On the complexity of higher-order matching in the linear λ-calculus

Sylvain Salvati and Philippe de Groote

LORIA UMR n^o 7503 – INRIA
Campus Scientifique, B.P. 239
54506 Vandœuvre lès Nancy Cedex – France
e-mail: salvati@loria.fr, degroote@loria.fr

Abstract. We prove that linear second-order matching in the linear λ-calculus with linear occurrences of the unknowns is NP-complete. This result shows that context matching and second-order matching in the linear λ-calculus are, in fact, two different problems.

1 Introduction

Higher-order unification, which consists in solving a syntactic equation between two simply-typed λ-terms (modulo β, or modulo $\beta\eta$), is undecidable [9], even in the second-order case [7]. Consequently, several restrictions of the problem have been introduced and studied in the literature(see [6] for a survey).

Higher-order matching is such a restriction. It consists in solving equations whose right-hand sides do not contain any unknown. This problem, which is indeed simpler, has been shown to be decidable in the second-order case [10], in the third-order case [5], and in the fourth order case [13]. Starting form the sixth-order case, higher-order matching modulo β is undecidable [12]. On the other-hand the decidability of higher-order matching modulo $\beta\eta$ is still open.

Another restriction consists in studying unification in the linear λ-calculus where every λ-abstraction $\lambda x.\, M$ is such that M contains exactly one free occurrence of x. The problem of unification in the linear λ-calculus, in the second order case, is related to context unification [1, 2], which consists of unifying trees in which occur second-order variables (*i.e.*, variables ranging on "trees with holes"). It has been studied by Levy under the name of *linear second-order unification* [11]. Nevertheless, its decidability is still open. On the other hand, the more restricted problem of higher-order matching in the linear λ-calculus has been shown to be decidable and even NP-complete [8] which is also the case of context matching [15]. A related problem consists in deciding whether a matching problem between simply typed λ-terms admits a linear solution. This more general problem is also decidable [4].

Finally, several other restrictions concern the way the unknowns occur in the equation. In particular, another notion of linearity appears in the literature. Indeed, linear unification (or matching) also designates equations whose unknowns occur only once.

In this paper, we will be concerned with these two different notions of linearity (equations between linear λ-terms, or linear occurrences of the unknowns). In order not to confuse them, we will speak of *matching in the linear λ-calculus*, in the first case, and we will use the expression *linear matching* for the second case. This question of vocabulary being settled, we may state our main result: *linear second-order matching in the linear λ-calculus is NP-complete.* Such a complexity, at first sight, might be surprising. Indeed, it seems to be folklore that context matching and second-order matching in the linear λ-calculus are equivalent problems. Our result, however, shows that this is not quite the case. Indeed, linear context matching is polynomial [15].

In fact, the key difference between a context $C[x_1, \ldots, x_n]$ and a second-order λ-term is the following : in a linear λ-term $\lambda x_1 \ldots x_n.C$, the order in which the λ-variables are abstracted does not especially correspond to the order in which these variables occur in the body of the term. This slight difference is sufficient to make our problem NP-complete while linear context matching is polynomial.

The paper is organized as follows. Section 2 contains prerequisite basic notions, and defines precisely what is linear second-order matching in the linear λ-calculus. In Section 3, we define a variant of the satifiability problem (which we call 1-neg-sat), and we prove its NP-completeness. Section 4 shows how to reduce 1-neg-sat to linear second-order matching in the linear λ-calculus, and Section 5 proves the correctness of the reduction. Finally, in Section 6, we state some related results.

2 Higher-order matching in the linear λ-calculus

This section reviews basic definitions and fixes the notations that we use in the sequel of the paper.

Definition 1. *Let $\mathscr{A}$ be a finite set, the elements of which are called* atomic types*. The set $\mathscr{F}$ of* linear functional types built upon $\mathscr{A}$ *is defined according to the following grammar:*

$$\mathscr{F} \quad ::= \quad \mathscr{A} \quad | \quad (\mathscr{F} \multimap \mathscr{F}).$$

■

We let the lowercase Greek letters $(\alpha, \beta, \gamma, \ldots)$ range over $\mathscr{F}$, and we adopt the usual convention that $\alpha_1 \multimap \alpha_2 \multimap \cdots \alpha_n \multimap \alpha$ stands for $\alpha_1 \multimap (\alpha_2 \multimap (\cdots (\alpha_n \multimap \alpha) \cdots))$, and we write $\alpha^n \multimap \beta$ for:

$$\underbrace{\alpha \multimap \ldots \multimap \alpha}_{n\times} \multimap \beta$$

The order of such a linear functional type is defined as usual:

$$\begin{aligned} &\text{order}(a) = 1 \text{ if } a \in \mathscr{A} \\ &\text{order}(\alpha \multimap \beta) = \max(\text{order}(\alpha) + 1, \text{order}(\beta)) \end{aligned}$$

Then, the notion of raw λ-term is defined as follows.

