# Third-Order Matching in the Polymorphic Lambda Calculus

Jan Springintveld*
*Department of Philosophy*
*Utrecht University*
*P.O. Box 80126, 3508 TC Utrecht, The Netherlands*

**Abstract**

We show that it is decidable whether a third-order matching problem in the polymorphic lambda calculus has a solution. The proof is constructive in the sense that an algorithm can be extracted from it that, given such a problem, returns a substitution if it has a solution and *fail* otherwise.

## 1 Introduction

This paper is a contribution to the theory of (pattern) matching in higher order type theory. The starting point is the fact that third-order matching is decidable in the simply typed lambda calculus with constant types (see [5]). The question we would like to answer is: what happens if we extend this calculus with the type features that are characteristic for the Calculus of Constructions [2]: dependent types, type constructors and polymorphism. In [3], Dowek showed that in lambda calculi with dependent types third-order matching is undecidable. In contrast, we showed in [15] that the presence of type constructors is not sufficient to make third-order matching undecidable. In this paper we show that in the polymorphic lambda calculus third-order matching is decidable.

Of course the last two results leave many questions unanswered. They leave open the relation between matching of finite order $n \geq 4$ in the simply typed lambda calculus (decidable by [13], [14]) and matching of order $n \geq 4$ in the extended systems. This is an option for future research. (In calculi with both type constructors and polymorphism, fourth-order matching is undecidable; see [3]. Matching of infinite order is undecidable in the polymorphic lambda calculus; see [7].)

The proofs in [15] and in this paper both consist of a reduction to third-order matching in the simply typed lambda calculus. It is interesting to see what the difficulties are in the two proofs of decidability. In the first system the problem is that, because of the presence of type constructors, the type structure is rich enough to encode second-order unification for simply typed terms (undecidable by [12]) as third-order unification for types. Naive algorithms quickly run into the situation where such unification problems have to be solved. So the task is to devise an algorithm that avoids them. In the second system the types can be considered

*This work is supported by the Netherlands Computer Science Research Foundation (SION) with financial support of the Netherlands Organisation for Scientific Research (NWO).

constant but the terms have a complicated $\beta$-reduction behaviour. This is important because an essential ingredient of the reduction will be a translation that preserves $\beta$-reduction. For instance, the term $(\lambda\alpha{:}*.x\alpha(\lambda y{:}\alpha.y))(C{\to}D)$ $\beta$-reduces in one step to $x(C{\to}D)(\lambda y{:}C{\to}D.y)$. Notice that the type $C{\to}D$ is substituted for $\alpha$ in $\lambda y{:}\alpha.y$. It is very difficult to represent such $\beta$-reduction steps as $\beta$-reduction steps of simply typed terms. A large part of this paper is devoted to solving this problem.

This paper is organized as follows. In Section 2, we present the two typed lambda calculi with which we shall be concerned in this paper: $\lambda\tau$ (the simply typed lambda calculus with one base type $O$) and $\lambda 2$, the polymorphic lambda calculus. In Section 3, we present terminology concerning matching problems and solutions. In Section 4, we discuss in more detail the difficulties in proving decidability of third-order matching in $\lambda 2$. As mentioned above, the proof of decidability is a reduction of third-order matching in $\lambda 2$ to third-order matching in $\lambda\tau$. Because of the difficulties sketched in Section 4, the reduction consists of two steps. First, we show how to reduce the problem whether a third-order matching problem in $\lambda 2$ has a solution to the problem whether a third-order matching problem in $\lambda 2$ in a certain simplified format, which we call *restricted*, has a solution. Second, we translate restricted third-order matching problems in $\lambda 2$ to third-order matching problems in $\lambda\tau$ in such a way that the original problem has a solution iff the translated problem has a solution. Decidability of third-order matching in $\lambda 2$ follows quickly. Before embarking on the outlined proof, we define a range of useful concepts in Section 5, culminating in the notion "restricted". The reduction of the general case to the restricted case is done in Section 6. In Section 7 we show that for restricted third-order matching problems the notion of a solution for a matching problem can be simplified. This leads to smoother proofs in the subsequent sections. The encoding of restricted third-order matching problems (and their solutions) is the subject of Section 8. Decidability of restricted third-order matching (hence of third-order matching in general) is proved in Section 9.

**Acknowledgements.** I would like to thank Gilles Dowek, Alex Sellink and Erik Barendsen. Discussions with Gilles Dowek led to some of the ideas in this paper. Discussions with Alex Sellink were helpful to clarify some notions and issues. Gilles Dowek and Erik Barendsen were helpful in tracking down references.

## 2 The systems $\lambda\tau$ and $\lambda 2$

In this section we introduce two typed lambda calculi: $\lambda\tau$, the simply typed lambda calculus with one base type $O$, and $\lambda 2$, the polymorphic lambda calculus. These systems are presented in the PTS format (see [1] or [11]).

**Definition 2.1** *(Terms and reductions). Pseudo-terms* are given by the following abstract syntax:

$$\mathcal{T} ::= \mathcal{C} \mid \mathcal{V} \mid \mathcal{T}\mathcal{T} \mid \lambda\mathcal{V}{:}\mathcal{T}.\mathcal{T} \mid \Pi\mathcal{V}{:}\mathcal{T}.\mathcal{T}.$$

Here $\mathcal{C}$ is an infinite set of constants and $\mathcal{V}$ is an infinite set of variables; $x$, $y$, $y'$, $y_1$, ... range over $\mathcal{V}$. Among the constants, three elements are singled out: $O$, $*$ and $\Box$. Roman letters range over $\mathcal{T}$. When $x \notin FV(A_2)$, we write $\Pi x{:}A_1.A_2$ as $A_1{\to}A_2$. We define $\mathcal{K}_* ::= O \mid$

$\mathcal{K}_* \to \mathcal{K}_*$. We apply the usual conventions concerning brackets; so $ABC$ means $(AB)C$ and $A \to B \to C$ means $A \to (B \to C)$. An *abstraction term* is a term of the form $\lambda x{:}A_1.A_2$. When $A$ contains a subterm of the form $\lambda x{:}A_1.A_2$ then we say that $A_1$ is a *domain* in $A$. An *application term* is a term of the form $A_1A_2$. Using the brackets convention we can write every application term as $A_1 \ldots A_n$ (for some $n \geq 2$) with $A_1$ not an application term. The set of free variables of $A$ is defined as usual and denoted by $FV(A)$. Also the substitution of $A$ for $x$ in $B$ (denoted by $B[x := A]$) and the relations $\to_\beta$, $\twoheadrightarrow_\beta$, $\to_\eta$, $\twoheadrightarrow_\eta$, $\twoheadrightarrow_{\beta\eta}$, $=_\beta$ and $=_{\beta\eta}$ are defined on pseudo-terms as usual. Syntactic equality (modulo $\alpha$-conversion) is denoted by $\equiv$.

**Definition 2.2** *(Contexts and Judgements).* In this paper $Q$, $Q_1$, ... range over $\{\exists, \forall\}$. $Qx : B$ is called a *(quantified) declaration.* A *(quantified) pseudo-context* is a finite ordered sequence of quantified declarations $Q_i x_i : C_i$, where the $x_i$ are pairwise distinct. Pseudo-contexts are denoted by capital Greek letters $\Delta$, $\Gamma$, $\Gamma_0$, ... The empty context is denoted by $\langle\rangle$.

If $\exists x : C$ occurs in $\Gamma$, then $x$ is said to be *existential in* $\Gamma$. If $\forall x : C$ occurs in $\Gamma$, then $x$ is said to be *universal in* $\Gamma$. If every declaration in $\Gamma$ is of the form $\exists x : C$, then $\Gamma$ is an *existential context.*

The intuition behind the quantification of variables is that universal variables are considered to be constant, in the sense that solutions to matching problems are not allowed to substitute terms for them; substitutions are only allowed to substitute terms for existential variables.

If $\Gamma \equiv \langle Q_1 x_1 : A_1, \ldots, Q_n x_n : A_n \rangle$, then $FV(\Gamma) := \{x_1, \ldots, x_n\} \cup \bigcup_{1 \leq i \leq n} FV(A_i)$, $dom(\Gamma) = \{x_1, \ldots, x_n\}$ and $\Gamma, Qx : B$ denotes $\langle Q_1 x_1 : A_1, \ldots, Q_n x_n : A_n, Qx : B \rangle$. (In general, we denote the concatenation of $\Gamma$ and $\Delta$ by $\Gamma, \Delta$.) Furthermore, for $1 \leq i \leq n$, $\Gamma_{x_i}$ denotes $\langle Q_1 x_1 : A_1, \ldots, Q_{i-1} x_{i-1} : A_{i-1} \rangle$ and $Ex(\Gamma)$ denotes $\langle \exists x_1 : A_1, \ldots, \exists x_n : A_n \rangle$. If the declaration $Qx : C$ occurs in $\Gamma$, then $\Gamma(x)$ denotes $C$. Let $X \subseteq dom(\Gamma)$. Then $\Gamma \setminus X$ denotes the context which is the result of removing declarations $Qx : A$, for $x \in X$, from $\Gamma$. $\Gamma \sqcup \Delta$ denotes $\Gamma, (\Delta \setminus dom(\Gamma))$, the concatenation of $\Gamma$ and $\Delta$ without duplicates. We write $\Gamma \subseteq \Delta$ if each $Qx{:}A$ in $\Gamma$ also occurs in $\Delta$; we call such $\Delta$ an *extension of* $\Gamma$.

A *judgement* is of the form $\Gamma \vdash A : B$, where $\Gamma$ is a quantified pseudo-context and $A$ and $B$ are pseudo-terms. When we want to indicate that a judgement is derived in a system $\lambda\circ$, we write $\Gamma \vdash_{\lambda\circ} A : B$. If $\Gamma \vdash_{\lambda\circ} A : B$, then we call $A$ *closed in* $\Gamma$ if all free variables $x$ in $A$ are universal in $\Gamma$ and moreover $\Gamma(x)$ is closed in $\Gamma_x$ (if $A$ is not closed in $\Gamma$, then $A$ is called *open in* $\Gamma$). We use the abbreviation $\Gamma \vdash A : B : C$ for $\Gamma \vdash A : B$ and $\Gamma \vdash B : C$. A pseudo-term $A$ is called *legal* when there exist a pseudo-context $\Gamma$ and a pseudo-term $B$ such that $\Gamma \vdash A : B$ or $\Gamma \vdash B : A$. A pseudo-context $\Gamma$ is called legal when there exist pseudo-terms $A$ and $B$ such that $\Gamma \vdash A : B$.

