Math. Struct. in Comp. Science (2000), *vol.* 10, *pp.* 361–372. Printed in the United Kingdom

Decidability of fourth-order matching

VINCENT PADOVANI

Université PARIS VII-C.N.R.S U.R.A. 753,
Equipe Preuves, Programmes et Systèmes,
2 Place Jussieu 75251 PARIS CEDEX 05 FRANCE
Email: padovani@pps.jussieu.fr

Received 9 November 1997; revised 15 November 1999

Higher Order Matching consists of solving finite sets of equations of the form $u =_{\beta\eta} v$, where u, v are simply typed terms of λ-calculus, and v is a closed term. Whether this problem is decidable is an open question. In this paper its decidability is proved when the free variables of all equations are of order at most 4 – we actually extend this result to a more general situation in which in addition to equations, disequations are also considered.

1. Introduction

Higher Order Matching is a special case of two well-known undecidable problems, *Higher Order Unification* (Huet 1976; Goldfarb 1989), and *Lambda-Definability* (Plotkin 1993; Loader 1995). The decidability of this problem remains open at the time of publication.

Higher Order Matching consists of solving finite sets of equations of the form $u =_{\beta\eta} v$, where u, v are simply typed terms of λ-calculus, and v is a closed term. By definition, the order of an instance is the maximal order of its free variables. In 1976, Huet proved the decidability of this problem when the free variables of the given equations are of order at most two. The third-order case was solved by Dowek in 1992. In 1995, we proved the decidability at all orders of the case where every right-member of an equation is a constant of ground type, as a corollary of the decidability of all minimal models of simply typed lambda-calculus – a result already conjectured by Statman and Plotkin around 1980. Schmidt-Schauß recently gave an alternate proof for this special case, providing a much simpler algorithm for the representation of the components of a minimal model. This decidability result was also extended by Loader to PCF_1 (Loader 1995).

In this paper we prove the decidability of fourth-order matching. Comon and Jurski gave an alternate proof of this decidability result: they showed how to compute a tree automaton accepting all solutions of a fourth-order problem (Comon and Jurski 1997). Schubert gave a partial decidability result for the fifth-order case, and proved that the decidability of higher order matching at order N is a consequence of the decidability of a simpler problem, the Interpolation Problem, at order $N + 2$ (Schubert 1997). Wolfram gave an algorithm for higher order matching whose soundness has been proved, but whose completeness is still conjectured (Wolfram 1992). Dowek proved the matching problem's undecidability in all type systems in which the primitive recursive functions are representable, for example Gödel's system T or polymorphic λ-calculi (Dowek 1993). In the latter system, Springintveld proved that the matching problem is decidable up to order three (Springintveld 1995).

This paper presents a decision algorithm for the fourth-order case of Higher Order Matching. Actually, our algorithm applies to the fourth-order case of *Dual Matching*, where in addition to equations, matching disequations (for example, $u \neq_\beta v$) are also considered. The technique is quite similar to the one we used in the atomic case. First, we define a collection of equivalences between terms. Each equivalence satisfies the property that there are only finitely many equivalence classes of any given type. This collection is in fact a variant of the one defined by Statman (Statman 1982; Statman 1992), and the resulting finiteness property – for which we give a completely syntactic proof – happens to follow from Statman's Finite Completeness theorem. Each set of equivalence classes is called a *limited model* of simply typed lambda calculus. A limited model is said to be decidable if, given an arbitrary type, one is able to compute a finite list of terms containing one representative of each equivalence class of this type. The decidability of all models built from this collection is shown to be equivalent to the decidability of Dual Matching. A context lemma – required in the finiteness proof – implies that in order to decide these models, we only need to decide a special case of Dual Matching, the *Dual Interpolation* problem, in which all equations (respectively, all disequations) are of the form $[x\,u_1 \ldots u_n = v]$ (respectively, $[x\,u_1 \ldots u_n \neq v]$), the terms $u_1, \ldots, u_n$ and v being closed, v being of ground type. Finally, Dual Interpolation is shown to be decidable up to order 4, all models being, accordingly, decidable up to this order, and consequently Dual Matching and Pattern Matching being decidable in the fourth-order case.

2. Simply typed terms

We will use a standard definition of simply typed lambda calculus with constants, with typing *à la* Church. This calculus is well known to be strongly normalizing (Barendregt 1984; Hindley and Seldin 1986; Krivine 1993). The definitions of β and η-reductions, of α-equivalence, of a correct substitution and of the position of a subterm in a term, are omitted. Simple types are generated using the following grammar, where ι ranges over a finite set of types variables†.

$$T ::= \iota \,|\, T \to T$$

Note that a type A always has the shape $(A_1 \to (\ldots A_n \to \iota)\ldots)$ where ι is a type variable. As usual, this latter form will be denoted $A \to \ldots \to A_n \to \iota$. The functional complexity

† In order to solve a specific instance of the (Dual) Matching problem, we do not need any other ground types than the ones appearing in this instance, hence we can assume without loss of generality that there are only finitely many ground types. Although this fact seems intuitively obvious, the proof is not trivial (Padovani 1996).

of a type is measured by an integer called its *order*[†]. Type variables are of order 1. The order of the type $A_1 \to \ldots \to A_n \to \iota$ $(n > 0)$ is equal to $\sup_{i=1}^{n}(\mathrm{Ord}(A_i)) + 1$.

