. As $\gamma \in \mathrm{Passive}(t)$ during normalisation of

$$tM_1 \dots M_k,$$

the occurring at γ symbol v is not substituted for. As $v \in \mathrm{Accessible}(t, e)$, there exists nonempty set A_γ of occurrences in $\mathrm{NF}(tM_1 \dots M_k)$ that appeared due to γ. Let us put $f_e(\gamma)$ so that

$$f_e(\gamma) \in A_\gamma.$$

Each such a map is one-to-one, since for passive occurrences $\gamma_1 \neq \gamma_2$ neither symbol at γ_1 nor the symbol at γ_2 is substituted for and so sets A_{γ_1} and A_{γ_2} are disjoint (the reduction does not cause unifying of variables that are not substituted for). □

Corollary 4.8 `(a bound on the number of passive symbols)`
For any interpolation equation set $E = \{e_1, \ldots, e_n\}$ *where*

$$e_i \;=\; [xM_1^i \ldots M_k^i \;=\; N^i],$$

with accessible solution t*, we have*

$$|\mathrm{Passive}(t) \cap \mathrm{Accessible}(t, E)| \;\leq \mathrm{SumResult}(E).$$

Proof:
We have

$$\begin{aligned}|\mathrm{Passive}(t) \cap \mathrm{Accessible}(t, E)| &= |\bigcup_{i=1}^{n} \mathrm{Passive}(t) \cap \mathrm{Accessible}(t, e_i)| \;\leq \\ &\leq \sum_{i=1}^{n} |\mathrm{Passive}(t) \cap \mathrm{Accessible}(t, e_i)|\end{aligned}$$

By Fact 4.7, we get

$$\begin{aligned}\sum_{i=1}^{n} |\mathrm{Passive}(t) \cap \mathrm{Accessible}(t, e_i)| &\leq \sum_{i=1}^{n} |\vec{f_{e_i}}(\mathrm{Passive}(t) \cap \mathrm{Accessible}(t, e_i))| \;\leq \\ &\leq \sum_{i=1}^{n} |N^i| \;\leq \mathrm{SumResult}(E)\end{aligned}$$

This completes the proof. □

Now we show a bound on the number of occurrences of active variables. This is done in two steps. The first one is for the so called variables from the main abstraction, the second one for the variables from the side abstractions.

4.3 Variables from the main abstraction

Now we show that if there exists a solution of a set E of equations with unsophisticated arguments then there exists a solution $\hat{t}$ such that paths in $\hat{t}$ cannot contain arbitrary many occurrences of variables from the main abstraction.

We begin with the definition of *variables from the main abstraction.*

Definition 21 `(variables from the main abstraction)`
Let $t = \lambda\vec{y}.t'$ *be a solution of a set* E*. Variables from* $\vec{y}$ *are called* variables from the main abstraction*. The rest of active variables are called* variables from side abstractions.

We prove a fact that will help us to understand what happens for deep paths.

Fact 4.9 `(the form of arguments when many occurrences)`
Let $E = \{e_1, \ldots, e_n\}$ *be a set of equations with unsophisticated arguments and* t *its solution. Additionally, let for each* i

$$e_i \;=\; [xM_1^i \ldots M_k^i \;=\; N^i].$$

Then for each i *there exists* p *such that if for a certain path* γ *in* t*, which is accessible wrt. one of* e_j*, the variable* y_i *from the main abstraction has on* γ *more than* $\mathrm{MaxResult}(E)$ *occurrences then*

- *the term M_i^j has the form*

$$\lambda\vec{z}.z_p C_1 \ldots C_r$$

where none of z_i in $\vec{z}$ occurs in C_l for $l \in \{1, \ldots, r\}$,

- *for each occurrence δ of the variable y_i, where the term at δ has the form*

$$y_i D_1 \ldots D_m,$$

the next occurrence on γ is an occurrence of D_p.

Proof:
The term M_i^j has the form

$$\lambda\vec{z}.vC_1 \ldots C_r$$

where v is a constant or a local variable. The only local variables that can occur here (as M_i^j is closed) are variables from $\vec{z}$ so the only bad case is the case when v is a constant. However, v cannot be a constant. Indeed, if v is a constant then observation of the normalisation of $tM_1^j \ldots M_k^j$ leads to the conclusion that $\mathrm{NF}(tM_1^j \ldots M_k^j)$ $(= N^j)$ has more than $\mathrm{MaxResult}(E)$ occurrences (which is impossible). The last claim is true since the term M_i^j is substituted on y_i and none of accessible occurrences of y_i (there are more than $\mathrm{MaxResult}(E)$ such occurrences) can disappear. □

Definition 22 (Direction)
For a given E and t in the situation from the previous fact, the number p from the previous fact is denoted by

$$\mathrm{Dir}(y_i).$$

We introduce the notion of *a compact wrt. variables from the main abstraction solution*. For such solutions we can give a bound on the number of occurrences of variables from the main abstraction.