**Definition 2.3** *(The systems).* The systems $\lambda\tau$ and $\lambda 2$ are defined using the rules in Table 1 and the specification $(\mathcal{S}, \mathcal{A}, \mathcal{R})$. Here $\mathcal{S}$ is a collection of *sorts*, $\mathcal{A}$ is a set of *axioms* of the form $c : s$ with $c \in \mathcal{C}$ and $s \in \mathcal{S}$ and $\mathcal{R}$ is a set of *rules* of the form $(s_1, s_2)$ with $s_1, s_2 \in \mathcal{S}$. In Table 1, $s$ ranges over $\mathcal{S}$. The system $\lambda\tau$ can be obtained by taking $\mathcal{S} = \{*\}$, $\mathcal{A} = \{O : *\}$ and $\mathcal{R} = \{(*, *)\}$. The system $\lambda 2$ can be obtained by taking $\mathcal{S} = \{*, \Box\}$, $\mathcal{A} = \{* : \Box\}$ and $\mathcal{R} = \{(*, *), (\Box, *)\}$. In this paper we let $\lambda\circ$ range over $\lambda\tau$ and $\lambda 2$ and (except in Table 1) $s$ over $\{*, \Box\}$.

| | | |
|---|---|---|
| Axiom | $\langle\rangle \vdash c : s$ | if $c : s \in \mathcal{A}$ |
| Start | $\dfrac{\Gamma \vdash B : s}{\Gamma, Qx{:}B \vdash x : B}$ | if $x \notin FV(\Gamma)$ |
| Weakening | $\dfrac{\Gamma \vdash A : B \quad \Gamma \vdash B' : s}{\Gamma, Qx{:}B' \vdash A : B}$ | if $x \notin FV(\Gamma)$ |
| Product | $\dfrac{\Gamma \vdash A_1 : s_1 \quad \Gamma, Qx{:}A_1 \vdash A_2 : s_2}{\Gamma \vdash \Pi x{:}A_1.A_2 : s_2}$ | if $(s_1, s_2) \in \mathcal{R}$ |
| Application | $\dfrac{\Gamma \vdash A_1 : \Pi x{:}B_1.B_2 \quad \Gamma \vdash A_2 : B_1}{\Gamma \vdash A_1 A_2 : B_2[x := A_2]}$ | |
| Abstraction | $\dfrac{\Gamma, Qx{:}A_1 \vdash A_2 : B_2 \quad \Gamma \vdash \Pi x{:}A_1.B_2 : s}{\Gamma \vdash \lambda x{:}A_1.A_2 : \Pi x{:}A_1.B_2}$ | |

Table 1: The rules

The following meta-theoretic properties hold in $\lambda\tau$ and $\lambda\underline{\omega}$. The results are taken from [1], [11], and [10].

**Theorem 2.4.**

1. (Substitutivity) *If $A \twoheadrightarrow_{\beta\eta} A'$, then $A[x := B] \twoheadrightarrow_{\beta\eta} A'[x := B]$.*

2. (Substitution Lemma) *Suppose $\Gamma, Qx : C, \Delta \vdash_{\lambda\circ} A : B$ and $\Gamma \vdash_{\lambda\circ} D : C$. Then $\Gamma, \Delta[x := D] \vdash_{\lambda\circ} A[x := D] : B[x := D]$.*

3. (Strong Normalization). *If $\Gamma \vdash_{\lambda\circ} A : B$, then there are no infinite $\beta\eta$-reduction sequences starting from $A$ or from $B$.*

4. (Confluence) *If $\Gamma \vdash_{\lambda\circ} A_1 : B$ and $\Gamma \vdash_{\lambda\circ} A_2 : B$ and $A_1 =_{\beta\eta} A_2$, then there exists a term $A_3$ such that $A_1 \twoheadrightarrow_{\beta\eta} A_3$ and $A_2 \twoheadrightarrow_{\beta\eta} A_3$.*

5. (Subject Reduction) *If $\Gamma \vdash_{\lambda\circ} A : B$ and $A \twoheadrightarrow_{\beta\eta} A'$, then $\Gamma \vdash_{\lambda\circ} A' : B$.*

6. (Unicity of Types) *If $\Gamma \vdash_{\lambda\circ} A : B$ and $\Gamma \vdash_{\lambda\circ} A : B'$, then $B \equiv B'$.*

7. (Thinning) *If $\Gamma \vdash_{\lambda\circ} A : B$, $\Gamma \subseteq \Delta$ and $\Delta$ is legal in $\lambda\circ$, then $\Delta \vdash_{\lambda\circ} A : B$.*

8. (Strengthening). *If $\Gamma, x : C, \Delta \vdash_{\lambda\circ} A : B$ and $x \notin FV(\Delta) \cup FV(A) \cup FV(B)$, then $\Gamma, \Delta \vdash_{\lambda\circ} A : B$.*

9. (Permutation) *If* $\Gamma, x : C, y : D, \Delta \vdash_{\lambda\circ} A : B$ *and* $\Gamma \vdash_{\lambda\circ} D : s$, *then* $\Gamma, y : D, x : C, \Delta \vdash_{\lambda\circ} A : B$.

By Confluence and Strong Normalization, each legal term $A$ has a unique $\beta\eta$-normal form. We denote it by $\mathrm{nf}(A)$. A term $A$ is called *normal* when $\mathrm{nf}(A) \equiv A$. By Unicity of Types we know that if a term $A$ has a type in a context $\Gamma$ then this type is unique. We denote it by $\tau(\Gamma\,;\,A)$.

Next, we give some more basic facts concerning the terms in our systems. The proofs of these facts are standard and omitted.

**Lemma 2.5.**

1. $\Gamma \vdash_{\lambda 2} A : \Box \Leftrightarrow (\Gamma$ *legal and* $A \equiv *)$.

2. $\Gamma \vdash_{\lambda\tau} A : s \Leftrightarrow (\Gamma$ *legal and* $s \equiv *$ *and* $A \in \mathcal{K}_*)$. *Such* $A$ *are called* $\lambda\tau$-types. $O$ *is an* atomic type, *the other types are* arrow types. *All* $\lambda\tau$*-types are* $\beta\eta$*-normal.*

3. $\Gamma \vdash_{\lambda\circ} A : s$, *then for all* $x \in dom(\Gamma)$ *such that* $\Gamma_x \vdash_{\lambda\circ} \Gamma(x) : *$ *we have:* $x \notin FV(A)$. *So* $A$ *contains no object variables.*

4. *If* $\Gamma \vdash_{\lambda 2} A : *$, *then* $A$ *is called a* $\lambda 2$-type *and* $A$ *is either a variable, declared in* $\Gamma$, *or of the form* $\Pi x{:}A_1.A_2$ *with* $\Gamma \vdash_{\lambda 2} A_1 : s$ *and* $\Gamma, Qx : A_1 \vdash_{\lambda 2} A_2 : *$. *In the first case,* $A$ *is called* atomic. *If* $x \in FV(A_2)$, *then* $A_1 \equiv *$. *We call a* $\lambda 2$*-type* polymorphic *when it has a subterm of the form* $\Pi x{:}A_1.A_2$, *with* $x \in FV(A_2)$. *All* $\lambda 2$*-types are* $\beta\eta$*-normal.*

5. *If* $\Gamma \vdash_{\lambda\tau} A : B : *$ *then* $A$ *is of one of the three following forms:*

   - $x$, *with* $Qx : B \in \Gamma$.
   - $A_1A_2$, *with, for some term* $B_1$, $\Gamma \vdash_{\lambda\tau} A_1 : B_1{\to}B$ *and* $\Gamma \vdash_{\lambda\tau} A_2 : B_1$.
   - $\lambda x{:}A_1.A_2$, *where* $A_1$ *is as described in (2) and* $\Gamma, Qx : A_1 \vdash_{\lambda\tau} A_2 : B_2 : *$, *for some term* $B_2$ *such that* $B \equiv A_1{\to}B_2$.

6. *If* $\Gamma \vdash_{\lambda 2} A : B : *$ *then* $A$ *is of one of the three following forms:*

   - $x$, *where* $Qx : B \in \Gamma$. *If* $B$ *is polymorphic then we say that* $x$ *is a* polymorphic variable (PV) in $\Gamma$.
   - $A_1A_2$, *with, for some terms* $B_1$, $B_2$, $\Gamma \vdash_{\lambda 2} A_1 : \Pi x{:}B_1.B_2$ *and* $\Gamma \vdash_{\lambda 2} A_2 : B_1$ *and* $B_2[x := A_2] \equiv B$.
   - $\lambda x{:}A_1.A_2$, *where* $A_1$ *is as described in (1) or (4) and* $\Gamma, Qx : A_1 \vdash_{\lambda 2} A_2 : B_2 : *$, *for some term* $B_2$ *such that* $B \equiv \Pi x{:}A_1.B_2$.

   *If* $A$ *is of the form* $xA_1 \ldots A_n$, *for some* $n \geq 0$ *and some PV* $x$ *in* $\Gamma$, *then we say that* $A$ starts with a PV.

7. *Suppose* $\Gamma \vdash_{\lambda\tau} A : *$. *Using the brackets convention we can write* $A$ *uniquely as* $A_1{\to}\cdots{\to}A_n{\to}B$, *with* $n \geq 0$ *and* $B \equiv O$. *Unless stated otherwise, we assume that* $\lambda\tau$*-types are written in this way.*

8. *Suppose* $\Gamma \vdash_{\lambda\tau} A : *$. *We can write* $A$ *uniquely as* $\Pi x_1{:}A_1 \ldots \Pi x_n{:}A_n.B$, *with* $n \geq 0$ *and* $B$ *a variable. Unless stated otherwise, we assume that* $\lambda 2$*-types are written in this way.*

9. *Suppose* $\Gamma \vdash_{\lambda\circ} A : B$. *There exist* $\Gamma_1, \Gamma_2 \subseteq \Gamma$ *such that* $\Gamma_1, \Gamma_2 \vdash_{\lambda\circ} A : B$ *and for all declarations* $x : C$ *in* $\Gamma$ *we have:* $Qx : C \in \Gamma_1 \Leftrightarrow C \equiv *$. $\Gamma_1$ *and* $\Gamma_2$ *are unique up to the order of the declarations. The variables in* $\Gamma_1$ *are called* type variables*; the variables in* $\Gamma_2$ *are called* object variables. From now on we assume that contexts are in this form.

10. *Suppose* $\Gamma \vdash_{\lambda\circ} A : \Pi x_1{:}B_1 \ldots \Pi x_n{:}B_n.C : s$, *with* $A$ *in* $\beta\eta$*-normal form. Then* $A \equiv \lambda x_1{:}B_1 \ldots \lambda x_m{:}B_m.xC_1 \ldots C_k$, *for some* $0 \leq m \leq n$, *some variable* $x$ *(possibly among* $\{x_1, \ldots, x_m\}$*) and* $\beta\eta$*-normal terms* $C_1, \ldots, C_k$ *(for some* $k \geq 0$*).*

It will sometimes be convenient to assume that terms are in $\eta$-long-$\beta$-normal form. This notion is defined below.

**Definition 2.6.** Suppose $\Gamma \vdash_{\lambda\circ} A : B$, where $A$ is $\beta\eta$-normal. We define $\mathrm{lnf}_\Gamma(A)$ by induction on the pair (length of $A$, length of $B$), ordered lexicographically. By Lemma 2.5, it suffices to consider the following cases.

- $A \in \mathcal{C}$. Then $\mathrm{lnf}_\Gamma(A) \equiv A$.
- $A \equiv \lambda x{:}A_1.A_2$. Then $\mathrm{lnf}_\Gamma(A) \equiv \lambda x{:}\mathrm{lnf}_\Gamma(A_1).\mathrm{lnf}_{\Gamma,\forall x:A_1}(A_2)$.
- $A \equiv \Pi x{:}A_1.A_2$. Then $\mathrm{lnf}_\Gamma(A) \equiv A$.
- $A \equiv xA_1 \ldots A_n$. Write $B$ as $\Pi y_1{:}B_1 \ldots \Pi y_m{:}B_m.C$. Then

  $$\mathrm{lnf}_\Gamma(A) \equiv \lambda y_1{:}B_1 \ldots \lambda y_m{:}B_m.xA'_1 \ldots A'_n y'_1 \ldots y'_m,$$

  where $A'_i \equiv \mathrm{lnf}_\Gamma(A_i)$ (for $1 \leq i \leq n$) and $y'_i \equiv \mathrm{lnf}_{\Gamma,\forall y_i:B_i}(y_i)$ (for $1 \leq i \leq m$).