Terms are built from a countable set of variables $x, y, z, \ldots$, and a countable set of constants $a, b, c, \ldots$ Each variable and each constant is assumed to be assigned a unique type. The set of *simply typed terms* is the least set $\mathscr{S}$ satisfying

1 $x : A \in \mathscr{S} \Leftrightarrow x$ has the type A (for all variable x),
2 $c : A \in \mathscr{S} \Leftrightarrow c$ has the type A (for all constant c),
3 $t : B \in \mathscr{S}, x : A \in \mathscr{S} \Rightarrow \lambda x\, t : A \to B \in \mathscr{S}$,
4 if $t : A \to B$ and $u : A \in \mathscr{S}$, then $(t\, u) : B \in \mathscr{S}$.

A term of the shape $(\ldots(u_0\, u_1)\ldots u_p)$ $(p > 0)$ will be abbreviated by $(u_0\, u_1 \ldots u_p)$, or even $u_0\, u_1 \ldots u_p$ if this does not cause ambiguity. By convention, $(u_0\, u_1 \ldots u_p)$ denotes the term u_0 if $p = 0$. The *order* of a typed term is the order of its type. A *closed* term contains no free variable, for example, the typed constant $a : \iota$ is a closed term.

It will be more convenient for our purposes to assume that α-equivalent terms are not identified: α-conversions, whenever required, will always be made explicit. We assume that every term t has a canonical normal form, which we will call *the* normal form of t.

In Dowek's proof (Dowek 1993), all terms in normal form are supposed to be in *η-long form*. This hypothesis is not essential, but we believe that it lightens the proofs, since a solution of a problem beginning with n λ's is then always applied to exactly n arguments – and, furthermore, it is sufficient then to consider β-equalities instead of $\beta\eta$-equalities. The notion of η-long normal form can be easily extended to all terms with the following definition: let $t = \lambda y_1 \ldots \lambda y_m\, (u_0\, u_1 \ldots u_p) : A_1 \to \ldots \to A_n \to \iota$ $(m \leqslant n)$ where u_0 is either a variable, a constant, or a term beginning with a λ; an *η-long form* of t is a term t' of the form

$$t' = \lambda y_1 \ldots \lambda y_m \lambda y_{m+1} \ldots y_n\, (u'_0 u'_1 \ldots u'_p y'_{n+1} \ldots y'_m) : A_1 \to \ldots \to A_n \to \iota$$

where:

— $y_{n+1}, \ldots, y_m$ are fresh variables of types $A_{n+1}, \ldots, A_m$, respectively,
— y'_j is the η-long form of y_j,
— if u_0 is a constant or a variable, then $u'_0 = u_0$,
otherwise u'_0 is the η-long form of u_0,
— u'_j $(j > 0)$ is the η-long form of u_j.

Note that:

1 every simply typed term is η-equivalent to some η-long form;
2 if τ is in η-long form, then so are all τ' such that $\tau\, \beta\, \tau'$; and
3 our goal is to solve $\beta\eta$-equality between terms.

Consequently, in the following, we can assume without loss of generality that every simply typed term is in η-long form.

Every term in η-long form is of the shape $\lambda y_1 \ldots \lambda y_n\, u : A_1 \to \ldots \to A_n \to \iota$ where u is a term in η-long form of type ι. This term will be denoted $\lambda y_1 \ldots y_n.u$, or even $\lambda \vec{y}.u$ if there is no ambiguity.

[†] We follow here a widespread convention in theoretical computer science that considers ground types as being of order 1.

3. Matching problems

A *matching equation* is any equation of the form $u =_\beta v$ such that v is a closed term. A *solution* of a matching equation $u =_\beta v$ is a sequence of terms $(t_1, \ldots, t_n)$ such that $u[t_1/x_1, \ldots, t_n/x_n] =_\beta v$, where $x_1, \ldots, x_n$ are the free variables of u. Of course, we require that t_i and x_i are of same type. A *matching problem* is a (finite) list of matching equations, required to be solved uniformly.

Interpolation is the special case of Higher Order Matching in which, for some variable x, all equations of a matching problem are of the simplified form $(x\ u_1 \ldots u_n) =_\beta v$ where $u_1, \ldots, u_n$ and v are closed and v is of ground type, thus x being the only free variable of the problem.

Dual Matching is the generalization of Higher Order Matching to lists of matching equations and *disequations* between terms, that is, equations and disequations of the form $u =_\beta v$ or $u \neq_\beta v$ with v closed. *Dual Interpolation* is the special case of Dual Matching in which all equations and disequations of a given problem are of the same form as in Interpolation, that is, $(x\ u_1 \ldots u_n) =_\beta v$ or $(x\ u_1 \ldots u_n) \neq_\beta v$ where $u_1, \ldots, u_n, v$ are closed, v is of ground type, and x is the only free variable of the problem.