Definition 23 (a compact solution - the main abstraction)
Let E be a set of interpolation equations. We say that an accessible solution t is compact wrt. variables from the main abstraction *iff for each variable y the variable can occur at most* $\mathrm{MaxResult}(E)$ *times on any path in t.*

Theorem 4.10 (existence of compact solutions 1)
If the set E of interpolation equations with unsophisticated arguments has an accessible solution t then there exists a compact wrt. variables from the main abstraction solution $\hat{t}$.

Moreover, we can assume the set of symbols occurring in $\hat{t}$ is contained in the set of symbols occurring in t.

Proof:
Given t, we construct $\hat{t}$. Induction on the sum of lengths of paths γ such that there exists a variable y from the main abstraction with more than $\mathrm{MaxResult}(E)$ occurrences on γ.

- If the sum is equal to zero then we put $\hat{t} = t$. Obviously, such $\hat{t}$ meets our requirements.
- If the sum is greater than zero then we construct an accessible solution t' which is a copy of t except one path that has cut off one occurrence of some variable from the main abstraction. This implies the sum of lengths of described above γ's is lower for t' so the induction hypothesis applied to the new solution gives desired $\hat{t}$.

Now we construct t'. First we introduce some notation. Let

$$E = \{e_1, \ldots, e_n\}$$

and for each i

$$e_i = [xM_1^i \ldots M_k^i = N^i].$$

Let y be a variable from the main abstraction, γ be a path that is

$$\mathrm{MaxResult}(E) + 1\mathrm{st}$$

occurrence of y and let δ be a MaxResult(E)th occurrence of y on δ. The term at γ has the form

$$yD_1 \ldots D_m$$

and the term at δ

$$yD'_1 \ldots D'_m.$$

Moreover, $D'_{\mathrm{Dir}(y)} = \lambda z_1 \ldots z_r.\ D''$. At last, we can put

$$t' = t[\mathrm{NF}(D_{\mathrm{Dir}(y)} z_1 \ldots z_r) \to \gamma]$$

The term t' is a copy of t except for one path that has cut off one occurrence of the variable y from the main abstraction. We have to prove t' is an accessible solution. In order to do this it is enough to prove that for each $i \in \{1, \ldots, n\}$

$$\mathrm{NF}(\hat{t}M_1^i \ldots M_k^i) = N^i$$

and that each occurrence in the term at γ in t' (the unlabelled counterpart of γ still exists in t', we take into account the counterpart) is accessible wrt. some equation or is an occurrence of a constant.

If γ is inaccessible wrt. $e_i \in E$ in t' then during the normalisation of

$$t'M_1^i \ldots M_k^i$$

all terms that come from the occurrence of γ disappear and

$$\mathrm{NF}(t'M_1^i \ldots M_k^i) = \mathrm{NF}(tM_1^i \ldots M_k^i) = N^i$$

If γ is accessible wrt. $e_i \in E$ then by Fact 4.9 reductions in the term at γ may look as follows

$$yD_1 \ldots D_k \to_\beta M_j^i D_1 \ldots D_k \to_\beta^* D_{\mathrm{Dir}(y)} C_1 \ldots C_r \to_\beta^* D^\bullet_{\mathrm{Dir}(y)}[C_1/v_1 \ldots C_r/v_r] \tag{3}$$

where

$$M^i_j = \lambda\vec{z}.z_{\mathrm{Dir}(y)}C_1\ldots C_r$$

and

$$D_{\mathrm{Dir}(y)} = \lambda v_1\ldots v_r.D^\bullet_{\mathrm{Dir}(y)}$$

As M^i_j is unsophisticated, terms $C_1,\ldots C_r$ are substituted for corresponding variables in $D'_{\mathrm{Dir}(y)}$ (see the term at the occurrence δ). So the given by the rules

1. substitute arguments $M^i_1,\ldots,M^i_k$
2. do reduction (3)
3. do for δ a reduction analogous to (3)

reduction strategy for $tM^i_1\ldots M^i_k$ leads to the same result as the given by the rules

1. substitute arguments $M^i_1,\ldots,M^i_k$
2. do analogous to (3) reduction for δ

strategy for $t'M^i_1\ldots M^i_k$. This is enough to get

$$\mathrm{NF}(t'M^i_1\ldots M^i_k) = N^i$$

Now we prove accessibility. Let δ be an inaccessible occurrence in the term at γ in t'. The occurrence has its counterpart δ' in the term at γ in t. The occurrence δ' in t cannot be accessible, because as we have realized, the two mentioned above strategies of reduction lead to the same result. The inaccessibility of δ' implies that the term at the occurrence is a constant and this implies further that the term at δ in t' must be the same constant. This ends the proof of accessibility of t'.