  The recursive calls on the variables $y_i$ are justified as follows. If $n > 0$ then the length of $y_i$ is less than the length of $A$. If $n = 0$ then the length of $y_i$ equals the length of $A$ but the length of the type of $y_i$ is less than the length of $B$.

This definition is extended to non-normal terms as follows. Suppose $\Gamma \vdash_{\lambda\circ} A : B$. Then $\mathrm{lnf}_\Gamma(A) := \mathrm{lnf}_{\mathrm{nf}(\Gamma)}(\mathrm{nf}(A))$. This definition is justified by Theorem 2.4. If $\Gamma \vdash_{\lambda\circ} A : B$, then $A$ is said to be *in LNF* if $\mathrm{lnf}_\Gamma(A) \equiv A$.

Note that if $\Gamma \vdash_{\lambda\circ} A : s$ then $A$ is in LNF.

**Lemma 2.7.** *Suppose* $\Gamma \vdash_{\lambda\circ} A : \Pi x_1{:}A_1 \ldots \Pi x_n{:}A_n.C : *$, *where all terms are in* LNF. *Then* $A \equiv \lambda x_1{:}A_1 \ldots \lambda x_n{:}A_n.xB_1 \ldots B_m$, *for some variable* $x$ *(possibly among* $\{x_1, \ldots, x_n\}$*) and terms* $B_1, \ldots, B_m$ *(in* LNF*).*

In the following definition we give some concepts to speak of the type of a subterm of a term $A$, subterms of $A$ that are types, etc.

**Definition 2.8.** Let $\Gamma \vdash_{\lambda 2} A : B$. By induction on the structure of $A$, we define a tree, *Tree*$(\Gamma\,;\,A)$, whose nodes are labeled with pairs $(\Delta\,;\,C)$, where $\Delta$ is an extension of $\Gamma$ and $C$ a subterm of $A$ such that $\tau(\Delta\,;\,C)$ exists.

- For $c \in \mathcal{C}$, *Tree*$(\Gamma\,;\,c)$ is a node labeled with $(\Gamma\,;\,c)$.
- *Tree*$(\Gamma\,;\,x)$ is a node labeled with $(\Gamma\,;\,x)$.
- *Tree*$(\Gamma\,;\,A_1A_2)$ consists of a root labeled with $(\Gamma\,;\,A_1A_2)$ and subtrees *Tree*$(\Gamma\,;\,A_1)$ and *Tree*$(\Gamma\,;\,A_2)$.
- *Tree*$(\Gamma\,;\,\lambda x{:}A_1.A_2)$ is a root labeled with $(\Gamma\,;\,\lambda x{:}A_1.A_2)$ and subtrees *Tree*$(\Gamma\,;\,A_1)$ and *Tree*$(\Gamma, \forall x : A_1\,;\,A_2)$. Here, $x$ is a fresh variable.
- *Tree*$(\Gamma\,;\,\Pi x{:}A_1.A_2)$ is a root labeled with $(\Gamma\,;\,\Pi x{:}A_1.A_2)$ and subtrees *Tree*$(\Gamma\,;\,A_1)$ and *Tree*$(\Gamma, \forall x : A_1\,;\,A_2)$. Here, $x$ is a fresh variable.

It follows from Lemma 2.5 that this is well-defined. Note that in the $\lambda$- and the $\Pi$-case, a fresh variable is added to the context. So *Tree*$(\Gamma\,;\,A)$ is defined up to the choice of such fresh variables. Note that for all labels $(\Delta\,;\,C)$ in *Tree*$(\Gamma\,;\,A)$ we have indeed that, for some term $D$, $\Delta \vdash_{\lambda\circ} C : D$. Also note that the tree structure is not really necessary but is quite natural.

Let $(\Delta\,;\,C)$ be a label in *Tree*$(\Gamma\,;\,A)$. Then $C$ is called a *subterm of* $A$; when we consider such a subterm, $\Delta$ is called its *current context*. (Note that $C$ is also a subterm of $A$ in the traditional sense). When we speak of the *type of* $C$ then we mean $\tau(\Delta\,;\,C)$. If $\Delta \vdash_{\lambda 2} C : *$, then $C$ is called a *type in* $A$. A *polymorphic argument in* $A$ is a type in $A$ which is not in a domain in $A$. To see why we use this phrase, check that such terms always occur as a term $A_i$ in a context $A_1 \ldots A_{i-1}A_iA_{i+1} \ldots A_n$ with $i \geq 2$. Sometimes we call $A_i$ a polymorphic argument *of* $A_1$. A *free type in* $A$ is a type $C$ in $A$ such that $\Gamma \vdash_{\lambda 2} C : *$. (So $C$ does not contain variables bound outside $C$.)

In several proofs we will underline some occurrences of terms in $A$. To each such underlined term naturally corresponds a right projection of a label of a node in *Tree*$(\Gamma\,;\,A)$. *Par abus de langage* we will call a term that can be underlined in $A$ a type if it corresponds in this sense to a type in *Tree*$(\Gamma\,;\,A)$ (similarly for 'polymorphic argument').

# 3 Matching problems

We present the notions of a *matching problem* and a *solution* for such a matching problem along the lines of [4], [6].

**Definition 3.1.** Suppose $\Gamma \vdash_{\lambda\circ} A : s$. We define $ord_\Gamma(A)$, the *order of* $A$ *in* $\Gamma$ as follows.

$$ord_\Gamma(A) = \begin{cases} 2 & \text{if } A \equiv * \\ 1 & \text{if } A \equiv O \\ 1 & \text{if } A \equiv x \text{ and } x \text{ universal in } \Gamma \\ \infty & \text{if } A \equiv x \text{ and } x \text{ existential in } \Gamma \\ \max(\{1 + ord_\Gamma(A_1), ord_{\Gamma,\exists x:A_1}(A_2)\}) & \text{if } A \equiv \Pi x{:}A_1.A_2 \end{cases}$$

The definition by cases is O.K. by Lemma 2.5. By convention, $\max(\{n, \infty\}) = \infty$ and $n + \infty = \infty$. When $\Gamma$ is clear from the context, we simply speak about the order of $A$. Note

that if $ord_A(\Gamma)$ is finite, then $A$ is closed in $\Gamma$. For all $\lambda\tau$-types $T$ we have that the order of $T$ is finite.

**Definition 3.2.**

1. A *substitution* is a finite set of triples $\langle x_i\,;\,\gamma_i\,;\,M_i\rangle$, such that the $x_i$ are pairwise distinct, $\gamma_i$ is an existential context and $dom(\gamma_i)$ consists of fresh variables, possibly occurring in $M_i$. We let $\sigma$, $\sigma'$, $\tau$, ... range over substitutions.

2. If $\langle x\,;\,\gamma\,;\,M\rangle \in \sigma$, then we say that $\sigma$ *binds* $x$. Put $dom(\sigma) = \{x \mid x \text{ bound by } \sigma\}$. $M$ is called a *substitution term*, $\gamma$ a *substitution context*. To indicate that the substitution context is an "auxiliary" context, we denote it by a small Greek letter.

3. A substitution $\sigma$ is extended to a function on pseudo-terms as follows:

$$\begin{array}{rcl}
\sigma(c) & = & c \quad \text{for } c \in \mathcal{C} \\
\sigma(x) & = & \begin{cases} M & \text{if } \langle x\,;\,\gamma\,;\,M\rangle \in \sigma \\ x & \text{otherwise.} \end{cases} \\
\sigma(A_1A_2) & = & \sigma(A_1)\sigma(A_2) \\
\sigma(\lambda x{:}A_1.A_2) & = & \lambda x{:}\sigma(A_1).\sigma(A_2) \\
\sigma(\Pi x{:}A_1.A_2) & = & \Pi x{:}\sigma(A_1).\sigma(A_2)
\end{array}$$

   If $\sigma = \{\langle x_i\,;\,\gamma_i\,;\,M_i\rangle \mid 1 \leq i \leq n\}$, then we sometimes write $\sigma(A)$ as $A[x_1 := M_1, \ldots, x_n := M_n]$ or $A[\vec{x} := \vec{M}]$.

4. A substitution $\sigma$ is extended to a function on pseudo-contexts as follows:

$$\begin{array}{rcl}
\sigma(\langle\rangle) & = & \langle\rangle \\
\sigma(\Gamma, Qx : C) & = & \begin{cases} \sigma(\Gamma), \gamma & \text{if } Q = \exists \text{ and } \langle x\,;\,\gamma\,;\,M\rangle \in \sigma \\ \sigma(\Gamma), Qx : \sigma(C) & \text{otherwise.} \end{cases}
\end{array}$$

5. Let $\Gamma$ be a legal context in $\lambda\circ$. Then we call $\sigma$ *well-typed in* $\Gamma$ when the following holds:

   (a) $\sigma$ binds no variables that are universal in $\Gamma$.

   (b) $\sigma(\Gamma)$ is legal in $\lambda\circ$.

   (c) For all existential variables $x$ in $\Gamma$ that are bound by $\sigma$ we have that $\sigma(\Gamma_x), \gamma \vdash_{\lambda\circ} M : \sigma(\Gamma(x))$, where $\langle x\,;\,\gamma\,;\,M\rangle$ is the unique triple in $\sigma$ that binds $x$. In other words, we start with $\Gamma_x \vdash_{\lambda\circ} x : \Gamma(x)$ and end up with $\sigma(\Gamma_x) \vdash_{\lambda\circ} \sigma(x) : \sigma(\Gamma(x))$.

   We assume that the substitution terms of well-typed substitutions are in LNF.

6. Let $\sigma$ and $\tau$ be substitutions. We define

$$\sigma\circ\tau = \{\langle x\,;\,\sigma(\gamma)\,;\,\sigma(t)\rangle \mid \langle x\,;\,\gamma\,;\,t\rangle \in \tau\} \cup \{\langle x\,;\,\gamma\,;\,t\rangle \in \sigma \mid x \text{ not bound by } \tau\}.$$

7. Let $\sigma$ be well-typed in some context $\Gamma$ which is legal in $\lambda\circ$ and suppose $dom(\sigma) \subseteq dom(\Gamma)$. We can write $\sigma$ uniquely as $\sigma = \sigma_\Box \cup \sigma_*$, where $\sigma_s$ is the set of triples $\langle x, \gamma, M\rangle$ such that $\Gamma_x \vdash_{\lambda\circ} \Gamma(x) : s$. For a fixed sort $s$, $\sigma$ is said to be an $s$-substitution if for every triple $\langle x\,;\,\gamma\,;\,M\rangle$ we have $\Gamma_x \vdash_{\lambda\circ} \Gamma(x) : s$. (Note that $\sigma_\Box$ and $\sigma_*$ depend on $\Gamma$.)