Definition 2. *Let $(\Sigma_\alpha)_{\alpha\in\mathscr{F}}$ be a family of pairwise disjoint finite sets indexed by $\mathscr{F}$, whose almost every member is empty. Let $(\mathscr{X}_\alpha)_{\alpha\in\mathscr{F}}$ and $(\mathscr{Y}_\alpha)_{\alpha\in\mathscr{F}}$ be two families of pairwise disjoint countably infinite sets indexed by $\mathscr{F}$, such that $(\bigcup_{\alpha\in\mathscr{F}}\mathscr{X}_\alpha)\cap(\bigcup_{\alpha\in\mathscr{F}}\mathscr{Y}_\alpha)=\varnothing$. The set $\mathscr{T}$ of raw λ-terms is defined according to the following grammar:*

$$\mathscr{T} \;::=\; \Sigma \mid \mathscr{X} \mid \mathscr{Y} \mid \lambda\mathscr{X}.\,\mathscr{T} \mid (\mathscr{T}\,\mathscr{T}),$$

where $\Sigma=\bigcup_{\alpha\in\mathscr{F}}\Sigma_\alpha$, $\mathscr{X}=\bigcup_{\alpha\in\mathscr{F}}\mathscr{X}_\alpha$, and $\mathscr{Y}=\bigcup_{\alpha\in\mathscr{F}}\mathscr{Y}_\alpha$. ■

In this definition, Σ is the set of constants, $\mathscr{X}$ is the set of λ-*variables*, and $\mathscr{Y}$ is the set of *meta-variables* or *unknowns*. We let the lowercase roman letters $(\mathrm{a},\mathrm{b},\mathrm{c},\ldots)$ range over the constants, the lowercase italic letters $(x,y,z,\ldots)$ range over the λ-variables, the uppercase bold letters $(\mathbf{X},\mathbf{Y},\mathbf{Z},\ldots)$ range over the unknowns, and the uppercase italic letters $(M,N,O,\ldots)$ range over the λ-terms.

We write $h(M_1,\ldots,M_n)$ for a λ-term of the form $((\ldots(h\,M_1)\ldots)\,M_n)$, where h is either a constant, a λ-variable, or a meta-variable. Given such a term, h is called the head of the term.

The notions of free and bound occurrences of a λ-variable are defined as usual, and we write $\mathrm{FV}(M)$ for the set of λ-variables that occur free in a λ-term M. Finally, a λ-term that does not contain any meta-variable is called a *pure λ-term*.

We then define the notion of term of the linear λ-calculus..

Definition 3. *The family $(\mathscr{T}_\alpha)_{\alpha\in\mathscr{F}}$ of sets of terms of the linear λ-calculus is inductively defined as follows:*

1. *if $\mathrm{a}\in\Sigma_\alpha$ then $\mathrm{a}\in\mathscr{T}_\alpha$;*
2. *if $\mathbf{X}\in\mathscr{Y}_\alpha$ then $\mathbf{X}\in\mathscr{T}_\alpha$;*
3. *if $x\in\mathscr{X}_\alpha$ then $x\in\mathscr{T}_\alpha$;*
4. *if $x\in\mathscr{X}_\alpha$, $M\in\mathscr{T}_\beta$, and $x\in\mathrm{FV}(M)$, then $\lambda x.\,M\in\mathscr{T}_{(\alpha\multimap\beta)}$;*
5. *if $M\in\mathscr{T}_{(\alpha\multimap\beta)}$, $N\in\mathscr{T}_\alpha$, and $\mathrm{FV}(M)\cap\mathrm{FV}(N)=\varnothing$, then $(M\,N)\in\mathscr{T}_\beta$.* ■

Clauses 4 and 5 imply that any term $\lambda x.\,M$ of the linear λ-calculus is such that there is exactly one free occurrence of x in M. Remark, on the other hand, that constants and unknowns may occur several times in the same linear λ-term.

We define the set of terms of the linear λ-calculus to be $\bigcup_{\alpha\in\mathscr{F}}\mathscr{T}_\alpha$. One easily proves that the sets $(\mathscr{T}_\alpha)_{\alpha\in\mathscr{F}}$ are pairwise disjoint. Consequently, we may define the type of a term M to be the unique linear type α such that $M\in\mathscr{T}_\alpha$. This allows the order of a term to be defined as the order of its type. In particular, we will speak about the order of a meta-variable.

The notions of α-conversion, η and β-reduction are defined as usual. In particular, we write $\twoheadrightarrow_\beta$ for the relation of β-reduction, and $=_\beta$ for the relation of β-conversion.

We let $M[x{:=}N]$ denote the usual capture-avoiding substitution of a λ-variable by a λ-term. Similarly, $M[\mathbf{X}{:=}N]$ denotes the capture-avoiding substitution of a meta-variable by a λ-term. We abbreviate $M[x_1{:=}N_1]\cdots[x_n{:=}N_n]$ as $M[x_i{:=}N_i]_{i=1}^n$.

We now give a precise definition of the matching problem with which we are concerned.