The *order* of a specific instance of Higher Order Matching, Dual Matching, Interpolation or Dual Interpolation, is defined as the maximal order of its free variables. When saying that a specific problem is *decidable at order N*, we mean that there exists an algorithm of some sort that takes as an input any instance of order at most N, and returns either the empty list, if this instance is not solvable, or a list containing one solution of this instance.

Lemma 3.1. An instance P of the Higher Order Matching, Dual Matching, Interpolation or Dual Interpolation problem, has a solution if and only if it has a closed solution.

Proof. Let $(t_1, \ldots, t_n)$ be any solution of P. Let σ be any substitution of constants that do not appear in the right-members of P for all variables free in $t_1, \ldots, t_n$. Check that $(\sigma(t_1), \ldots, \sigma(t_n))$ is still a solution of P. $\square$

3.1. *Observational equivalence*

Definition 3.1. Let v be a normal term. Two closed terms t and t' of the same type are *observationally equivalent with respect to v*, if and only if for all u, $u[t/x] =_\beta v$ if and only if $u[t'/x] =_\beta v$. This relation between t and t' will be denoted $t =_v t'$.

Note that in the particular case where v is a closed term, the observational equivalence of two terms t and t' with respect to v expresses that t and t' are solutions of the same matching equations and disequations of right member v. In other words, for any closed v, $t =_v t'$ if and only if for all u, t is a solution of $u =_\beta v$ if and only if t' is a solution of $u =_\beta v$; equivalently, for all u, $(t_1, \ldots, t, \ldots, t_n)$ is a solution of $u =_\beta v$ if and only if $(t_1, \ldots, t', \ldots, t_n)$ is a solution of $u =_\beta v$.

In this section, we shall prove that for all v there are only finitely many $=_v$ equivalence classes of any given type. By considering the quotient set of all closed terms by $=_v$, we can build a syntactic model of simply typed λ-calculus – a model where observational equivalence is *limited by v* – and legitimately consider the problem of its *decidability* up to some order N, that is, the existence of an algorithm that takes as an input an arbitrary type A of order at most N, and returns a complete set of representatives of all v-equivalence classes of type A. The decidability of limited models in general is still an open question. While proving the finiteness of the components of a limited model, we will simultaneously prove that the decidability of all limited models up to order N is equivalent to the decidability of Dual Interpolation at order N. Actually, the result we intend to prove is the following: Dual Interpolation is decidable at order four, consequently all limited models are decidable up to order four. This result will be linked to (fourth-order) Dual Matching with Lemma 3.5, which states that the decidability of limited models up to order N implies the decidability of Dual Matching at order N.

An alternative way to express observational equivalence, which we shall prove to coincide with this first relation, is given by the following definition.

Definition 3.2. Let v be a normal term. Let S_v be the (finite) set of all subterms of v of ground type. Two closed terms t and t' of same type $A_1 \to \dots \to A_n \to \iota$ are *extensionally equivalent with respect to* v if and only if for all $u_1 : A_1, \dots, u_n : A_n$, for all $s \in S_v$, $(t\, u_1 \dots u_n) =_\beta s \Leftrightarrow (t' u_1 \dots u_n) =_\beta s$.

This relation between t and t' will be denoted $t \sim_v t'$. The reader can check that requiring t and t' to be closed implies the following:

1 this relation depends only on the α-class of v, that is, if $v' =_\alpha v$, then $t \sim_v t'$ if and only if $t \sim_{v'} t'$; and
2 for all $v' =_\alpha v$ and for all $s' \in S_{v'}$, $t \sim_v t'$ implies $t \sim_{s'} t'$.

3.1.1. *Context lemma* Our first aim will be to prove that observational equivalence and extensional equivalence coincide.

Lemma 3.2. (Context Lemma) If $t \sim_v t'$, then $(u[t/x] : \iota =_\beta s \Leftrightarrow u[t'/x] : \iota =_\beta s)$ for all u and for all $s \in S_v$.

Proof. We shall prove, by induction on the length K of the left-normalization of $u[t/x]$, and for each K, by induction on the length L of u, that for all v, $t \sim_v t'$ and $u[t/x] =_\beta s \in S_v$ implies $u[t'/x] =_\beta s$. We can of course assume that t, t' and u are already in normal form. If $K = 0$, then $u[t/x] = s$. Either x is not free in u and the conclusion is immediate, or t, t' are of ground type and t equals some closed subterm of s, consequently t equals some closed subterm of v, and, by definition, $t = t'$. Suppose $K > 0$. The case of $u = (\lambda y_1 \dots y_n.u_0\ u_1 \dots u_n)$ follows immediately from the induction hypothesis. If $u = (\varepsilon\, w_1 \dots w_n)$ where ε is a constant or a variable not equal to x, then $s =_\alpha s' = (\varepsilon\ \lambda\vec{x}^1.s_1 \dots \lambda\vec{x}^n.s_n)$, where the $\vec{x}^j$ are such that no free variable in u is bound in s', so $u =_\alpha (\varepsilon\ \lambda\vec{x}^1.u_1 \dots \lambda\vec{x}^n.u_n)$ with $u_i[t/x] =_\beta s_i$, $s_i \in S_{s'}$. As $t \sim_v t'$, we also have $t \sim_{s'} t'$ (see Remark (2) Definition 3.2), and by the induction hypothesis (on L), $u_i[t'/x] =_\beta s_i$ for all i, hence $u[t'/x] =_\beta s$. The remaining case is $u = (x\ u_1 \dots u_n)$. We have $t = \lambda x_1 \dots x_n.w$ and $w[u_1[t/x]/x_1, \dots, u_n[t/x]/x_n] =_\beta s$. By the induction hypothesis (on K), $w[u_1[t'/x]/x_1, \dots, u_n[t'/x]/x_n] =_\beta s$, therefore $(t\ u_1[t'/x] \dots u_n[t'/x]) =_\beta s$. By hypothesis on t, t', $(t'\ u_1[t'/x] \dots u_n[t'/x]) =_\beta s$. □