**Remark 3.3.** Let $\Gamma$ be legal and $\sigma$ be well-typed in $\Gamma$. Then by inspection of the definition of substitutions and well-typedness one easily verifies that for all triples $\langle x\,;\,\gamma\,;\,t\rangle$ in $\sigma$ and all $y$ in $dom(\sigma)$, we have: $y \notin FV(t)$. Given that contexts are assumed to be in the form described in Lemma 2.5 (9) and given Lemma 2.5 (3), it is no restriction to assume that a declaration of the form $Qx : *$ only occurs in a substitution context in $\sigma_\Box$ and that the substitution contexts in $\sigma_\Box$ only contain such declarations. From now on we assume that substitutions satisfy this restriction. It is easy to check that the application of such substitutions to contexts of the form described in Lemma 2.5 (9) yields contexts of the same form. A consequence of (and motivation for) this convention is that $\sigma_\Box$ is well-typed in $\Gamma$ (and $\sigma_*$ well-typed in $\sigma_\Box(\Gamma)$).

**Definition 3.4.**

1. A *unification problem in* $\lambda\circ$ is a triple $\langle\Gamma\,;\,A\,;\,B\rangle$, where $\Gamma$ is a quantified context such that there exists a term $C$ such that $\Gamma \vdash_{\lambda\circ} A : C : s$ and $\Gamma \vdash_{\lambda\circ} B : C$. We assume that $\Gamma$, $A$ and $B$ are in LNF. If $\Gamma \vdash_{\lambda\circ} C : *$, then we say that $\langle\Gamma\,;\,A\,;\,B\rangle$ is a unification problem *for objects*; if $\Gamma \vdash_{\lambda\circ} C : \Box$, then we say that $\langle\Gamma\,;\,A\,;\,B\rangle$ is a unification problem *for types*. Note that in $\lambda\tau$ there are no unification problems for types.

2. A *matching problem in* $\lambda\circ$ is a unification problem $\langle\Gamma\,;\,A\,;\,B\rangle$ in $\lambda\circ$ such that $B$ is closed in $\Gamma$. Note that this implies that $C$ is closed in $\Gamma$.

3. A unification (or matching) problem $\langle\Gamma\,;\,A\,;\,B\rangle$ is *of order* $n$ if the types of the existential variables in $\Gamma$ have order at most $n$ in $\Gamma$.

4. A *solution* for a unification problem $\langle\Gamma\,;\,A\,;\,B\rangle$ in $\lambda\circ$ is a substitution $\sigma$, well-typed in $\Gamma$, such that $\sigma(A) =_{\beta\eta} \sigma(B)$. (So a solution for a *matching* problem $\langle\Gamma\,;\,A\,;\,B\rangle$ in $\lambda\circ$ is a substitution $\sigma$, well-typed in $\Gamma$, such that $\sigma(A) =_{\beta\eta} B$.) $\sigma$ is called a solution for a collection of unification (matching) problems $\{P_i \mid i \in I\}$ if $\sigma$ is a solution for every $P_i$. By a standard argument, a set of unification (matching) problems $\{\langle\Gamma\,;\,A_i\,;\,B_i\rangle \mid 1 \leq i \leq n\}$ (for some $n \in \mathbb{N}$) can be encoded as a single unification (matching) problem. We call the unification (matching) problems in such a set $\Gamma$-*compatible*. For fixed sort $s$, a solution $\sigma$ for $\langle\Gamma\,;\,A\,;\,B\rangle$ is called an *$s$-solution* for $\langle\Gamma\,;\,A\,;\,B\rangle$ if $\sigma$ is an $s$-substitution (w.r.t. $\Gamma$).

In the next section we give an example of a third-order matching problem in $\lambda 2$ and its solution. We end this section by stating some properties of matching problems and substitutions.

**Lemma 3.5.**

1. *Suppose* $\Gamma \vdash_{\lambda\circ} A : B$ *and let* $\sigma$ *be well-typed in* $\Gamma$. *Then* $\sigma(\Gamma) \vdash_{\lambda\circ} \sigma(A) : \sigma(B)$.

2. *Suppose* $\Gamma \vdash_{\lambda\circ} A : s$ *and let* $\sigma$ *be well-typed in* $\Gamma$. *Then* $ord_{\sigma(\Gamma)}(\sigma(A)) \leq ord_\Gamma(A)$.

3. *Let $\Gamma$ be legal, $\tau$ well-typed in $\Gamma$ and $\sigma$ well-typed in $\tau(\Gamma)$. Then $\sigma\circ\tau$ is well-typed in $\Gamma$ and $(\sigma\circ\tau)(\Gamma) = \sigma(\tau(\Gamma))$.*

**Proof.** See [4]. ⊠

**Lemma 3.6.** *Let $P = \langle\Gamma\,;\,A\,;\,B\rangle$ be a matching problem for types in $\lambda 2$ and $\tau$ a solution for $P$. Then $sol(P) = \{\langle x\,;\,\langle\rangle\,;\,M\rangle \mid \langle x\,;\,\gamma;\,M\rangle \in \tau \wedge x \in FV(A)\}$ is a solution for $P$ and for all solutions $\rho$ for $P$ we have: every $x$ bound by $sol(P)$ is bound by $\rho$ and the substitution term for $x$ in $\rho$ equals the substitution term for $x$ in $sol(P)$. Note that $sol(P)$ is well-defined.*

**Proof.** By induction on the length of $A$, using Lemma 2.5. ⊠

The following technical result is used in Subsection 6.4.

**Lemma 3.7.** *Let $\Gamma$ be legal in $\lambda 2$ and $\sigma$ a $*$-substitution, well-typed in $\Gamma$. Suppose that $\sigma(\Gamma) \vdash_{\lambda 2} A : B$, where $A$ is closed in $\sigma(\Gamma)$ and every variable in $A$ is universal in $\Gamma$ and of closed type in $\Gamma$. Then $\Gamma \vdash_{\lambda 2} A : B$ and $A$ is closed in $\Gamma$.*

**Proof.** By Thinning and Strengthening and the fact that types are not affected by $*$-substitutions. ⊠

# 4 Outline

The aim of this paper is to establish the decidability of third-order matching in $\lambda 2$. Roughly speaking, the proof is a reduction to third-order matching in $\lambda\tau$, which is decidable by [5] (but see the remark after Theorem 9.6). An important part of the reduction is a translation that maps third-order matching problems of a certain format in $\lambda 2$ (and their solutions) to third-order matching problems in $\lambda\tau$ (and their solutions) in such a way that the original problem has a solution iff the translated problem has a solution. The translation is given in Section 8. In order to explain the translation, we sketch two problems that we encountered while constructing it.

The first problem can be described as follows. Consider the judgement $\langle\forall A : *, \forall B : *, \forall x : \Pi\alpha{:}*.\alpha\rangle \vdash_{\lambda 2} x(A{\to}B)(xA) : B$. We see that the type $A{\to}B$ occurs as argument of the variable $x$ (therefore $A{\to}B$ is called a *polymorphic argument*). There is a straightforward way to encode such occurrences of types as terms of type $O$, using fresh variables $c^{\to}$, $c^{\Pi}$ that encode the type forming operators of $\lambda 2$. But it is hopeless to find a $\lambda\tau$-type that corresponds to $\Pi\alpha{:}*.\alpha$ such that when we assign this type to $x$, the translated judgement is valid in $\lambda\tau$. For each occurrence of $x$ we would have to assign a different $\lambda\tau$-type to $x$. Instead, we take a simple translation for $\Pi\alpha{:}*.\alpha$ and add, for each occurrence of $x$, a fresh variable which fixes the resulting type inequalities. The translated judgement becomes: $\Gamma \vdash_{\lambda 2} p_1 x(c^{\to}AB)(p_2 xA) : O$, where $\Gamma \equiv \langle\forall c^{\to} : O{\to}O, \forall p_1 : (O{\to}O){\to}O{\to}O{\to}O, \forall p_2 : (O{\to}O){\to}O{\to}O, \forall A : O, \forall B : O, \forall x : O{\to}O\rangle$. We have to make sure that each time we use an auxiliary variable of a certain $\lambda\tau$-type, we take the same variable. Otherwise, terms that are equal in $\lambda 2$ are translated to inequal terms in $\lambda\tau$.

This solves the problem that concerns the "static" aspect of polymorphism, i.e. the aspect related to the fact that types of terms may depend on polymorphic arguments in that term. Now we turn to the second problem, which concerns the "dynamic" aspect of polymorphism, i.e. the fact that, by $\beta$-reduction, types may be moved around. Consider the third-order matching problem $P = \langle \Gamma\,;\, A\,;\, B\rangle$, where $\Gamma \equiv \langle \forall C : *, \forall D : *, \forall x : \Pi\alpha{:}*.(\alpha{\to}\alpha){\to}D, \exists f : *{\to}D\rangle$, $A \equiv f(C{\to}D)$ and $B \equiv x(C{\to}D)(\lambda y{:}C{\to}D.\lambda z{:}C.yz)$. A solution for $P$ is $\sigma = \{\langle f\,;\, \langle\rangle\,;\, \lambda\alpha{:}*.x\alpha(\lambda y{:}\alpha.y)\rangle\}$. To see that $\sigma$ is a solution for $P$, check that $(\lambda\alpha{:}*.x\alpha(\lambda y{:}\alpha.y))(C{\to}D)$ $\beta$-reduces in one step to $x(C{\to}D)(\lambda y{:}C{\to}D.y)$ and that the latter term is the $\beta\eta$-normal form of $B$. We see that in the $\beta$-reduction step, $C{\to}D$ is substituted for $\alpha$ in $\lambda y{:}\alpha.y$, changing the arrow structure of the domain in that term. In the spirit of the translation sketched above, we would translate $\lambda y{:}\alpha.y$ to $\lambda y{:}O.y$ and $\lambda y{:}C{\to}D.y$ to $\lambda y{:}O{\to}O.y$. Now the problem is: how do we represent in $\lambda\tau$ the substitution of types for variables in types? Although we have made various attempts, we did not succeed in solving this problem directly.

Fortunately, we do not have to solve the problem in its full form. Note that if the argument of $\lambda\alpha{:}*.x\alpha(\lambda y{:}\alpha.y)$ in $\sigma(A)$ is an atomic type $E$ rather than $C{\to}D$ the problem would not occur. Then the $\beta$-reduction step would change $\lambda y{:}\alpha.y$ into $\lambda y{:}E.y$ and both terms are translated to $\lambda y{:}O.y$. This turns out to be the key idea. For we can define a translation with the desired properties for third-order matching problems in the so-called *restricted* format. A matching problem $P = \langle \Gamma\,;\, A\,;\, B\rangle$ is called restricted when (among other things) for every existential object variable $f$ in $A$ all polymorphic arguments of all occurrences of $f$ are atomic types. The effect of this is that when we substitute a term for $f$, the polymorphic arguments of an occurrence of the substituted term are all atomic. Moreover we show that for every third-order matching problem in $\lambda 2$ one can effectively find a restricted third-order matching problem such that the restricted problem has a solution iff the original problem has a solution. This suffices to establish decidability of third-order matching in $\lambda 2$.

Although all this works out fine, we should say that the present proof only works for matching problems of order 3. We will in the proof indicate the places where things go wrong if the restriction to order 3 is relaxed.

## 5 Properties of $\lambda 2$-terms

In this section we will define several notions, culminating in the notion "restricted". Along the way we prove some important lemmas about these notions.