Definition 4. *A* matching problem in the linear λ-calculus *is a pair of terms of the linear λ-calculus $\langle M, N\rangle$ of the same type such that N is pure (i.e., does not contain any meta-variable).*

Such a problem admits a solution if and only if there exists a substitution $(\mathbf{X}_i{:=}O_i)_{i=1}^n$ such that $M[\mathbf{X}_i{:=}O_i]_{i=1}^n =_\beta N$, where $\{\mathbf{X}_1, \ldots, \mathbf{X}_n\}$ is the set of meta-variables that occur in M. ■

In the above definition, we have taken the relation of β-conversion to be the notion of equality between λ-terms. Nevertheless, all the results we will establish remain valid when taking the relation of $\beta\eta$-conversion as the notion of equality.

In the sequel of this paper, a pair of λ-terms $\langle M, N\rangle$ obeying the conditions of the above definition will also be called a *syntactic equation*. The order of such an equation is defined to be the maximum of the orders of the meta-variables occurring in left-hand side the equation. Finally, such an equation is said to be linear if the meta-variables occurring in its left-hand side occur only once.

In this paper, we will mainly be concerned with linear second-order matching in the linear λ-calculus, i.e, the problem of solving a linear second-order syntactic equation between terms of the linear λ-calculus.

3 1-Neg-sat

In this section, we define a variant of the satisfiability problem, due to Kilpeläinen [14].

We first remind the reader of some basic definitions. Given a finite set $\mathcal{A} = \{a_1, \ldots, a_n\}$ of boolean variables, a literal is defined to be either a boolean variable a_i or its negation $\neg a_i$. A clause is a finite set of literals, and a satisfiability problem consists in a finite set of clauses. A positive literal a_i is satisfied by a valuation $\eta : \mathcal{A} \to \{0, 1\}$ if and only if $\eta(a_i) = 1$, and a negative literal $\neg a_i$ is satisfied if and only if $\eta(a_i) = 0$, in which case we also write $\eta(\neg a_i) = 1$. Then, a satisfiability problem $\mathcal{C}$ admits a solution if and only if there exists a valuation $\eta : \mathcal{A} \to \{0, 1\}$ such that for all $C \in \mathcal{C}$ there exists $l \in C$ with $\eta(l) = 1$. As is well known, satisfiability is NP-complete [3].

We now introduce the variant of the satisfiability problem that we call *1-neg-sat*.

Definition 5. *Let $\mathcal{A}$ be a finite set of boolean variables, and $\mathcal{C}$ be a finite set of clauses over $\mathcal{A}$. $\mathcal{C}$ is called a 1-neg-sat problem if and only if for all $a \in \mathcal{A}$, there exists exactly one $C \in \mathcal{C}$ such that $\neg a \in C$.* ■

The next result is due to Kilpeläinen [14].

Lemma 1. *1-Neg-sat is NP-complete.*

Proof. We show that any satisfiability problem can be reduced to a 1-neg-sat-problem.

Let $\mathcal{C}$ be a finite set of clauses over the set of boolean variables $\mathcal{A} = \{a_1, \ldots, a_n\}$. We introduce a set $\mathcal{B} = \{b_1, \ldots, b_n\}$ of fresh boolean variables, and we define $\mathcal{D}$ to be the set of clauses $\bigcup_{i=1}^{n} \{\{a_i, b_i\}, \{\neg a_i, \neg b_i\}\}$. Clearly, any valuation η that satisfies $\mathcal{D}$ is such that $\eta(a_i) = 0$ if and only if $\eta(b_i) = 1$. Conversely, any valuation η such that $\eta(a_i) = 0$ if and only if $\eta(b_i) = 1$ satisfies $\mathcal{D}$.

Then we define $\mathcal{C}^*$ as the set of clauses obtained from $\mathcal{C}$ by replacing each occurrence of $\neg a_i$ by b_i. By construction, $\mathcal{C}^* \cup \mathcal{D}$ is a 1-neg-sat problem. Moreover, any valuation that satisfies this problem satisfies $\mathcal{C}$. Conversely, given a valuation η that satisfies $\mathcal{C}$, the valuation η' such that $\eta'(a_i) = \eta(a_i)$ and $\eta'(b_i) = \neg\eta(a_i)$ satisfies $\mathcal{C}^* \cup \mathcal{D}$. □

4 Reduction of 1-neg-sat

In this section we show how to associate to any 1-neg-sat problem a linear second order syntactic equation in the linear λ-calculus.

Let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be a 1-neg-sat problem defined over the set of boolean variables $\mathcal{A} = \{a_1, \ldots, a_n\}$. We define $\mathrm{neg} : \mathcal{A} \to \mathcal{C}$ to be the function such that:

$$\mathrm{neg}(a_i) = C_j \quad \text{if and only if} \quad \neg a_i \in C_j.$$

Similarly, we define $\mathrm{pos} : \mathcal{A} \to \mathscr{P}(\mathcal{C})$ such that:

$$\mathrm{pos}(a_i) = \{C_j \in \mathcal{C} \mid a_i \in C_j\}.$$

For each $i \in \{1, \ldots, n\}$, let m_i be the cardinality of $\mathrm{pos}(a_i)$, and define $\psi_i : \{1, \ldots, m_i\} \to \{1, \ldots, m\}$ to be a function such that:

$$\mathrm{pos}(a_i) = \{C_{\psi_i(1)}, \ldots, C_{\psi_i(m_i)}\}.$$

In case $m_i = 0$, by convention, ψ_i is defined to be the empty function.