Corollary 3.1. If $t \sim_v t'$, then $t =_v t'$.

Proof. Suppose $t \sim_v t'$ with $v = \lambda z_1 \dots z_p.v_0$. For $u = \lambda z_1 \dots z_p.u_0$ we have $u[t/x] =_\beta v$ implies $u_0[t/x] =_\beta v_0$. By the Context Lemma, $u_0[t'/x] =_\beta v_0$. Since t' is closed, $u[t'/x] =_\beta v$. The same property holds for t', □

Lemma 3.3. Conversely, if $t =_v t'$, then $t \sim_v t'$.

Proof. We use proof by contradiction. Suppose that $t =_v t'$, $(tu_1 \dots u_n) =_\beta s \in S_v$ and $(t'u_1 \dots u_n) \neq_\beta s$. Choose a position of s in v. Replace s with $(xu_1 \dots u_n)$ at this position. Call u the resulting term. Since t and t' are closed, $u[t/x] =_\beta v$ and $u[t'/x] \neq_\beta v$, which is absurd. □

These two results mean that in all the following, we can identify $=_v$ and $\sim_v$. In short, two terms t and t' will be said to be *v-equivalent* if and only if $t =_v t'$ if and only if $t \sim_v t'$.

3.1.2. *Finiteness and reduction of Dual Matching to Dual Interpolation*

Definition 3.3. Let v be a term in normal form. The quotient of the set of all closed terms by the relation $=_v$ will be called a *limited model* of simply typed λ-calculus. We call *type* and *order* of a v-equivalence class the (unique) type and order of its elements.

When saying that *limited models are decidable up to order* N, we mean that there exists an algorithm that can take as an input any normal term v and any type A of order at most N, and return a list of closed terms containing one representative of each v-equivalence class of type A. Of course, the decidability of limited models up to any order implies that in all models there are only finitely many classes of any given type.

Lemma 3.4. For all N, for all type A of order N, for all v, the number of v-equivalence classes of type A is finite[†]. The decidability of Dual Interpolation at order N implies the decidability of limited models up to order N.

Proof. We use induction on N.

($N = 1$). For t, t' of ground type, $t =_v t'$ boils down to $t =_\beta s =_\beta t'$ for some $s \in S_v$, or there exists no $s \in S_v$ such that $t =_\beta s$ or $t' =_\beta s$. Hence, for any ground type ι, there are at most $(|S_v| + 1)$ v-equivalence classes of type ι. In order to represent all of them, we only need to enumerate all closed subterms of v of type ι, plus a single constant of type ι not in S_v. This specifies a computable function R_1, which given a pair (ι, v) where ι is a ground type, returns a complete list of representatives of the v-equivalence classes of type ι.

($N > 1$). Let R be a function such that for all pairs (B, w) where B is of order at most $N - 1$ we have $R(B, w)$ is a list containing one representative of each w-equivalence class of type B. By the induction hypothesis, for all pairs (B, w), we have $R(B, w)$ is finite. Assuming R is undefined for all types of order at least N, we shall extend R to all types

[†] As mentioned in the introduction, this finiteness property can also be deduced from Statman's Finite Completeness theorem.

of order N, every new value being a finite list of terms. As a consequence, a limited model contains finitely many classes of any type of order N. Furthermore, assuming we are given a computable function R and a decision algorithm for Dual Interpolation at order N, our method allows us to specify a computable extension of R. Since R_1 is computable, the decidability of Dual Interpolation at order N implies that we can successively specify computable extensions of R_1 to order 2, order 3,. . . , order N.

Given an arbitrary type $A = A_1 \to \ldots \to A_n \to \iota$ of order N, and a normal term v, we shall define a value for $R(A, v)$ as follows.

Let S_v be the set of all subterms of v of ground type. Let V be the set of all free variables of S_v. Let D and E be two sets of constants such that D (respectively, E) contains, for each $z : B \in V$, a fresh constant of type B.

In the following, $[D/V]$ (respectively, $[E/V]$) is shorthand for the simultaneous substitution of each variable in V by the corresponding new constant in D (respectively, E), while $[V/D]$ (respectively, $[V/E]$) denotes the opposite substitution.