**Definition 5.1.** Suppose $\Gamma \vdash_{\lambda 2} A : B$. Let $n \in \mathbb{N} \cup \{\infty\}$. An *$n$-redex in $A$* is a $\beta$-redex $(\lambda x{:}A_1.A_2)A_3$ in $A$ such that the order of the type of $\lambda x{:}A_1.A_2$ (in the current context) is at most $n$. A *polymorphic $\beta$-redex* in $A$ is a $\beta$-redex in $A$ of the form $(\lambda x{:}*.A_2)A_3$. A term is called *finite-redexed* if there exists an $n \in \mathbb{N}$ such that all $\beta$-redexes in $A$ are $n$-redexes. (This classification of $\beta$-redexes is vaguely similar to the classification of redexes in [11], Definition 72.)

**Lemma 5.2.** *Suppose that $\Gamma \vdash_{\lambda 2} (\lambda x{:}A_1.A_2)A_3 : B$, where $\Gamma \vdash_{\lambda 2} \lambda x{:}A_1.A_2 : \Pi x{:}A_1.B_2$ and $ord_\Gamma(\Pi x{:}A_1.B_2)$ is finite. Then we have: $\Gamma, \forall x : A_1 \vdash_{\lambda 2} A_2 : B$ and $x$ does not occur in $B$. (So $\Gamma \vdash_{\lambda 2} \lambda x{:}A_1.A_2 : A_1{\to}B$.)*

**Proof.** By Subject Reduction we know that $\Gamma \vdash_{\lambda 2} A_2[x := A_3] : B$. By Lemma 2.5 (6) we know that $\Gamma, \forall x : A_1 \vdash_{\lambda 2} A_2 : B_2$ and $\Gamma \vdash_{\lambda 2} A_3 : A_1$. So by the Substitution Lemma $\Gamma \vdash_{\lambda 2} A_2[x := A_3] : B_2[x := A_3]$. If $\Gamma \vdash_{\lambda 2} A_1 : *$, then $x$ is an object variable and does not occur in $B_2$. Hence $B_2[x := A_3] \equiv B_2 \equiv B$ (the second equality holds by Unicity of Types). If $A_1 \equiv *$, then $x \notin FV(B_2)$ because, otherwise, the type of $\lambda x{:}A_1.A_2$ would be polymorphic hence of order $\infty$, contradicting the assumption. So again $B_2[x := A_3] \equiv B_2 \equiv B$. We have treated all possible cases. ⊠

**Definition 5.3.** Suppose $\Gamma \vdash_{\lambda 2} A : B$. We say that $A$ is *atomic-polymorphic-redexed* (APR, for short) if for all subterms in $A$ of the form $(\lambda x{:}C_1.C_2)D_1 \ldots D_n$ with $n \geq 1$ the following holds for every $1 \leq i \leq n$: if $D_i$ is a type then $D_i$ is a universal variable (in the current context).

One easily checks that both the property of being APR and the property that every $\beta$-redex is an $n$-redex (for finite $n$) hold for $\beta$-normal terms but are not preserved under $\beta$-reduction. Fortunately, the *conjunction* of the property of being APR and the property that every $\beta$-redex is a 3-redex is preserved under $\beta$-reduction.

**Definition 5.4.** Suppose $\Gamma \vdash_{\lambda 2} A : B$. We say that $A$ is *simple* if $A$ is APR and every $\beta$-redex in $A$ is a 3-redex.

Simple terms have the important property that the translation of Section 8 preserves $\beta$-reduction on simple terms. To prove this it is essential to prove that simplicity is closed under $\beta$-reduction.

Note that the conjunction of the property of being APR and the property that every $\beta$-redex is a 4-redex is *not* preserved under $\beta$-reduction. Consider $A \equiv (\lambda x{:}* \rightarrow B.x(\Pi\alpha{:}*.\alpha))\lambda\beta{:}*.y\beta$ where $B$ is a universal variable of type $*$ and $y$ is a variable of type $* \rightarrow B$. Then $A$ is APR and its only $\beta$-redex is a 4-redex. But $A$ $\beta$-reduces in one step to $(\lambda\beta{:}*.y\beta)(\Pi\alpha{:}*.\alpha)$ which is not APR. This is the first place where the restriction to *third*-order matching is essential.

We prove that simplicity is preserved by $\beta$-reduction. As usual, we first prove a substitution lemma.

**Lemma 5.5.** *Suppose $\Gamma, Qx : D, \Delta \vdash_{\lambda 2} A : B$ and $\Gamma \vdash_{\lambda 2} C : D$. Suppose that $A$ and $C$ are simple. Suppose furthermore that $ord_\Gamma(D) \leq 2$ and that if $D \equiv *$ then $C$ is a universal variable in $\Gamma$. Then $A[x := C]$ is simple.*

**Proof.** By induction on the structure of $A$. The only non-trivial case is where $A \equiv A_1 A_2$. We distinguish several cases. For $T \in \mathcal{T}$, write $T^*$ for $T[x := C]$.

- $A_1 \equiv xD_1 \ldots D_n$ $(n \geq 0)$. Because of typing reasons, $D \not\equiv *$ and none of the $D_i$'s or $A_2$ are types. Assume $C \equiv \lambda y{:}C_1.C_2$. Then $A^* \equiv (\lambda y{:}C_1.C_2)D_1^* \ldots D_n^* A_2^*$. By induction hypothesis, $D_1^*, \ldots, D_n^*, A_2^*$ are simple (and they are not types!). Because $ord_\Gamma(D) \leq 2$, the redex in $A^*$ that is explicitly shown is of order at most 2. So $A^*$ is simple.

- $A_1 \equiv \lambda y{:}D_1.D_2$. If $x$ is an object variable, then the result follows immediately from the induction hypothesis, because object variables do not occur in types. If $x$ is a type variable then the result follows easily because substitution of universal type variables for type variables does not destroy the properties in question.

- In all other cases the result follows immediately from the induction hypothesis.

⊠

**Lemma 5.6.** *Let $\Gamma \vdash_{\lambda 2} A : B$, where $A$ is simple. Suppose $A \twoheadrightarrow_\beta A'$. Then $A'$ is simple.*

**Proof.** By induction on the generation of $\twoheadrightarrow_\beta$, using Lemma 5.5. We only treat the case where $A \equiv A_1 A_2$ and $A \rightarrow_\beta A'$.

- $A' \equiv A_1 A_2'$ and $A_2 \rightarrow_\beta A_2'$.
  - $A_1 \equiv \lambda x{:}*.A_1'$. This is impossible because $A_2$ would be a type and types are in $\beta$-normal form.
  - All other cases follow immediately from the induction hypothesis.
- $A' \equiv A_1' A_2$ and $A_1 \rightarrow_\beta A_1'$. We only treat the case where $A_1 \equiv (\lambda x{:}D_1.D_2)E_1 \ldots E_n$ (for some $n \geq 0$). In the other cases the result follows immediately from the induction hypothesis. If the reduction step takes place in a term $E_i$ ($1 \leq i \leq n$), then the result follows from the induction hypothesis (remember that types are $\beta$-normal). If the reduction step takes place in $D_2$ (yielding $D_2'$) then the result follows from the induction hypothesis plus the fact that by Subject Reduction the type of $\lambda x{:}D_1.D_2'$ is the same as the type of $\lambda x{:}D_1.D_2$. The only remaining case is the contraction of the redex that is explicitly shown. If $n = 0$ then, by induction, we have already considered this case (the $\beta$-step); so suppose $n > 0$. Consider the result of contracting this redex: $(D_2[x := E_1])E_2 \ldots E_n A_2$. By Lemma 5.5, $D_2[x := E_1]$ is simple. Suppose that $D_2[x := E_1]$ is of the form $\lambda y{:}F_1.F_2$. Let $\Gamma \vdash_{\lambda 2} \lambda y{:}F_1.F_2 : \Pi y{:}F_1.G$. We have to check that $ord_\Gamma(\Pi y{:}F_1.G) \leq 3$. Let $\Gamma, \forall x : D_1 \vdash_{\lambda 2} D_2 : G'$. Since $A$ is simple, $ord_{\Gamma, \exists x:D_1}(G') \leq 3$. So $x \notin FV(G')$. By the Substitution Lemma, $\Gamma \vdash_{\lambda 2} D_2[x := E_1] : G'[x := E_1]$. So $\Gamma \vdash_{\lambda 2} D_2[x := E_1] : G'$. Thus $\Pi y{:}F_1.G \equiv G'$. We are done.

⊠

Now we describe the situation in which simple terms appear in our analysis of matching problems and solutions.

**Definition 5.7.** Suppose $\Gamma \vdash_{\lambda 2} A : B$, with $A$ in LNF. Let $X = \{x_1, \ldots, x_n\} \subseteq FV(A)$ be a set of object variables in $\Gamma$. Then $A$ is called *simplified w.r.t. $X$* if for all $x \in X$, for all occurrences $xt_1 \ldots t_m$ of $x$ in $A$ and for all $1 \leq i \leq m$ we have that if $t_i$ is a type then it is a universal variable (in the current context). We call a matching problem $\langle \Gamma\,;\, A\,;\, B \rangle$ simplified when $A$ is simplified w.r.t. the set of free object variables in $A$ that are existential in $\Gamma$.

**Lemma 5.8.** *Let $\Gamma \vdash_{\lambda 2} A : B$, with $A$ in* LNF. *Suppose $A$ is simplified w.r.t. $X = \{x_1, \ldots, x_n\}$. Suppose that for all $x \in X$, $ord_\Gamma(\Gamma(x)) \leq 3$. Let $\{S_1, \ldots, S_n\}$ be a set of terms in LNF such that, for all $1 \leq i \leq n$, $\Gamma_{x_i} \vdash_{\lambda 2} S_i : \Gamma(x) : *$ and, for all $1 \leq j \leq n$, $x_j \notin FV(S_i)$. Then $A[\vec{x} := \vec{S}]$ is simple.*

**Proof.** From the fact that the variables in $X$ are object variables and do not occur in any of the terms $S_i$, it easily follows that $\Gamma \setminus X \vdash_{\lambda 2} A[\vec{x} := \vec{S}] : B$. Simplicity follows immediately from the definition of *simplified*. ⊠

The following specialization of simplicity will also be useful.

**Definition 5.9.** Suppose $\Gamma \vdash_{\lambda 2} A : B$. We call $A$ *redex monomorphic* if for all subterms in $A$ of the form $(\lambda x{:}C_1.C_2)D_1 \ldots D_n$ with $n \geq 1$ we have that no $D_i$ $(1 \leq i \leq n)$ is a type. Note that such $A$ are also APR. We call $A$ *hypersimple* when $A$ is redex monomorphic and every $\beta$-redex in $A$ is a 3-redex. Note that, indeed, every hypersimple term is simple. Terms that are $\beta$-normal are hypersimple.

The essence of hypersimplicity is that in $\beta\eta$-reduction sequences, hypersimple terms behave like $\lambda\tau$-terms. We show that hypersimplicity is preserved under $\beta\eta$-reduction.