□

Remark 4.11 *The proof for variables from the main abstraction involves only the assumption about unsophisticated arguments.*

Now we modify our solution so that we can estimate the number of variables from side abstractions.

4.4 Variables from side abstractions

We show the way of reconstructing a compact wrt. variables from the main abstraction solution so that the number of variables from side abstractions is dependent on the number of occurrences of variables from the main abstraction.

Definition 24 `(a compact solution - side abstractions)`
We say that a compact wrt. variables from the main abstraction solution t is compact wrt. variables from side abstractions *iff on each accessible path γ of the solution and for each variable y from side abstraction declared in the term*

$$\lambda\vec{x}.\lambda y.\lambda\vec{z}.D$$

variable y occurs on γ in D at most $\mathrm{MaxResult}(E)$ *times.*

We can prove the constructibility of such solutions now.

Theorem 4.12 `(existence of compact solutions 2)`
If the set E of fifth order interpolation equations has a compact wrt. variables from the main abstraction solution t then there exists a compact wrt. variables from side abstractions solution $\hat{t}$.

Moreover, we can assume the set of symbols occurring in $\hat{t}$ is contained in the set of symbols contained in t.

Proof:
Induction on the number of occurrences of variables y from the side abstractions that are declared on a certain path γ in a subterm of the form

$$\lambda\vec{x}.\lambda y.\lambda\vec{z}.D \tag{4}$$

such that y on γ and in D occurs more than $\mathrm{MaxResult}(E)$ times.

- If there are no such occurrences then the term t is compact wrt. variables from the side abstraction.
- If there exists such y then let us take $\mathrm{MaxResult}(E)+1$st occurrence δ of y in D and on γ. The term at δ has the form

  $$yD_1 \dots D_m$$

  and at most one of D_i for $i \in \{1,\dots,m\}$ is accessible. The last claim follows from the fact that, as there are more than $\mathrm{MaxResult}(E)$ occurrences of y, the only term that may be substituted for the variable in the reduction of the left hand side of the equation wrt. which γ is accessible is a second order term that reduces to

  $$\lambda z_1 \dots z_m.z_i$$

  otherwise the normal form would have more than $\mathrm{MaxResult}(E)$ occurrences.

  Of course, one of D_i is accessible (otherwise δ is inaccessible). Indeed, if none of D_i is accessible then the term substituted for y would ignore its arguments so it would ignore δ by the first occurrence of y on γ, too. Now we can put

  $$t' = t[D_k \to \delta]$$

  where D_k is the only accessible term.

  One can see t' is a term that

- has less occurrences of variables from side abstractions than t,
- the set of symbols occurring in it is contained in the set of symbols occurring in t,
- if t' is a solution then it is compact wrt. variables from the main abstraction.

We can draw the conclusion that if t' is a solution then we get our fact after applying the induction hypothesis to t'.

Now we show that t' is a solution. If δ is not accessible wrt. $e_i \in E$ then δ does not affect the normal form and t' is a solution of such an equation, too. If δ is accessible wrt. $e_i \in E$ then we reduce left-hand side according to the strategy consisting in following exactly one residuum of a term D and holding all the reductions inside the term D. By accessibility of D_k, there exists among terminal terms of such strategies at least one term t'' for which, after replacing one of copies of D_k by a suitable constant and further normalising, we get occurrences of the constant in the resulting normal form.

In t'', we have a term substituted for y. The normal form of the term may have one of the two shapes

$$\lambda z_1 \dots z_m . v P_1 \dots P_r \tag{5}$$

where v is a symbol that is not substituted for (if something was substituted then t'' would not be a terminal term of one of our strategies), or

$$\lambda z_1 \dots z_m . z_l . \tag{6}$$

The case (5) cannot happen, because then on the path δ, it would occur more then $\mathrm{MaxResult}(E)$ occurrences of v and the occurrences could not disappear during the reduction due to accessibility of δ. So the only possibility is (6). Moreover, accessibility of D_k implies that

$$\lambda z_1 \dots z_m . z_k .$$

Because this reasoning applies always to the situation when we use D_k, we get the same result by replacing the term at δ by D_k. This completes the proof.