Now, let o be an atomic type. In order to define the syntactic equation associated to $\mathcal{C}$, we introduce the following constants and meta-variables:

1. a constant a of type o;
2. a constant f of type $o^n \multimap o$;
3. for each clause $C \in \mathcal{C}$, a constant $\overline{C}$ of type $o^m \multimap o$;
4. a meta-variable $\mathbf{X}$ of type $o^m \multimap o$;
5. m^2 meta-variables $\mathbf{X}_{11}, \ldots, \mathbf{X}_{1m}, \ldots, \mathbf{X}_{m1}, \ldots, \mathbf{X}_{mm}$ of type o.

For $(i, j) \in \{1, \ldots, n\} \times \{1, \ldots, m\}$, we define the following terms:

$$R_{ij} = \begin{cases} \overline{C_{\psi_i(j)}}(a, \ldots, a) \text{ if } j \leq m_i \\ a \text{ otherwise.} \end{cases}$$

Then, for $i \in \{1, \ldots, n\}$, we define:

$$R_i = \overline{\mathrm{neg}(a_i)}(R_{i1}, \ldots, R_{im})$$

Finally, the syntactic equation $\langle L_{\mathcal{C}}, R_{\mathcal{C}} \rangle$, associated to $\mathcal{C}$, is defined as follows:

$$L_{\mathcal{C}} = \mathbf{X}(\overline{C_1}(\mathbf{X}_{11}, \ldots, \mathbf{X}_{1m}), \ldots, \overline{C_m}(\mathbf{X}_{m1}, \ldots, \mathbf{X}_{mm}))$$

$$R_{\mathcal{C}} = f(R_1, \ldots, R_n)$$

Let us illustrate the above reduction by an example. Consider the following clauses:

$$C_1 = \{a_1\}, C_2 = \{a_1, a_2\}, C_3 = \{\neg a_1, \neg a_2\}$$

to which we associate the constants $\overline{C_1}$, $\overline{C_2}$, and $\overline{C_3}$ of type $o \multimap o \multimap o \multimap o$, respectively. We have $\mathrm{neg}(a_1) = C_3$, $\mathrm{neg}(a_2) = C_3$, $\mathrm{pos}(a_1) = \{C_1, C_2\}$, and $\mathrm{pos}(a_2) = \{C_2\}$. If $\psi_1(1) = 1$, $\psi_1(2) = 2$ and $\psi_2(1) = 2$, the terms R_{ij} are the following:

$$\begin{array}{lll} R_{11} = \overline{C_1}(a, a, a) & R_{12} = \overline{C_2}(a, a, a) & R_{13} = a \\ R_{21} = \overline{C_2}(a, a, a) & R_{22} = a & R_{23} = a \end{array}$$

Hence, the syntactic equation associated to this 1-neg-sat problem is as follows:

$$L_{\mathcal{C}} = \mathbf{X}(\overline{C_1}(\mathbf{X}_{11}, \mathbf{X}_{12}, \mathbf{X}_{13}), \overline{C_2}(\mathbf{X}_{21}, \mathbf{X}_{22}, \mathbf{X}_{23}), \overline{C_3}(\mathbf{X}_{31}, \mathbf{X}_{32}, \mathbf{X}_{33}))$$

$$R_{\mathcal{C}} = f(\overline{C_3}(\overline{C_1}(a, a, a), \overline{C_2}(a, a, a), a), \overline{C_3}(\overline{C_2}(a, a, a), a, a))$$

The intuition behind this reduction is the following. If the syntactic equation admits a solution, the term substituted for $\mathbf{X}$ must be of the form:

$$\lambda x_1 \ldots x_m . f(S_1, \ldots, S_n)$$

where each term S_i is either some λ-variable x_k, or some application of the form:

$$\overline{\mathrm{neg}(a_i)}(S_{i1}, \ldots, S_{im}).$$

The first case corresponds to a boolean variable a_i such that $\eta(a_i) = 0$, while the second case corresponds to a boolean variable a_i such that $\eta(a_i) = 1$.

Back to our example, one sees that the given equation admits the following solution:

$$\begin{cases} \mathbf{X} & := \lambda x_1\, x_2\, x_3.\, f(c_3(x_1, x_2, a),\, x_3) \\ \mathbf{X}_{31} & := c_2(a, a, a) \\ \mathbf{X}_{ij} & := a \text{ for } i \neq 3 \text{ or } j \neq 1 \end{cases}$$

which corresponds, indeed, to the only valuation that satisfies $\mathcal{C}$, namely, the valuation η such that $\eta(a_1) = 1$ and $\eta(a_2) = 0$.

5 Correctness of the reduction

Consider again a 1-neg-sat problem $\mathcal{C} = \{C_1, \ldots, C_m\}$ defined over the set of boolean variables $\mathcal{A} = \{a_1, \ldots, a_n\}$, and let the λ-terms R_{ij}, R_i, $L_{\mathcal{C}}$, and $R_{\mathcal{C}}$ be defined as in the previous section.