**Lemma 5.10.** *Suppose $\Gamma, Qx : D, \Delta \vdash_{\lambda 2} A : B$ and $\Gamma \vdash_{\lambda 2} C : D$. Suppose that $A$ and $C$ are hypersimple. Suppose furthermore that $ord_\Gamma(D) \leq 2$ and that $D \not\equiv *$. Then $A[x := C]$ is hypersimple.*

**Proof.** By induction on the structure of $A$. Details are similar to the details in the proof of Lemma 5.5. ⊠

**Lemma 5.11.** *Let $\Gamma \vdash_{\lambda 2} A : B$, where $A$ is hypersimple. Suppose $A \twoheadrightarrow_{\beta\eta} A'$. Then $A'$ is hypersimple.*

**Proof.** By induction on the generation of $\twoheadrightarrow_\beta$, using Lemma 5.10. Details are similar to the details in the proof of Lemma 5.6, except for the details concerning $\eta$-reduction steps. These do not cause any trouble. The only interesting case for $\eta$-reduction is where $A \equiv (\lambda x{:}A_1.A_2x)A_3$, $x \notin FV(A_2)$, $A' \equiv A_2A_3$ and $A_2$ is an abstraction term. One easily checks that $A_2A_3$ is hypersimple. The order of the type of $A_2$ is the order of the type of $\lambda x{:}A_1.A_2x$, hence this order is less than or equal to 3. By assumption, $A_3$ is not a type. ⊠

The following definition concerns the arguments of existential object variables in matching problems.

**Definition 5.12.** Suppose $\Gamma \vdash_{\lambda 2} A_1 \to \cdots \to A_n \to A : *$, with $n \geq 0$ and $A$ atomic. Suppose that $ord_\Gamma(A_1 \to \cdots \to A_n \to A)$ is finite. Note that $A \not\equiv *$. We call $A_1 \to \cdots \to A_n \to A$ *structured* when there exists $1 \leq i \leq n$ such that $A_1, \ldots, A_i$ are all equal to $*$ and none of $A_{i+1}, \ldots, A_n$ are equal to $*$. It is easy to see that for each type $B$ as above there exists a permutation $\phi_B$ of $\{1, \ldots, n\}$ such that $A_{\phi_B(1)} \to \ldots \to A_{\phi_B(n)} \to A$ is structured. We denote this type by $\phi(B)$. We call a matching problem $\langle \Gamma\,;\, A\,;\, B \rangle$ of finite order in $\lambda 2$ structured when all the types of the existential object variables that occur in $A$ are structured.

We need one more notion to define the concept "restricted".

**Definition 5.13.** Let $P = \langle \Gamma\,;\, A\,;\, B \rangle$ be a matching problem in $\lambda 2$. We call $P$ *type-closed* when either $P$ is a matching problem for types, or the type of every variable in $\Gamma$ is closed in $\Gamma$ and every type $T$ in $A$ is closed in its current context. We call $P$ *restricted* when $P$ is structured, type-closed and simplified.

Notice that every third-order matching problem for types is restricted. So there exists a trivial reduction of third-matching for types to restricted third-order matching for types. The difficult part is to reduce third-order matching for objects to restricted third-order matching for objects. This will be the subject of the next section.

# 6 Reduction to restricted third-order matching

In Section 8 and Section 9, we will establish that it is decidable whether a restricted third-order matching problem $P = \langle \Gamma\,;\, A\,;\, B\rangle$ in $\lambda 2$ has a solution or not. This settles the case of matching problems for types. In this section we will reduce third-order matching for objects to restricted third-order matching for objects.

In Subsection 6.1, we prove (Lemma 6.1) that we may w.l.o.g. assume that $P$ is structured. In Subsection 6.2, we will prove (Lemma 6.6) that it is no restriction to assume that the types of the free variables in $P$ are closed. This is used in Subsection 6.3, where we prove (Lemma 6.9) that it is no restriction to assume that $P$ is type-closed. So a solution may be assumed to be $*$-substitution that will leave the polymorphic arguments in $A$ unaffected This allows us to replace w.l.o.g. certain polymorphic arguments in $A$ by fresh universal type variables, obtaining a simplified matching problem.

## 6.1 Reduction to the structured case

**Lemma 6.1.** *Suppose $P = \langle \Gamma\,;\, A\,;\, B\rangle$ is a matching problem of finite order in $\lambda 2$. From $P$ we can (effectively) construct structured matching problem $P'$ (of finite order in $\lambda 2$) such that $P$ has a solution iff $P'$ has a solution.*

**Proof.** Let $X$ be the set of existential object variables in $\Gamma$ that are free in $A$. Let $x \in X$; write $\Gamma(x) \equiv S_1 \to \cdots \to S_n \to S$. Let $x'$ be a fresh variable of type $\phi(\Gamma(x))$. Let $t_x \equiv \lambda y_1{:}S_1 \ldots \lambda y_n{:}S_n.x' y_{\phi_{\Gamma(x)}(1)} \ldots y_{\phi_{\Gamma(x)}(n)}$. Put $\rho = \{\langle x\,;\, \langle \exists x' : \phi(\Gamma(x))\rangle\,;\, t_x\rangle \mid x \in X\}$. This substitution is obviously well-typed in $\Gamma$. Let $A'$ be the $\beta$-normal form of $\rho(A)$ and let $\Gamma'$ be $\rho(\Gamma)$. Put $P' = \langle \Gamma'\,;\, A'\,;\, B\rangle$. Then $P'$ is an structured matching problem of finite order in $\lambda 2$. Suppose $P$ has solution $\sigma$. We produce a solution $\sigma'$ for $P'$. For $x$ as above, we let $s_x \equiv \lambda y_{\phi_{\Gamma(x)}(1)}{:}S_{\phi_{\Gamma(x)}(1)} \ldots \lambda y_{\phi_{\Gamma(x)}(n)}{:}S_{\phi_{\Gamma(x)}(n)}.x y_1 \ldots y_n$. Define $\rho^{-1}$ as $\{\langle x'\,;\, \langle \exists x : \Gamma(x)\rangle\,;\, s_x\rangle \mid x \in X\}$. One can check that $\rho^{-1}$ is well-typed in $\rho(\Gamma)$, that $\rho^{-1}(\rho(\Gamma)) \equiv \Gamma$ and $\rho^{-1}(\rho(A)) \twoheadrightarrow_\beta A$. So $\sigma \circ \rho^{-1}$ is a solution for $P'$. Conversely, let $\tau$ be a solution for $P'$. Then $\tau' \circ \rho$ is a solution for $P$. $\boxtimes$

The motivation for structuring third-order matching problems $P = \langle \Gamma\,;\, A\,;\, B\rangle$ for objects in $\lambda 2$ is that if $\sigma$ is a solution for $P$ then all $\beta$-redexes in $\sigma(A)$ are as described in Lemma 6.2 and there exists a very surveyable $\beta\eta$-reduction path from $\sigma(A)$ to $B$. We can contract all polymorphic $\beta$-redexes first, after which all terms in the reduction sequence will be hypersimple and behave like $\lambda\tau$-terms. The existence of such a nice reduction path will make it easier to analyse what happens to the subterms of $A$ after application of $\sigma$ to $A$. This analysis is given in the Commutation Lemma below. This lemma allows us to prove that it is no restriction to assume that matching problems are restricted.

**Lemma 6.2.** *Let $A \equiv (\lambda x_1{:}* \ldots \lambda x_n{:}*.\lambda y_1{:}S_1 \ldots \lambda y_m{:}S_m.E)C_1 \ldots C_n D_1 \ldots D_m$ with $n, m \geq 0$, $E$ $\beta$-normal and not an abstraction term. Suppose that $\Gamma \vdash_{\lambda 2} A : B$ and that for all $1 \leq i \leq m$, $S_i \not\equiv *$ and $D_i$ is hypersimple. Let $ord_\Gamma(\lambda x_1{:}* \ldots \lambda x_n{:}*.\lambda y_1{:}S_1 \ldots \lambda y_m{:}S_m.E) \leq 3$. The result of $n$ times contracting the outermost $\beta$-redex in $A$ is $(\lambda y_1{:}S_1 \ldots \lambda y_m{:}S_m.E[\vec{x} := \vec{C}])D_1 \ldots D_m$ and this term is hypersimple.*

**Proof.** For typing reasons, no $x_i$ $(1 \leq i \leq n)$ occurs in a domain $S_j$ $(1 \leq j \leq m)$. Clearly, $E[\vec{x} := \vec{C}]$ is $\beta$-normal, hence hypersimple. Because of typing reasons, none of the $D_i$'s are types. So it remains to show that $ord_\Gamma(\lambda y_1{:}S_1 \ldots \lambda y_m{:}S_m.E[\vec{x} := \vec{C}]) \leq 3$. This follows from Lemma 5.2. ⊠

**Lemma 6.3.** *Suppose $\Gamma \vdash_{\lambda 2} A : B$ and that all $\beta$-redexes in $A$ are of the form as described in Lemma 6.2, except that we do not demand that the $D_i$'s are hypersimple. Exhaustively contracting polymorphic $\beta$-redexes in $A$ according to the Leftmost Innermost Reduction Order (see [8]) yields a hypersimple term.*

**Proof.** Use Lemma 6.2. Because we contract polymorphic $\beta$-redexes from the inside to the outside, we have that every contracted redex is of the form as described in that lemma. ⊠

**Lemma 6.4** *(Commutation Lemma)*. *Let $\Gamma \vdash_{\lambda 2} A : B$. Suppose that all $\beta$-redexes in $A$ are of the form $(\lambda x_1{:}* \ldots \lambda x_n{:}*.\lambda y_1{:}S_1 \ldots \lambda y_m{:}S_m.E)C_1 \ldots C_n D_1 \ldots D_m$, where $n, m \geq 0$, $E$ is $\beta$-normal and is not an abstraction term, $ord_\Gamma(\lambda x_1{:}* \ldots \lambda x_n{:}*.\lambda y_1{:}S_1 \ldots \lambda y_m{:}S_m.E) \leq 3$ and, for all $1 \leq i \leq m$, $S_i \not\equiv *$. Suppose that $A \twoheadrightarrow_{\beta\eta} A'$ via a reduction path that begins with contracting, exhaustively, all polymorphic $\beta$-redexes according to the Leftmost Innermost Reduction Order. Consider an occurrence of such a redex in $A$ and fix a term $C_i$ in that occurrence.*

1. *We can underline $C_i$ in $A$ and follow the underlined term in the reduction sequence described above to $A'$. At each ($\beta$- or $\eta$-) reduction step from $E$ to $F$ in this reduction sequence, either $F$ contains no underlined terms or the right projections in $Tree(\Gamma\,;\,E)$ that correspond to underlined terms in $E$ and the right projections in $Tree(\Gamma\,;\,F)$ that correspond to underlined terms in $F$ are syntactically equal (for a suitable choice of variables added in the construction of $Tree(\Gamma\,;\,E)$ and $Tree(\Gamma\,;\,F)$).*

2. *Write $E \xrightarrow{\theta} F$ if there is some ($\beta$- or $\eta$-) step from $E$ to $F$ in the reduction sequence described above. For $T \in \mathcal{T}$, write $T \xrightarrow{-} T$ if $T'$ is the result of replacing all underlined terms in $T$ (if any) by a fresh variable $z$ of type $*$. The following diagram commutes.*

$$\begin{tikzcd} E \arrow[r, "\theta"] \arrow[d, "-"'] & F \arrow[d, "-"'] \\ E' \arrow[r, "\theta"] & F' \end{tikzcd}$$

**Proof.**

1. When we, initially and exhaustively, contract all polymorphic $\beta$-redexes then in this part of the reduction sequence, $C_i$ may be removed and may be moved around, but is not changed. By Lemma 6.3, this results in a term $A''$ that is hypersimple. Contraction of a $\beta$-redex in a hypersimple term does not involve types except as domains. These

may be removed or moved around but are not changed. Finally, consider an $\eta$-step from $\lambda x{:}A_1.A_2x$ to $A_2$. Types in $\lambda x{:}A_1.A_2x$ may occur in $A_1$, in $A_2$ or may be $x$. Note that $A_2x$ is not itself a type. The $\eta$-step leaves the types in $A_2$ unchanged.