For each $s \in S_v$ of type ι, let $Arg(A, s) = (\Pi_{i=1}^{n} R(A_i, s[D/V]))[V/D]$. Let x be a fresh variable of type A. Let $Car(A, v)$ be the (finite) set of all dual interpolation problems Φ constructed with the following method:

for each $s : \iota \in S_v$ and for each $(r_1, \ldots, r_n) \in Arg(A, s)$, add to Φ either

(a) $(x\, r_1 \ldots r_n)[D/V] =_\beta s[D/V])$ and $(x\, r_1 \ldots r_n)[E/V] =_\beta s[E/V])$, or
(b) $(x\, r_1 \ldots r_n)[E/V] \neq_\beta s[E/V]$ and $(x\, r_1 \ldots r_n)[E/V] \neq_\beta s[E/V]$.

Let T be the set of all closed terms of type A that do not contain the constants of D and E. We let $R(A, v)$ be any of the finite lists of elements of T containing a solution of each problem in $Car(A, v)$ that has a solution in T.

We claim that:

1 for all terms $t : A$ there exists in T a term v-equivalent to t,
2 every term in T is a solution of a unique dual problem in $Car(A, v)$, and
3 two terms in T are v-equivalent if and only if they are solutions of the same problem.

Note that if the three properties are satisfied, $R(A, v)$ is a complete set of representatives of all v-equivalence classes of type A. Furthermore, if R is computable, $Car(A, v)$ can be computed as a function of (A, v). If Dual Interpolation is decidable at order N, a value for $R(A, v)$ can be computed as a function of $Car(A, v)$.

Let us prove (1). Let σ be any substitution of fresh constants for all constants in D and E. Let $t : A$ be any closed term. Then $\sigma(t) \in T$. As the constants in D and E do not appear in v, we have $u[t/x] =_\beta v$ implies $u[\sigma(t)/x] =_\beta v$ for all u. As the constants substituted in t do not appear in v, we have $u[t/x] \neq_\beta v$ implies $u[\sigma(t)/x] \neq_\beta v$ for all u. Thus $\sigma(t) =_v t$.

Considering the definition of $Car(A, v)$, it is sufficient, in order to show (2), to prove that for any $t \in T$ and for any $(r_1, \ldots, r_n) \in Arg(A, s)$, $(t\, r_1 \ldots r_n)[D/V] =_\beta s[D/V]$ if and only if $(t\, r_1 \ldots r_n)[E/V] =_\beta s[E/V]$. Again, this result follows from the fact that the constants in D and E do not appear in t.

Let us prove (3). If t and t' in T are v-equivalent, then for any $(r_1, \dots, r_n) \in Arg(A, s)$, we have $(t\, r_1 \dots r_n) =_\beta s$ if and only if $(t\, r_1 \dots r_n)[D/V] =_\beta s[D/V]$ if and only if $(t\, r_1 \dots r_n)$ $[E/V] =_\beta s[E/V]$ (see the proof of (2)). Furthermore, $(t\, r_1 \dots r_n) =_\beta s$ if and only if $(t' r_1 \dots r_n) =_\beta s$ (by definition) if and only if $(t' r_1 \dots r_n)[D/V] =_\beta s[D/V]$ if and only if $(t' r_1 \dots r_n)[E/V] =_\beta s[E/V]$ (again, see the proof of (2)). Hence, t and t' are solutions of the same problem. Conversely, assume t and t' in T are solutions of the same problem. Suppose t and t' are not v-equivalent. By definition, there exists $u_1, \dots, u_n$, $s \in S_v$ such that, for instance, $(t\, u_1 \dots u_n) =_\beta s$ and $(t' u_1 \dots u_n) \neq_\beta s$. We have $(t\, u_1 \dots u_n)[D/V] =_\beta s[D/V]$ (because t is closed) and $(t' u_1 \dots u_n)[D/V] \neq_\beta s[D/V]$ (because no constant of D appears in t'). By hypothesis, there exists in $\Pi_{i=1}^{n} R(A_i, s[D/V])$ a sequence of the form $(r_1, \dots, r_n)[D/V]$ such that we have $(r_1, \dots, r_n) \in Arg(A, s)$ with $u_i[D/V] =_{s[D/V]} r_i[D/V]$ for all i. Now $(t\, r_1 \dots r_n)[D/V] =_\beta s[D/V]$ and thus we have $(t' r_1 \dots r_n)[D/V] \neq_\beta s[D/V]$, which is a contradiction. □

Lemma 3.5. The decidability of limited models up to order N implies the decidability of Dual Matching at order N.