We first prove that the syntactic equation $\langle L_{\mathcal{C}}, R_{\mathcal{C}} \rangle$ admits a solution whenever $\mathcal{C}$ is satisfiable. To this end, suppose that $\mathcal{C}$ is satisfied by a valuation η.

Consequently, there exists a choice function ϕ that picks in each clause a literal that is satisfied by η. More precisely, we defined $\phi : \{1, \ldots, m\} \to \{1, \ldots, n\}$ to be a function such that

$$\text{either } \eta(a_{\phi(i)}) = 1 \text{ and } a_{\phi(i)} \in C_i, \text{ or } \eta(a_{\phi(i)}) = 0 \text{ and } \neg a_{\phi(i)} \in C_i.$$

Remark that this function is such that if $\phi(i) = \phi(j)$ and $\eta(a_{\phi(i)}) = 0$ then $i = j$. This is due to the constraint that a negative literal occurs in only one clause.

Given $\{x_1, \ldots, x_m\}$ a set of λ-variables, we define the family of terms S_i, for $i \in \{1, \ldots, n\}$, as follows:

$$S_i := \begin{cases} x_j \;\text{ if } \eta(a_i) = 0 \text{ and } j \text{ such that } \phi(j) = i \text{ exists} \\ \overline{\text{neg}(a_i)}(S_{i1}, \ldots, S_{im}) \;\text{ otherwise} \end{cases}$$

where, in the second case, the family of terms S_{ij} is the following:

$$S_{ij} := \begin{cases} x_k \;\text{ if } \eta(a_i) = 1 \text{ and } k \text{ such that } \phi(k) = i \text{ and } R_{ij} = \overline{C_k}(a, \ldots, a) \text{ exists} \\ R_{ij} \;\text{ otherwise} \end{cases}$$

Finally, we define:

$$S = \lambda x_1 \ldots x_m . f(S_1, \ldots, S_n)$$

As we will show, the above term is the main ingredient of a solution to the syntactic equation $\langle L_{\mathcal{C}}, R_{\mathcal{C}} \rangle$. In order to establish this fact, we first prove that S is indeed a λ-term of the linear λ-calculus.

Lemma 2. *For all $j \in \{1, \ldots, m\}$, $x_j \in \text{FV}(S_{\phi(j)})$.*

Proof. We proceed by case analysis, according to the value of $\eta(a_{\phi(j)})$.

Suppose that $\eta(a_{\phi(j)}) = 0$. Then, by definition, we have that $S_{\phi(j)} = x_j$. Hence, $x_j \in \text{FV}(S_{\phi(j)})$.

On the other hand, when $\eta(a_{\phi(j)}) = 1$, we have, by definition of ϕ, that $C_j \in \text{pos}(a_{\phi(j)})$. Consequently, there exists k such that $R_{\phi(j)k} = \overline{C_j}(a, \ldots, a)$. Therefore, by definition, $S_{\phi(j)k} = x_j$, which implies $x_j \in \text{FV}(S_{\phi(j)})$. □

Lemma 3. *If $x_j \in \text{FV}(S_i)$ then $\phi(j) = i$, for any $j \in \{1, \ldots, m\}$ and any $i \in \{1, \ldots, n\}$.*

Proof. An immediate consequence of the definition of the family of terms S_j. □

Lemma 4. *For all $i \in \{1, \ldots, n\}$ and all $C_k \in \text{pos}(a_i)$, there exists exactly one $j \in \{1, \ldots, m\}$ such that $R_{ij} = \overline{C_k}(a, \ldots, a)$.*

Proof. An immediate consequence of the definition of the family of terms R_{ij}. □

Lemma 5. *$S = \lambda x_1 \ldots x_m . f(S_1, \ldots, S_n)$ is a term of the linear λ-calculus.*

Proof. We have to prove that each of the λ-variables $x_1, \ldots, x_m$ has exactly one occurrence in $f(S_1, \ldots, S_n)$. By Lemma 2, we know that $x_1, \ldots, x_m \in \mathrm{FV}(f(S_1, \ldots, S_n))$. Hence, it remains to show that for any $j \in \{1, \ldots, m\}$, x_j occurs at most once in $f(S_1, \ldots, S_n)$. By Lemma 3, this amounts to prove that for any $i \in \{1, \ldots, n\}$, x_j occurs at most once in S_i. So, suppose that $x_j \in \mathrm{FV}(S_i)$. Then, either $x_j = S_i$, or $x_j = S_{ik}$ for some k. In the second case, k is such that $R_{ik} = \overline{C_j}(a, \ldots, a)$. Hence, it is unique by Lemma 4. Therefore, in both cases, there is only one occurence of x_j in S_i. □

It appears in the proof of Lemma 2 that for all $i \in \{1, \ldots, m\}$ either there exists $k \in \{1, \ldots, n\}$ such that $x_i = S_k$, or there exists $k \in \{1, \ldots, n\}$ and $l \in \{1, \ldots, m\}$ such that $x_i = S_{kl}$. This fact allows the family of terms T_{ij} (for $i, j \in \{1, \ldots, m\}$) to be defined as follows:

$$T_{ij} = \begin{cases} R_{kj} \text{ if } k \text{ such that } x_i = S_k \text{ exists} \\ a \text{ if } k \text{ and } l \text{ such that } x_i = S_{kl} \text{ exist} \end{cases}$$

It is immediate that these terms are terms of the linear λ-calculus.