2. Commutation follows easily from (1).

$\boxtimes$

## 6.2 Removing variables with open types

In this subsection we show how to remove variables with open types from matching problems of finite order in $\lambda 2$.

**Definition 6.5.** Let $P = \langle \Gamma\,;\, A\,;\, B\rangle$ be a matching problem of finite order $n$ in $\lambda 2$. Let $\Delta$ be $\langle \forall c : \Pi\alpha{:}*.\alpha\rangle, \Gamma$, where $c$ is a fresh variable. Let $Z = \{z_1, \ldots, z_l\}$ be the set of variables $z \in dom(\Gamma)$ such that $\Gamma(z)$ is open in $\Gamma$. Note that such variables are object variables and that none of them are existential in $\Gamma$ or occur in $B$. Write $c_z \equiv c(\Gamma(z))$. Let $A'$ be the result of replacing every $z \in Z$ in $A$ by $c_z$. Let $\Gamma'$ be the result of deleting the variables in $Z$ from $\Delta$. Evidently, $\Gamma' \vdash_{\lambda 2} A' : C$. We write $P' = \langle \Gamma'\,;\, A'\,;\, B\rangle$. One easily checks that $P'$ is a matching problem of order $n$ in $\lambda 2$.

**Lemma 6.6.** *Let $P = \langle \Gamma\,;\, A\,;\, B\rangle$ be a matching problem in $\lambda 2$. Let $P'$ be as in Definition 6.5. Then $P$ has a solution $\Leftrightarrow$ $P'$ has a solution.*

**Proof.** Let $B'$ be nf$(B)$.

"$\Rightarrow$" Suppose $P$ has solution $\sigma$. Write $\sigma_\Box = \{\langle x_i\,;\, \gamma_i\,;\, T_i\rangle \mid 1 \leq i \leq n\}$ and $\sigma_* = \{\langle y_i\,;\, \delta_i\,;\, S_i\rangle \mid 1 \leq i \leq m\}$. For $z \in Z$, write $c'_z \equiv \sigma(c_z) \equiv \sigma_\Box(c_z) \equiv c(\sigma_\Box(\Gamma(z)))$. Let $\sigma'$ be the result of replacing every substitution term $S$ in $\sigma_*$ by $S' \equiv S[\vec{z} := \vec{c'_z}]$. We claim that $\sigma'$ is a solution for $P'$. Evidently, $\sigma'$ is well-typed in $\Gamma'$. We have to show that $\sigma'(A') \twoheadrightarrow_{\beta\eta} B'$ or in other words $A'[\vec{x} := \vec{T}][\vec{y} := \vec{S'}] \twoheadrightarrow_{\beta\eta} B'$. We know that $A[\vec{x} := \vec{T}][\vec{y} := \vec{S}] \twoheadrightarrow_{\beta\eta} B'$. Thus by Substitutivity $A[\vec{x} := \vec{T}][\vec{y} := \vec{S}][\vec{z} := \vec{c'_z}] \twoheadrightarrow_{\beta\eta} B'[\vec{z} := \vec{c'_z}] \equiv B'$. Now consider the following equivalences (that they are equivalences follows from [1], Proposition 2.1.6).

$A[\vec{x} := \vec{T}][\vec{y} := \vec{S}][\vec{z} := \vec{c'_z}] \twoheadrightarrow_{\beta\eta} B' \quad \Leftrightarrow$
$A[\vec{x} := \vec{T}][\vec{z} := \vec{c'_z}][\vec{y} := \vec{S'}] \twoheadrightarrow_{\beta\eta} B' \quad \Leftrightarrow$
$A[\vec{z} := \vec{c_z}][\vec{x} := \vec{T}][\vec{y} := \vec{S'}] \twoheadrightarrow_{\beta\eta} B' \quad \Leftrightarrow$
$A'[\vec{x} := \vec{T}][\vec{y} := \vec{S'}] \twoheadrightarrow_{\beta\eta} B'$.

We may conclude that $\sigma'$ is a solution for $P'$.

"$\Leftarrow$" Let $\tau$ be a solution for $P'$. Write $\tau_\Box = \{\langle x_i\,;\, \gamma_i\,;\, T_i\rangle \mid 1 \leq i \leq n\}$ and $\tau_* = \{\langle y_i\,;\, \delta_i\,;\, S_i\rangle \mid 1 \leq i \leq m\}$. Let $\tau'_*$ be $\{\langle x\,;\, \gamma'\,;\, S'\rangle \mid \langle x\,;\, \gamma\,;\, S\rangle \in \tau_*\}$, where $\gamma' = \gamma, \exists c_x : \Pi\alpha{:}*.\alpha$ ($c_x$ some fresh variable) and $S' \equiv S[c := c_x]$. Let $\tau'$ be $\tau'_* \cup \tau_\Box$. It is easy to see that $\tau'$ is well-typed in $\Gamma$ and in $\Gamma'$. By an easy Substitutivity argument, $\tau'(A') \twoheadrightarrow_{\beta\eta} B'$. We claim that $\tau'$ is a solution for $P$. We are done when we can show that $\tau'(A) \twoheadrightarrow_{\beta\eta} B'$. For every $z \in Z$, write $c''_z \equiv \tau_\Box(c_z)$. Let $A''$ be nf$(\tau'(A))$. We prove that $A'' \equiv B'$. We know that $A[\vec{x} := \vec{T}][\vec{y} := \vec{S'}] \twoheadrightarrow_{\beta\eta} A''$ and by Substitutivity $A[\vec{x} := \vec{T}][\vec{y} := \vec{S'}][\vec{z} := \vec{c''_z}] \twoheadrightarrow_{\beta\eta} A''[\vec{z} := \vec{c''_z}]$. Consider the following equivalences:

$A[\vec{x} := \vec{T}][\vec{y} := \vec{S'}][\vec{z} := \vec{c''_z}] \twoheadrightarrow_{\beta\eta} A''[\vec{z} := \vec{c''_z}] \quad \Leftrightarrow$
$A[\vec{x} := \vec{T}][\vec{z} := \vec{c''_z}][\vec{y} := \vec{S'}] \twoheadrightarrow_{\beta\eta} A''[\vec{z} := \vec{c''_z}] \quad \Leftrightarrow$
$A[\vec{z} := \vec{c_z}][\vec{x} := \vec{T}][\vec{y} := \vec{S'}] \twoheadrightarrow_{\beta\eta} A''[\vec{z} := \vec{c''_z}] \quad \Leftrightarrow$
$A'[\vec{x} := \vec{T}][\vec{y} := \vec{S'}] \twoheadrightarrow_{\beta\eta} A''[\vec{z} := \vec{c''_z}]$.

We also know that $B'$ is the $\beta\eta$-normal form of $A'[\vec{x} := \vec{T}][\vec{y} := \vec{S'}]$. Thus $A''[\vec{z} := \vec{c''_z}] \twoheadrightarrow_{\beta\eta} B'$. But $A''[\vec{z} := \vec{c''_z}]$ is $\beta\eta$-normal and thus $A''[\vec{z} := \vec{c''_z}] \equiv B$. Recall that $B$ does not contain a term $c''_z$. Thus $A''$ does not contain a $z \in Z$ and $A'' \equiv A''[\vec{z} := \vec{c''_z}] \equiv B''$. We are done. ⊠

With a little extra effort we may also remove existential variables with open types (declared in substitution contexts) from substitution terms. Thus we may assume that variables with open types occur neither in matching problems nor in solutions.

### 6.3 Reduction to the case with closed types

In this subsection we will show that we can without loss of generality assume that structured third-order matching problems for objects are type-closed. We need the following lemma.

**Lemma 6.7.** *Suppose $P = \langle \Gamma\,;\, A\,;\, B\rangle$ is a matching problem of finite order in $\lambda 2$, $\sigma$ a solution for $P$. Suppose $\Gamma \vdash_{\lambda 2} A : C$. Let $B'$ be nf$(B)$. In the previous subsection we have seen that it is no restriction to assume that every variable in $\Gamma$ or in $\sigma$ has a closed type.*

1. *Let $\sigma'_*$ be $\sigma_*$, except that every existential type variable in a substitution term in $\sigma_*$ is replaced by $\Pi\alpha{:}*.\alpha$. Then $\sigma'_*$ is well-typed in $\Gamma$.*

2. *Let $\Xi$ be the result of inserting, for each triple $\langle x\,;\,\gamma\,;\,M\rangle \in \sigma_\Box$, $\gamma$ to the left of $x$ in $\Gamma$. Then $\Xi \vdash_{\lambda 2} A : C$ and $\sigma_*$ is well-typed in $\Xi$. So nf$(\sigma_*(A))$ exists.*

3. *Let $\rho = \{\langle x\,;\,\langle\rangle\,;\,M\rangle \mid \langle x\,;\,\gamma\,;\,M\rangle \in \sigma_\Box$ and $x \in FV(\text{nf}(\sigma_*(A)))\}$. Note that each substitution term in $\rho$ is closed in $\Gamma$.*

   *Let $\rho^+$ be $\rho \cup \{\langle x\,;\,\langle\rangle\,;\,\Pi\alpha{:}*.\alpha\rangle \mid x$ existential type variable in $\rho(A)\}$. Then $\sigma'_*$ is well-typed in $\rho^+(\Gamma)$ and $\sigma'_*(\rho^+(A)) \twoheadrightarrow_{\beta\eta} B'$.*

   *Note that all types in $\rho^+(A)$ are closed in their current context.*

**Proof.**

1. Since all types of variables in $\Gamma$ are closed, the only effect of applying $\sigma_\Box$ to $\Gamma$ is that the (existential) variables in $dom(\sigma_\Box)$ are removed from $\Gamma$ while possibly new existential variables are added. This does not affect any other declaration in $\Gamma$. Since for each triple $\langle x\,;\,\gamma\,;\,M\rangle \in \sigma_*$ existential type variables occur neither in $M$ nor in $\Gamma(x)$ nor in the type of a variable in $\gamma$ we see by Strengthening that $M$ is still typable when we delete the variables inserted by $\sigma_\Box$ from $\sigma(\Gamma)$. By Thinning we can add the variables in $dom(\sigma_\Box)$ to $\sigma(\Gamma)$. The resulting context is exactly $\sigma'_*(\Gamma)$.

2. For similar reasons.

3. As before, $\sigma'_*$ is well-typed in $\rho^+(\Gamma)$. It is easy to see that $\sigma_*(\rho(A)) \twoheadrightarrow_{\beta\eta} B'$. From this it follows quickly that $\sigma'_*(\rho^+(A)) \twoheadrightarrow_{\beta\eta} B'$.

⊠

Let $P = \langle \Gamma\,;\, A\,;\, B\rangle$ be a structured third-order matching problem for objects in $\lambda 2$. In the following definition, the set $\Psi(P)$ is a formalization of: "for every set $S_1$ of existential type variables in $A$ and every set $S_2$ of free types in $B$, try to match $S_1$ with $S_2$".