Proof. Let $v_1, \dots, v_n$ be the right-members of the (dis)equations of a dual matching problem P of order N, of free variables $x_1 : A_1, \dots, x_p : A_p$. Let $w = (a\, v_1 \dots v_n) : \iota$, where a is a fresh constant of suitable type. Suppose we are given an algorithm that, fed with each pair (A_j, w), returns a complete set R_j of representatives of each w-class of type A_j. Note that $=_{v_i} \supset =_w$, and hence R_j is also complete with respect to each v_i. Let $(t_1, \dots, t_p)$ be any (closed, by Lemma 3.1) solution of P. By hypothesis, there exists in $\Pi_{j=1}^{p} R_j$ a sequence $(r_1, \dots, r_p)$ such that $r_j =_w t_j$ for all j. As $r_j =_{v_i} t_j$ for all i, j, the sequence $(r_1, \dots, r_p)$ is also a solution of P. □

4. Decidability of fourth-order Dual Interpolation

The decidability of Dual Interpolation at order four is not very hard to understand in principle. All terms being in η-long form, a solution t of an interpolation equation $(x\, u_1 \dots u_n) =_\beta v$ is necessarily of the form $\lambda y_1 \dots \lambda y_n.u$. Aside from the y_i's, subterms of u may contain additional free variables (for example, x is a subterm of $\lambda y\, (y\, \lambda x\, x)$). Some of them are never substituted for in any reduction of $(t\, u_1 \dots u_n)$: these are the bound variables introduced by constants of higher order type (like z_1, z_2 in $(c\lambda z_1(z_1 \lambda z_2\, z_2))$). All other additional variables are of order at most $N - 3$, where N is the order of t. If $N = 4$, all bound variables of t substituted for in the reduction process are, aside from the y_i's, of ground type.

A careful examination of the reduction of t applied to $u_1, \dots, u_n$ will reveal the following. When considering an arbitrary first-order subterm w of t, there are two possibilities: either w is 'destroyed' by the reduction, *i.e.*, we are allowed to graft at the position of w any term of same ground type without affecting the normal form of $(t\, u_1\, \dots u_n)$; or w is effectively involved in the reduction process, as formalized in Definition 4.1. In this latter case, at order four, when we substitute $u_1, \dots, u_n$ for $y_1, \dots, y_n$ in w, and compute the normal form of the resulting term, in other words when we *evaluate* the subterm w with respect to the equation $(x\,u_1 \dots u_n) =_\beta v$, the resulting normal form is necessarily of the following form, up to renaming of its free and bound variables: a subterm of v, where some subterms have been replaced by variables of ground type – which are actually all bound variables of t free in w, distinct from the y_i, and substituted for in all reductions of $(t\,u_1 \dots u_n)$. In short, the evaluation of w will yield a subterm of v *pruned* with variables of ground type. A concrete example will clarify this point: if v is of the form $(a(bcd))$, then either w is destroyed by the reduction – in which case, by convention, its evaluation equals $\perp$ – or the evaluation of w must be of either of the following forms: $(a(bcd))$, (bcd), c, d, $(a(bxd))$, $(a(bcx))$, $(a(bxx'))$, (ax), x, (bxd), (bcx), or (bxx'), where x, x' are variables of ground type. Hence, up to renaming of free and bound variables, the number of all possible evaluations with respect to a fourth-order interpolation equation is finite, and can be bounded as a function of its right-member.

Once this result is established, the decidability of Dual Interpolation at order four is straightforward. The notion of evaluation is extended to inequalities. Finiteness still holds when, given a fourth-order dual problem Φ, we consider the set of all possible evaluations with respect to Φ, that is, all lists built by evaluating some subterm of a solution with respect to each equality and inequality of the problem. Now, if t is an arbitrary solution of Φ, and contains a path of length greater than the number of all possible evaluations with respect to Φ – this number can be bounded as a function of the sequence of right-members of Φ – then there are two positions on this path such that the subterm at the lowest position has the same evaluation as the highest one (again, up to renaming of free and bound variables). In that case, we can take the lowest subterm, perform all necessary renamings, graft it at the highest position, and obtain a new solution of Φ of shorter length.

Unfortunately, the method fails at higher orders. Order five is the first such that an evaluation may contain an arbitrary number of second-order variables, stacked one upon another.

4.1. *Preliminaries*

Definition 4.1. A position in a term u is said to be *accessible* if the graft of a fresh constant at this position yields a term of distinct normal form. For instance, the position of c is accessible in $(\lambda xy.x\,c\,d)$, whereas the position of d is inaccessible.

Lemma 4.1. Let v be a normal term, let $t = \lambda x_1 \dots x_n.t_0$ be a normal term of order at most two. If $u[t/z] =_\beta v$, and if at least one position of t is accessible in $u[t/z]$, then t_0 is equal, up to renaming of bound variables, to some subterm of v pruned with the variables $x_1, \dots, x_n$.

Proof. We use induction on the length of v, and for each length, induction on the length of u, which is supposed to be already in normal form.