We are now in a position of establishing that the syntactic equation $\langle L_{\mathcal{C}}, R_{\mathcal{C}} \rangle$ admits a solution provided that $\mathcal{C}$ is satisfiable.

Proposition 1. *Let $\mathcal{C}$ be a 1-neg-sat problem, and $\langle L_{\mathcal{C}}, R_{\mathcal{C}} \rangle$ be the associated syntactic equation. If $\mathcal{C}$ is satisfiable, then $\langle L_{\mathcal{C}}, R_{\mathcal{C}} \rangle$ admits a solution.*

Proof. The fact that $\mathcal{C}$ is satisfiable allows the terms S, and T_{ij} to be defined, and we prove that

$$L_{\mathcal{C}}[\mathbf{X} := S][\mathbf{X}_{ij} := T_{ij}]_{i=1}^{m\ m}_{j=1} \quad \twoheadrightarrow_\beta \quad R_{\mathcal{C}}$$

Indeed, we have:

$$\begin{aligned} L_{\mathcal{C}}[\mathbf{X} := S][\mathbf{X}_{ij} := T_{ij}]_{i=1}^{m\ m}_{j=1} & \\ &= S(\overline{C_1}(T_{11}, \ldots, T_{1m}), \ldots \overline{C_m}(T_{m1}, \ldots, T_{mm})) \\ &\twoheadrightarrow_\beta f(S_1, \ldots, S_n)[x_j := \overline{C_j}(T_{j1}, \ldots, T_{jm})]_{j=1}^{m} \end{aligned}$$

Then, it remains to show that for all $i \in \{1, \ldots, n\}$:

$$S_i[x_j := \overline{C_j}(T_{j1}, \ldots, T_{jm})]_{j=1}^{m} = R_i$$

There are two cases:

1. $S_i = x_k$, for some $k \in \{1, \ldots, m\}$.

   In this case, we have that $T_{kl} = R_{il}$, for all $l \in \{1, \ldots, m\}$. We also have $\eta(a_i) = 0$ and $\phi(k) = i$, which implies that $\mathrm{neg}(a_i) = C_k$. Consequently:

$$\begin{aligned} x_k[x_j := \overline{C_j}(T_{j1}, \ldots, T_{jm})]_{j=1}^{m} &= x_k[x_k := \overline{C_k}(R_{i1}, \ldots, R_{im})] \\ &= \overline{C_k}(R_{i1}, \ldots, R_{im}) \\ &= \overline{\mathrm{neg}(a_i)}(R_{i1}, \ldots, R_{im}) \\ &= R_i \end{aligned}$$

2. $S_i = \overline{\mathrm{neg}(a_i)}(S_{i1}, \ldots, S_{im})$.

   In this case, it is sufficient to show that for all $k \in \{1, \ldots, m\}$:

$$S_{ik}[x_j := \overline{C_j}(T_{j1}, \ldots, T_{jm})]_{j=1}^{m} = R_{ik}$$

   There are two subcases. In the case $S_{ik} = x_l$, for some $l \in \{1, \ldots, m\}$, we have that $R_{ik} = \overline{C_l}(a, \ldots, a)$ and $T_{kj} = a$, for all $j \in \{1, \ldots, m\}$. Therefore:

$$\begin{aligned} x_l[x_j := \overline{C_j}(T_{j1}, \ldots, T_{jm})]_{j=1}^{m} &= x_l[x_l := \overline{C_l}(a, \ldots, a)] \\ &= \overline{C_l}(a, \ldots, a) \\ &= R_{ik} \end{aligned}$$

   Otherwise, we have $S_{ik} = R_{ik}$, and the desired property follows immediately. □

It remains to prove that $\mathcal{C}$ is satisfiable whenever $\langle L_{\mathcal{C}}, R_{\mathcal{C}} \rangle$ admits a solution. We first establish a technical lemma concerning the form of the possible solutions of $\langle L_{\mathcal{C}}, R_{\mathcal{C}} \rangle$.

Lemma 6. *If the equation $\langle L_{\mathcal{C}}, R_{\mathcal{C}} \rangle$ admits a solution then the variable $\mathbf{X}$ is substituted by a term of the form*

$$\lambda x_1 \ldots x_m . f(U_1, \ldots, U_n)$$

where the terms U_i are such that:

1. *either $U_i = x_k$ (for some $k \in \{1, \ldots, m\}$), in which case $neg(a_i) = C_k$,*
2. *or $U_i = \overline{\mathrm{neg}(a_i)}(U_{i1}, \ldots, U_{im})$ where the terms U_{ij} are such that:*
   (a) *either $U_{ij} = x_k$ (for some $k \in \{1, \ldots, m\}$), in which case $C_k \in \mathrm{pos}(a_i)$,*
   (b) *or $U_{ij} = R_{ij}$.*