□

Remark 4.13 *The assumption that the equations are of the fifth order was used only in the proof of Theorem 4.12.*

To formulate and prove next facts, we have to introduce some notation. We let $\mathrm{Active}_{\mathrm{main}}(t, \gamma)$ denote the set of occurrences of variables from the main abstraction on the path γ and let $\mathrm{Active}_{\mathrm{side}}(t, \gamma)$ denote the set of occurrences of variables from side abstractions on the path γ.

Let γ be a path and δ an occurrence on it. Moreover, let δ be an occurrence of a variable x from the main abstraction. Let the term grafted on δ has the form

$$xt_1 \dots t_k$$

and let the next occurrence on γ be an occurrence of a term t_i of the form

$$t_i \;=\; \lambda\vec{y}.t_i'.$$

In such a situation we let $\mathrm{Decl}(t, \gamma, \delta)$ denote the set of occurrences of variables that are declared in $\vec{y}$ and let $\mathrm{Decl}(t, \gamma, \delta, y_l)$, where y_l is one of variables from $\vec{y}$, denote the set of occurrences of the variable y_l in t_i'.

A last, we let $\mathrm{VarDec}(t, \delta)$ denote the set of variables contained in the list l, where l is the first component of the last label $(l,\ v)$ on δ. Intuitively, it is the set of variables declared at the beginning of the subterm of t that occurs at the end of δ.

The maximal number of arguments in types of subterms of the term t is denoted by

$$\mathrm{MaxSubterm}(t)$$

Corollary 4.14 (side abstractions and the main abstraction)
Let E be an arbitrary set of interpolation equations and t its solution that is compact wrt. variables from side abstractions. If the number of occurrences on an accessible path γ of variables from the main abstraction is bounded by K then the number of occurrences of variables from side abstractions on the path is bounded by

$$K\ \mathrm{MaxSubterm}(t)\mathrm{MaxResult}(E).$$

Proof:
We estimate the number of occurrences of variables from side abstractions

$$\begin{aligned}
|\mathrm{Active}_{\mathrm{side}}(t,\gamma)| &= |\bigcup_{\delta\in\mathrm{Active}_{\mathrm{main}}(t,\gamma)} \mathrm{Decl}(t,\gamma,\delta)| \leq \\
&\leq \sum_{\delta\in\mathrm{Active}_{\mathrm{main}}(t,\gamma)} |\mathrm{Decl}(t,\gamma,\delta)| = \\
&= \sum_{\delta\in\mathrm{Active}_{\mathrm{main}}(t,\gamma)} |\bigcup_{y\in\mathrm{VarDec}(t,\delta)} \mathrm{Decl}(t,\gamma,\delta,y)| \leq \\
&\leq \sum_{\delta\in\mathrm{Active}_{\mathrm{main}}(t,\gamma)} \sum_{y\in\mathrm{VarDec}(t,\delta)} |\mathrm{Decl}(t,\gamma,\delta,y)|
\end{aligned}$$

According to the definition of a compact wrt. variables from side abstractions solution, for each variable from side abstractions the number of its occurrences on an accessible path is not greater than $|N^j|$, where j is the number of the equation wrt. which γ is accessible. This implies estimation

$$\sum_{\delta\in\mathrm{Active}_{\mathrm{main}}(t,\gamma)} \sum_{y\in\mathrm{VarDec}(t,\delta)} |\mathrm{Decl}(t,\gamma,\delta,y)| \leq \sum_{\delta\in\mathrm{Active}_{\mathrm{main}}(t,\gamma)} \sum_{y\in\mathrm{VarDec}(t,\delta)} |N^j|$$

Further, we get easily

$$\sum_{\delta\in\mathrm{Active}_{\mathrm{main}}(t,\gamma)}\ \sum_{y\in\mathrm{VarDec}(t,\delta)} |N^j| \leq \sum_{\delta\in\mathrm{Active}_{\mathrm{main}}(t,\gamma)}\ \sum_{y\in\mathrm{VarDec}(t,\delta)} \mathrm{MaxResult}(E) \leq$$
$$\leq K\ \mathrm{MaxSubterm}(t)\mathrm{MaxResult}(E)$$

and this is our result. □

Now we are ready to draw final conclusions of the section.