If $u = (z\,u_1 \dots u_n)$, then $u[t/z] = (t\,u_1[t/z] \dots u_n[t/z]) =_\beta (t\,u'_1 \dots u'_n)$, where u'_i is the normal form of $u_i[t/z]$. As t is of order at most two, $x_1, \dots, x_n$ are of ground type, hence $t_0[u'_1/x_1, \dots, u'_n/x_n] =_\alpha v$, and the result is obvious. If $u =_\alpha \lambda\vec{y}.u_0$ with $u[t/x] =_\alpha \lambda\vec{y}.u_0[t/x]$, then $v =_\alpha \lambda\vec{y}.v_0$ and $u_0[t/x] =_\beta v_0$. Otherwise, u is of the form $(\varepsilon\,u_1 \dots u_n)$, where ε is a constant or a variable not equal to z. In that case, $v = (\varepsilon\, v_1 \dots v_n)$, and there is at least one u_i such that $u_i[t/z] =_\beta v_i$, a position of t being accessible in $u_i[t/x]$. In the last two cases, the conclusion follows easily from the induction hypothesis. □

4.2. *Key Lemma*

Lemma 4.2. (Key Lemma at order 4) Let u_0, v be normal terms of ground type. Let w be a subterm of u_0 of ground type. Let $y_1, \dots, y_n$ be variables of order at most three. Suppose a position of w in u_0 is an accessible position in $u_0\,[u_1/y_1, \dots, u_n/y_n] =_\beta v$. Then the normal form of $w[u_1/y_1, \dots, u_n/y_n]$ is equal, up to renaming of free and bound variables, to some subterm of v pruned with variables of ground type.

Proof. We use induction on the length of the position of w in u_0. If $w = u_0$, then $w[u_1/y_1, \dots, u_n/y_n] =_\beta v$. Assume that w is a strict subterm of u_0. There are two possible cases.

1 $u_0 = (y_i\, t_1 \dots t_p)$, the position of w being in t_j. Because the position of w in u_0 is an accessible position in $u_0\,[\vec{u}/\vec{y}] = (u_i\, t_1[\vec{u}/\vec{y}] \dots t_p[\vec{u}/\vec{y}]) =_\beta v$, we have:

(a) The position of t_j in u_0 is an accessible position in $(u_i\, t_1[\vec{u}/\vec{y}] \dots t_p[\vec{u}/\vec{y}])$. Hence, $t_j[\vec{u}/\vec{y}]$ has at least one accessible position in $U[t_j[\vec{u}/\vec{y}]/z_j] =_\beta v$, where for fresh $z_1, \dots, z_p$, we have $u_i =_\alpha \lambda z_1 \dots z_p.u_i^0$ and U is equal to u^0 where t_k is substituted for z_k for all k but j. As y_i is of order at most three, $t_j[\vec{u}/\vec{y}]$ is of order at most two, hence $t_j =_\alpha \lambda \vec{x}.t_j^0$ where $\vec{x}$ are fresh variables of ground types. By Lemma 4.1, the normal form v_0 of $t_j^0[\vec{u}/\vec{y}]$ is equal, up to renaming of its bound variables, to some subterm of v pruned with the $\vec{x}$.

(b) The position of w in t_j^0 that is the postfix of the position of w in u_0, is an accessible position in $t_j^0[\vec{u}/\vec{y}] =_\beta v_0$. By the induction hypothesis, the normal form of $w[\vec{u}/\vec{y}]$ is equal, up to renaming of free and bound variables, to some subterm of v_0 pruned with variables of ground type, and *a fortiori* to some subterm of v pruned with variables of ground type.

2 $u_0 = (\varepsilon\, t_1 \dots t_p)$ where ε is a constant or a variable not in $\vec{y}$, the position of w being in t_j. We have $v =_\alpha (\varepsilon\, v_1 \dots v_p)$, and for $t_j =_\alpha \lambda \vec{z}.t_j^0$ where $\vec{z}$ are fresh variables, $v_j =_\alpha \lambda \vec{z}.v_j^0$. Furthermore, the position of w in t_j^0 which is the postfix of the position of w in u_0, is an accessible position in $t_j^0[\vec{u}/\vec{y}] =_\beta v_j^0$. The conclusion follows from the induction hypothesis. □

4.3. *Evaluation at order four*

Definition 4.2. Let F be a fourth-order interpolation equation or disequation, of left-member $(x\, u_1 \dots u_n)$, of right-member v. Let $t = \lambda y_1 \dots y_n.u_0$ be a solution of F. The *evaluation* of a position of a subterm $w : \iota$ of t of ground type, with respect to F, is defined as:

— the normal form of $w[\vec{u}/\vec{y}]$, if, up to renamings, this normal form equals a subterm of v pruned with variables of ground type;
— $\bot : \iota$ otherwise.

The evaluation of a position π in t, with respect to a fourth-order dual problem $\Phi = \langle F_1, \dots, F_n \rangle$ of which t is a solution, is equal to $\langle e_1, \dots, e_n \rangle$, where e_i is the evaluation of π with respect to F_i. This list will be called the Φ-*evaluation* of π.

The next two lemmas are immediate consequences of the definition of evaluation and Lemma 4.2.

Lemma 4.3. Up to renaming of free and bound variables, the set of all evaluations with respect to a fourth-order dual problem is a finite set, whose cardinal can be bounded as a computable function of the right-members of the problem.

Lemma 4.4. Let t be any solution of a fourth-order dual problem Φ. Suppose a position π in t yields the Φ-evaluation E, while the position π' in t yields an evaluation equal to E up to a renaming, which, applied to the subterm at position π' in t, yields the term w. Then, if we graft w at position π in t, the resulting term is still a solution of Φ.