Proof. Suppose that

$$\begin{cases} \mathbf{X} &= U \\ \mathbf{X}_{ij} &= V_{ij} \end{cases}$$

is a solution to the syntactic equation $\langle L_{\mathcal{C}}, R_{\mathcal{C}} \rangle$. Then, we must have:

$$U(\overline{C_1}(V_{11}, \ldots, V_{1m}), \ldots \overline{C_m}(V_{m1}, \ldots, V_{mm})) \quad \twoheadrightarrow_\beta \quad f(R_1, \ldots, R_n)$$

This implies that U is indeed of the form

$$\lambda x_1 \ldots x_m . f(U_1, \ldots, U_n)$$

where for all $i \in \{1, \ldots, n\}$:

$$U_i[x_j := \overline{C_j}(V_{j1}, \ldots, V_{jm})]_{j=1}^{m} \quad \twoheadrightarrow_\beta \quad R_i$$

Now, the head of each U_i is either some λ-variable x_k or some constant. In the first case, $U_i = x_k$, and we must have that

$$\overline{C_k}(V_{k1}, \ldots, V_{km}), \quad = \quad R_i$$

which implies that $\mathrm{neg}(a_i) = C_k$. In the second case, the head of U_i must be the head of R_i, which implies that U_i is of the form

$$\overline{\mathrm{neg}(a_i)}(U_{i1}, \ldots, U_{im})$$

Moreover, we must have that

$$U_{ik}[x_j := \overline{C_j}(V_{j1}, \ldots, V_{jm})]_{j=1}^{m} \quad \twoheadrightarrow_\beta \quad R_{ik}$$

Now, if the head of U_{ik} is some λ-variable x_l, we must have $U_{ik} = x_l$, and:

$$\overline{C_l}(V_{l1}, \ldots, V_{lm}) = R_{ik}$$

This implies that $C_l \in \mathrm{pos}(a_i)$. Otherwise, we have

$$U_{ik} = R_{ik}.$$

□

We are now in a position of proving the second half of our reduction result.

Proposition 2. *Let $\mathcal{C}$ be a 1-neg-sat problem, and $\langle L_\mathcal{C}, R_\mathcal{C} \rangle$ be the associated syntactic equation. If $\langle L_\mathcal{C}, R_\mathcal{C} \rangle$ admits a solution, then $\mathcal{C}$ is satisfiable.*

Proof. According to Lemma 6, if $\langle L_\mathcal{C}, R_\mathcal{C} \rangle$ admits a solution, then te term U substituted for $\mathbf{X}$ is of the form

$$\lambda x_1 \ldots x_m . f(U_1, \ldots, U_n)$$

where:

1. either $U_i = x_k$, for some $k \in \{1, \ldots, m\}$,
2. or $U_i = \overline{\mathrm{neg}(a_i)}(U_{i1}, \ldots, U_{im})$.

We define a valuation η as follows:

$$\eta(a_i) = \begin{cases} 0 \text{ if } U_i = x_j \text{ for some } j \in \{1, \ldots, m\} \\ 1 \text{ otherwise} \end{cases}$$

Now, for every clause C_j such that $x_j = U_i$, for some $i \in \{1, \ldots, n\}$, we have, by Lemma 6, that $\mathrm{neg}(a_i) = C_j$, i.e., $\neg a_i \in C_j$. Consequently, these clauses are satisfied by η.

As for the other clauses C_k, since U is a term of the linear λ-calculus, there must exists some term U_{ij} such that $U_{ij} = x_k$. In this case, according to Lemma 6, $C_k \in \mathrm{pos}(a_i)$, i.e., $a_i \in C_k$. Consequently, these clauses are also satisfied by η. □

As a consequence of Lemma 1, and Propositions 1, and 2, we get the main theorem of this paper.

Theorem 1. *Linear second-order matching in the linear λ-calculus is NP-complete.*

6 Related results

The main difference between a context and a linear second-order λ-term is that the latter has the ability of rearranging its arguments in any order. This explains why linear second-order matching in the linear λ-calculus is NP-complete while linear context matching is not. Nevertheless, this difference is not significant when the arguments of the second-order meta-variable do not contain any meta-variable (first-order or second-order). Consider a second-order equation of the form:

$$\mathbf{X}(T_1, \ldots, T_n) = T$$

where $T_1, \ldots, T_n$, and T are first-order pure linear λ-terms (such an equation is called an interpolation equation). It is not difficult to see that it may be solved in polynomial time. Indeed, it amounts to check whether the union of the multisets of the subterms of $T_1, \ldots, T_n$ is included in the multiset of the subterms of T.

This polynomiality result, which is quite specific, cannot be generalized. Indeed, as we prove in the next proposition, interpolation in third-order case is NP-complete.