Corollary 4.15 (the solution one can find)
Let E be a set of interpolation equations with unsophisticated arguments. There exist recursively dependent on E

- *a simple types indexed set of constants C such that for each type the set of constants of the type is finite,*
- *a finite set of types $\mathcal{T}$,*
- *a number g*

such that if there exists a solution of E then there exists a solution of E such that all the types of subterms belong to $\mathcal{T}$, all the constants are from C, and the length of paths is less or equal g.

Proof:
Let $E = \{e_1, \ldots, e_n\}$ be a set of interpolation equations with unsophisticated arguments of the form

$$e_i = [xM_1^i \ldots M_k^i = N^i]$$

and let t be a solution of E. We construct the mentioned in the fact objects

- C is the sum $C = \bigcup_{i=1}^{n} \mathrm{Const}(N^i) \cup \mathcal{P}$,
- $\mathcal{T}$ is the set from the Fact 4.4,
- g is given by the pattern

$$g = k\ \mathrm{MaxResult}(E) + k\ (\mathrm{MaxResult}(E))^2\mathrm{MaxSubterm}(t) + $$
$$+ \mathrm{SumResult}(E) + 1$$

(In the last pattern, $\mathrm{MaxSubterm}(t)$ is by Fact 4.4 recursively dependent on E.)

Now we show a solution $\hat{t}$ that meets our requirements. Fact 4.2 gives a solution t' which has a good filling, for which the set of types of symbols occurring in it is contained in $\mathcal{T}$, and for which the set of constants occurring in it is contained in C. The solution t' is accessible so by Theorem 4.10 and Theorem 4.12 there exists a solution t'' which is compact wrt. variables from side abstractions and for which the set of symbols occurring in it is contained in the set of symbols occurring in t'. The last property ensures that C and $\mathcal{T}$ retain their properties for t''. By the compactness of t'', we know that the number of variables from the main abstraction is bounded by

$$k \ \mathrm{MaxResult}(E).$$

Furthermore, by Corollary 4.14 we get that the number of occurrences of variables from side abstractions is bounded by

$$k \ \mathrm{MaxResult}(E)\mathrm{MaxSubterm}(t)\mathrm{MaxResult}(E).$$

The number of passive symbols, by Corollary 4.8, is bounded by

$$\mathrm{SumResult}(E).$$

A filling is the last kind of symbol that may occur on a path in t'' so we put 1. This implies that defined g is a bound on the depth of a path in t''. □

4.5 Decidability

Now we are ready to draw the conclusion of the decidability.

Theorem 4.16 `(the decidability of the interpolation problem)`
The fifth order interpolation problem with unsophisticated arguments is decidable.

Proof:
For a given interpolation equations set E the algorithm proceeds as follows

1. generate by means of the algorithm in Fact 4.15 a simple types indexed set C of constants, a finite set of types $\mathcal{T}$, and a number g,
2. generate on the basis of $C, \mathcal{T},\ g$, and the type σ of unknown quantity in E by means of the algorithm from Fact 2.1 the set A of all terms of the type σ with constants from C, types of subterms from $\mathcal{T}$, and with depth not greater than g,
3. check if there exists any element in A that is a solution of E; if so — we have found a solution, if not — the set has no solution.

The algorithm stops, because the procedures used in steps 1 and 2 stop and because the set A is finite according to Fact 2.1.

The correctness of the algorithm is obvious as when we end successfully we have a solution in hand.

When the algorithm stops saying that the solution does not exist it means that there is no solution in A. Existence of a solution in such a situation would contradict the Corollary 4.15. This completes the proof. □

Remark 4.17 *The described above algorithm constructs a solution that potentially contains constants that were absent from the original problem. The constants are fillings. This does not affect the result, because the fillings are used to replace other terms so each used filling has at least one corresponding term. When necessary we can replace a filling by such a term.*

References

[1] H.P. Barendregt. *The Lambda Calculus, its Syntax and Semantics*. North Holland, 2 edition, 1984.

[2] H.P. Barendregt. Lambda calculi with types. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 117–309. Oxford Science Publications, 1992.

[3] G. Dowek. Third order matching is decidable. *Annals of Pure and Applied Logic*, 1993.

[4] B. Lang G. Huet. Proving and applying program transformations expressed with second order patterns. *Acta Informatica*, (11):31–55, 1978.

[5] G. Huet. *Résolution d'Équations dans les Languages d'Ordre* $1, 2, \ldots, \omega$. PhD thesis, Université Paris VII, 1976.

[6] Vincent Padovani. *Filtrage d'Ordre superieur*. PhD thesis, Université Paris VII, January 1996.

[7] D.A. Wolfram. *The Clausal Theory of Types*. PhD thesis, University of Cambridge, 1989.