4.4. *Main results*

Theorem 4.1. Fourth-order Dual Interpolation is decidable.

Proof. Call a *path* in a term t any sequence of successive positions of subterms of ground types. Let Φ be a fourth-order dual problem. Let K be the cardinal of all Φ-evaluations, up to renaming of free and bound variables. By Lemma 4.3, this integer is well-defined, and can be computed as a function of the right-members of Φ.

Let t be any solution of Φ. We can assume that all paths in t are of length shorter that K. In the contrary case, two positions on this path must have the same Φ-evaluation, up to some renaming applied to the evaluation of the lowest subterm. If we do apply this renaming to the lowest subterm, this will yield a term that can be grafted at the upper position, so that, by Lemma 4.4, the resulting term is still a solution of Φ. This operation can be repeated as many times as required, until we get a solution of Φ where all paths are of length less than K.

We can also assume that t is closed, and, furthermore, that all constants in t belong to a finite set C that contains, on the one hand, all constants that appear in the right-members of Φ, and on the other hand, one constant that does not appear in the right-members of Φ for each ground type. Indeed, let ε be a constant or a variable that does not belong to this set. Suppose ε appears in t in a subterm of the form $(\varepsilon\, t_1 \dots t_n) : \iota$. Any position of this subterm is necessarily inaccessible with respect to all equations of Φ, and we can graft a fresh constant of type ι at this position without affecting their results. Furthermore, all disequations will still be satisfied, as the grafted constant does not appear in their right-members.

In other words, Φ has a solution if and only if some solution of Φ belongs to the set of all closed normal terms in η-long form, whose constants belong to C, and whose paths are of length at most K. This latter set is clearly finite and computable. □

Applying Lemmas 3.5 and 3.4, we finally obtain the following theorem.

Theorem 4.2. All limited models are decidable up to order four. Dual Matching is decidable at order four. Higher Order Matching is decidable at order four.

References

Barendregt, H. (1984) *The Lambda Calculus, its Syntax and Semantics*, North Holland.

Comon, H. and Jurski, Y. (1997) Higher-order matching and tree automata. In: Proceedings Conf. on Computer Science Logic. *Springer-Verlag Lecture Notes in Computer Science* **1414** 157–176.

Dowek, G. (1993) Third Order Matching is Decidable. In: Proceedings of Logic in Computer Science. *Annals of Pure and Applied Logic*.

Dowek, G (1993) The undecidability of pattern matching in calculi where primitive recursive functions are representable. *Theorical Computer Science* **107**.

Goldfarb, W. D. (1989) The undecidability of the second-order unification problem. *Theorical Computer Science* **355** 225–230.

Hindley J. R. and Seldin, J.P. (1986) *Introduction to Combinators and λ-Calculus*, Cambridge University Press.

Huet, G. (1976) *Résolution d'équations dans les langages d'ordre 1, 2, ... ω,* Thèse de doctorat d'état, Université Paris VII.

Krivine J. L. (1993) *Lambda Calculus, Types and Models*, Ellis Horwood series in computer and their applications 1–66.

Loader, R (1995) Lambda Definability is Undecidable. In: Anderson, A. and Zeleny, M. (eds.) *Church Memorial Volume*, Kluwer Academic Press (to appear).

Loader, R. (1995) Unary PCF is Decidable. *Theorical Computer Science* (to appear).

Padovani, V. (1995) Decidability of All Minimal Models. In: Proceedings of the Annual Meeting Types for Proof and Programs - Torino 1995. *Springer-Verlag Lecture Notes in Computer Science* **1158**.

Padovani, V. (1996) *Filtrage d'Ordre Supérieur*, Thèse de doctorat, Universit/'e Paris VII.

Plotkin, G. (1993) Lambda-Definability and Logical Relations. Memorandum SAI-RM-4, School of Artificial Intelligence, University of Edinburgh.

Schimdt-Schauß, M. (1999) Decidability of Behavioral Equivalence in Unary PCF. *Theorical Computer Science* (to appear).

Schubert, A. (1997) Linear interpolation for the higher order matching problem. In: Proceedings of TAPSOFT'97. *Springer-Verlag Lecture Notes in Computer Science* **1214**.

Springintveld, J. (1995) Third Order Matching in the polymorphic lambda calculus. In: Proceedings of Higher Order Algebra, Logic and Term Rewriting. *Springer-Verlag Lecture Notes in Computer Science* **1074**.

Springintveld, J. (1995) Third Order Matching in the presence of type constructors. In: Proceedings of The Second International Conference on Typed Lambda Calculi and Applications. *Springer-Verlag Lecture Notes in Computer Science* **915**.

Statman, R. (1982) Completeness, invariance and lambda-definability. *Journal of Symbolic Logic* **47**.

Statman, R. and Dowek, G. (1992) On Statman's completeness theorem, Technical Report, CMU-CS-92-152, University of Carnegie Mellon.

Wolfram, D. A. (1992) *The clausal theory of types*, Cambridge University Press.