Proposition 3. *Third-order interpolation in the linear λ-calculus is NP-complete.*

Proof. The proof consists in a reduction of 1-Neg-sat that we obtain by reducing the equation $\langle L_{\mathcal{C}}, R_{\mathcal{C}} \rangle$ (as defined in section 4) to a third-order interpolation equation

Let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be a 1-neg-sat problem defined over the set of boolean variables $\mathcal{A} = \{a_1, \ldots, a_n\}$. We build a third-order interpolation equation $\langle L, R \rangle$ which has a solution if and only if the equation $\langle L_{\mathcal{C}}, R_{\mathcal{C}} \rangle$ has a solution. From Propositions 1 and 2, this is equivalent to say that $\langle L, R \rangle$ has a solution if and only if $\mathcal{C}$ is satisfiable. Therefore, from Lemma 1, we obtain that third-order interpolation is NP-complete problem.

We first define

$$\begin{cases} L = Y(\lambda x_1 \ldots x_m.\overline{C_1}(x_1, \ldots, x_m), \ldots, \lambda x_1 \ldots x_m.\overline{C_m}(x_1, \ldots, x_m)) \\ R = R_{\mathcal{C}} \end{cases}$$

Then it remains to prove that $\langle L, R_{\mathcal{C}} \rangle$ has a solution if and only if $\langle L_{\mathcal{C}}, R_{\mathcal{C}} \rangle$ has a solution.

Suppose $\langle L_{\mathcal{C}}, R_{\mathcal{C}} \rangle$ has a solution :

$$\begin{cases} \mathbf{X} &= U \\ \mathbf{X}_{ij} &= V_{ij} \end{cases}$$

then the term

$$S = \lambda y_1 \ldots y_m.U(y_1(V_{11}, \ldots, V_{1m}), \ldots, y_m(V_{m1}, \ldots, V_{mm}))$$

is a solution of $\langle L, R_{\mathcal{C}} \rangle$. Indeed:

$$\begin{array}{ll} S(\lambda x_1 \ldots x_m.C_1(x_1, \ldots, x_m), \ldots, \lambda x_1 \ldots x_m.C_m(x_1, \ldots, x_m)) & \twoheadrightarrow_\beta \\ U(C_1(V_{11}, \ldots, V_{1m}), \ldots, C_m(V_{m1}, \ldots, V_{mm})) & \twoheadrightarrow_\beta R_{\mathcal{C}} \end{array}$$

Conversely if $\langle L, R_{\mathcal{C}} \rangle\ [Y := S]$, then $S = \lambda y_1 \dots y_m.S'$ and one can find terms of linear λ-calculus U, V_{11}, $\dots$, V_{1m},$\dots$,V_{m1}, $\dots$, V_{mm} such that:

$$U(y_1(V_{11}, \dots, V_{1m}), \dots, y_m(V_{m1}, \dots, V_{mm})) \twoheadrightarrow_\beta S'$$

and then

$$\begin{cases} \mathbf{X} &= U \\ \mathbf{X}_{ij} &= V_{ij} \end{cases}$$

is obviously a solution of $\langle L_{\mathcal{C}}, R_{\mathcal{C}} \rangle$. □

References

1. H. Comon. Completion of rewrite systems with membership constraints. Part I: Deduction rules. *Journal of Symbolic Computation*, 25(4):397–419, 1998.
2. H. Comon. Completion of rewrite systems with membership constraints. Part II: Constraint solving. *Journal of Symbolic Computation*, 25(4):421–453, 1998.
3. S. A. Cook. The complexity of theorem proving procedures. *Proceedings of the 3rd annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.
4. D. Dougherty and T. Wierzbicki. A decidable variant of higher order matching. In *Proc. 13th Conf. on Rewriting Techniques and Applications, RTA'02*, volume 2378, pages 340–351, 2002.
5. G. Dowek. Third order matching is decidable. *Annals of Pure and Applied Logic*, 69(2–3):135–155, 1994.
6. G. Dowek. Higher-order unification and matching. In A. Robinson and A. Voronkov (eds.), *Handbook of Automated Reasoning*, pp. 1009-1062, Elsevier, 2001.
7. W. D. Goldfarb. The undecidability of the second-order unification problem. *Theoretical Computer Science*, 13(2):225–230, 1981.
8. Ph. de Groote. Higher-order linear matching is np-complete. *Lecture Notes in Computer Science*, 1833:127–140, 2000.
9. G. Huet. The undecidability of unification in third order logic. *Information and Control*, 22(3):257–267, 1973.
10. G. Huet. *Résolution d'équations dans les langages d'ordre* $1, 2, \dots, \omega$. Thèse de Doctorat d'Etat, Université Paris 7, 1976.
11. J. Levy. Linear second-order unification. In H. Ganzinger, editor, *Rewriting Techniques and Applications, RTA'96*, volume 1103 of *Lecture Notes in Computer Science*, pages 332–346. Springer-Verlag, 1996.
12. R. Loader. Higher order β matching is undecidable. *Logic Journal of the IGPL*, 11(1): 51–68, 2002.
13. V. Padovani. *Filtrage d'ordre supérieure.* Thèse de Doctorat, Université de Paris 7, 1996.
14. P. Kilpeläinen. Ordered and unordered tree inclusion. *SIAM. J. Comput.*, 24(2): 340–356, 1995.
15. M. Schmidt-Schauß and J. Stuber. On the complexity of linear and stratified context matching problems. Rapport de recherche A01-R-411, LORIA, December 2001.