**Definition 6.8.** Let $P = \langle \Gamma\,;\, A\,;\, B\rangle$ be a third-order matching problem for objects in $\lambda 2$. Let $X = \{x_1, \ldots, x_n\}$ be the set of existential type variables that are free in $A$. Let $\{b_1, \ldots, b_m\}$ be the set of free types in $B'$, where $B' \equiv \mathrm{nf}(B)$. For $1 \leq i \leq n$, define

$$\Phi_i(P) = \{\{\langle \Gamma\,;\, x_i\,;\, b_j\rangle\} \mid 1 \leq j \leq m\}.$$

Define $\Psi_0(P) = \{\emptyset\}$ and for $1 \leq i \leq n$, define

$$\Psi_i(P) = \Phi_i(P) \cup \Psi_{i-1}(P) \cup \bigcup_{p \in \Phi_i(P)} \bigcup_{q \in \Psi_{i-1}(P)} p \cup q.$$

Finally, put $\Psi(P) = \Psi_n(P)$. Note that each element $S$ of the finite set $\Psi(P)$ is a finite set of $\Gamma$-compatible matching problems for types. By Lemma 3.6 we have that if $S$ has a solution then $sol(S)$ exists. Note that all substitution terms in $sol(S)$ are closed.

Next we define

$$\Pi(P) = \{\sigma^+(P) \mid \text{there exists } S \in \Psi(P) \text{ such that } \sigma = sol(S)\},$$

where $\sigma^+ = \sigma \cup \{\langle x\,;\, \langle\rangle\,;\, \Pi\alpha{:}*.\alpha\rangle \mid x \text{ existential type variable in } \sigma(A)\}$. Note that each element of the finite set $\Pi(P)$ is a structured, type-closed matching problem for objects.

Next we prove that finding solutions for $P$ is equivalent to finding $*$-solutions for some matching problem in $\Pi(P)$.

**Lemma 6.9.** *Let $P = \langle \Gamma\,;\, A\,;\, B\rangle$ be a structured third-order matching problem for objects in $\lambda 2$. Then $P$ has a solution $\Leftrightarrow$ there exists a problem $Q \in \Pi(P)$ such that $Q$ has a $*$-solution.*

**Proof.**

"$\Rightarrow$" Let $\sigma$ be a solution for $P$. Let $B'$ be $\mathrm{nf}(B)$. Let $X$ be the set of existential type variables in $A$. We put $S = \{\langle \Gamma\,;\, x\,;\, \sigma_\Box(x)\rangle \mid x \in X \cap FV(\mathrm{nf}(\sigma_*(A)))\}$. By Substitutivity we have that for all $x \in FV(\mathrm{nf}(\sigma_*(A)))$, $\sigma_\Box(x)$ is a free type of $B'$ (hence closed in $\Gamma$). So $S \in \Psi(P)$ and $\sigma_\Box$ is a solution for $S$. Let $\rho$ be $sol(S)$. Put $Q = \rho^+(P)$, where $\rho^+$ is defined as before. Then $Q \in \Pi(P)$. By Lemma 6.7, we can change $\sigma_*$ to a $*$-substitution $\sigma'_*$ such that $\sigma'_*$ is well-typed in $\rho^+(\Gamma)$ and $\sigma'_*(\rho^+(A)) \twoheadrightarrow_{\beta\eta} B'$. Thus $\sigma'_*$ is a $*$-solution for $Q$.

"$\Leftarrow$" Let $\sigma$ be a solution for some $S \in \Psi(P)$ and let $\tau$ be a $*$-solution for $\sigma^+(P)$. Then $\tau \circ \sigma^+$ is a solution for $P$. $\boxtimes$

## 6.4 Reduction to the case of atomic type arguments

In this subsection we will show that we can without loss of generality assume that third-order matching problems for objects are restricted.

**Definition 6.10.** Let $\langle \Gamma\,;\, A\,;\, B\rangle$ be a structured, type-closed third-order matching problem for objects in $\lambda 2$. Suppose $\Gamma \vdash_{\lambda 2} A : C$ (and so $\Gamma \vdash_{\lambda 2} B : C$). Write $A'$ for the result of the following operation: for every occurrence of an existential variable $x$ in $A$, replace every polymorphic argument $T$ of that occurrence of $x$ by a variable $M_T$. We demand that all occurrences in $A$ of a polymorphic argument that are replaced, are replaced by the same variable and that no (occurrences of) two syntactically unequal polymorphic arguments are replaced by the same variable. Let $\mathsf{M} = \{M_1, \ldots, M_n\}$ be the set of variables thus obtained. Write $\Delta$ for the context consisting of declarations in the set $\{\forall M : * \mid M \in \mathsf{M}\}$ (in any order). Let $\Gamma'$ be $\Gamma, \Delta$. Then $\Gamma' \vdash_{\lambda 2} A' : C$. Note that, for a suitable choice of variables, a substitution $\sigma$ is well-typed in $\Gamma$ iff $\sigma$ is well-typed in $\Gamma'$.

Let $B'$ be $\mathrm{nf}(B)$. Let $\Psi(B)$ be the set of all terms $D$ such that $\Gamma' \vdash_{\lambda 2} D : C$ and $D$ is the result of the following operation: for every type in $B'$ that also occurs in $A$ (i.e. the right projection corresponding to this type in $\mathit{Tree}(\Gamma\,;\, B')$ also occurs in $\mathit{Tree}(\Gamma\,;\, A)$, up to a suitable choice of added variables) and is replaced by a variable in the transformation of $A$ into $A'$, replace some (possibly none) occurrences of this type in $B'$ by the same variable. Note that $\Psi(B)$ is a finite set. Also note that when we replace, in two terms of the same type, syntactically equal types by the same variable (not occurring in the original term) and the resulting terms are syntactically equal then the original terms are syntactically equal.

Define

$$\Theta(P) = \{\langle \Gamma'\,;\, A'\,;\, \mathrm{lnf}_{\Gamma'}(D)\rangle \mid D \in \Psi(P)\}.$$

Note that $\Theta(P)$ is a finite set of $\Gamma'$-compatible restricted third-order matching problems for objects in $\lambda 2$.

**Lemma 6.11.** *Let $P = \langle \Gamma\,;\, A\,;\, B\rangle$ be a structured, type-closed third-order matching problem for objects in $\lambda 2$. Then $P$ has a solution $\Leftrightarrow$ some $Q \in \Theta(P)$ has a solution.*

**Proof.** Let $B'$ be $\mathrm{nf}(B)$.

"$\Rightarrow$" Suppose $P$ has solution $\sigma$; then $\sigma(A) \twoheadrightarrow_{\beta\eta} B'$. As remarked above, $\sigma$ is also well-typed in $\Gamma'$ and so the $\beta\eta$-normal form of $\sigma(A')$ exists; let's call it $D$. Since we may assume that $\sigma$ is a $*$-substitution (so does not affect types) we can write $\sigma(A')$ as the result of replacing occurrences of polymorphic arguments in $\sigma(A)$ by variables from $\mathsf{M}$. By the Commutation Lemma (2), we can replace some types in $B'$ by variables from $\mathsf{M}$ to obtain $D$, with each such replaced type syntactically equal to a (replaced) type in $\sigma(A)$ and replaced by the same variable as its counterpart in $\sigma(A)$. Note that, since $\sigma$ is a $*$-substitution and no subterms of substitution terms of $\sigma$ are replaced, each type replaced in $B'$ is in fact syntactically equal to a type replaced in the transformation of $A$ into $A'$ and replaced by the same variable as its counterpart in $A$. Since $D$ is obtained from $B'$, we know that $D$ is closed in $\sigma(\Gamma')$ and that every variable in $FV(D)$ is universal and of closed type in $\Gamma'$. By Lemma 3.7, $\Gamma' \vdash_{\lambda 2} D : C$. We may conclude that $Q = \langle \Gamma'\,;\, A'\,;\, \mathrm{lnf}_{\Gamma'}(D)\rangle \in \Theta(P)$ and $\sigma$ is a solution for $Q$.

"$\Leftarrow$" Suppose some $Q = \langle \Gamma'\,;\, A'\,;\, D\rangle \in \Theta(P)$ has solution $\tau$. Note that variables in $\mathsf{M}$ do not occur in $\tau$. Write $D'$ for $\mathrm{nf}(D)$. We have $\tau(A') \twoheadrightarrow_{\beta\eta} D'$. The substitution $\tau$ is well-typed in $\Gamma$ and so the $\beta\eta$-normal form of $\tau(A)$ exists; let's call it $E$. Since $\tau$ is a $*$-substitution (so does not affect types in $A$) we can write $\tau(A')$ as the result of replacing some occurrences of polymorphic arguments in $\tau(A)$ by variables from $\mathsf{M}$. By the Commutation Lemma (2),

$D'$ is the result of replacing some occurrences of types in $E$ by variables from $\mathsf{M}$, with each such replaced type syntactically equal to a (replaced) type in $\tau(A)$ and replaced by the same variable as its counterpart in $\tau(A)$. Again we have that, since $\tau$ is a $*$-substitution and no subterms of substitution terms of $\tau$ are replaced, each such replaced type is syntactically equal to a type replaced in the transformation of $A$ into $A'$ and replaced by the same variable as its counterpart in $A$ and hence syntactically equal to a type replaced in $B'$ and replaced by the same variable as its counterpart in $B'$. Since $\tau$ does not contain variables from $\mathsf{M}$, $E$ does not contain variables from $\mathsf{M}$. By the injectivity of the replacing operation, we have that $E \equiv B'$. Thus $\tau$ is a solution for $P$. ⊠

The proof of Lemma 6.11 essentially uses the Commutation Lemma. This lemma can only be applied if $P$ is a third-order matching problem. We do not know if (a suitable variant of) Lemma 6.11 can be generalized to higher (finite) orders.

**Corollary 6.12.** *Third-order matching for objects in $\lambda 2$ reduces to restricted third-order matching for objects in $\lambda 2$.*

**Proof.** In Lemma 6.1, Lemma 6.6, Lemma 6.9 and Lemma 6.11 we have given an effective reduction. ⊠

# 7 A simple result on $\eta$-long $\beta$-normal forms

In this section we prove the following useful result (see Corollary 7.10). Consider a simplified third-order matching problem $P = \langle \Gamma\,;\, A\,;\, B\rangle$ for objects in $\lambda 2$ and let $\sigma$ be a $*$-solution for $P$. Then $\sigma$ is well-typed in $\Gamma$ and $\sigma(A) =_{\beta\eta} B$. By Church-Rosser, the second property is equivalent to $\sigma(A) \twoheadrightarrow_{\beta\eta} \mathrm{nf}(B)$. But in fact we have that $\sigma(A) \twoheadrightarrow_{\beta} B$. So to check that a $*$-substitution is a solution for a simplified third-order matching problem for objects in $\lambda 2$, we neither have to do full $\beta\eta$-conversion nor do we have to $\beta\eta$-reduce $B$ to $\mathrm{nf}(B)$ (which would mean losing the property that the term to which $\sigma(A)$ reduces is in LNF).

To prove this result we introduce the notion of *pre-$\eta$-long $\beta$-normal form* (PLNF). This weakened notion of $\eta$-long $\beta$-normal form is such that $\sigma(A)$ will be in PLNF and the property of being in PLNF will be preserved under $\beta$-reduction. In fact, the $\eta$-long $\beta$-normal form of a term $A$ is the $\beta$-normal form of the pre-$\eta$-long $\beta$-normal form of $A$. By inspection of the proof one can verify that in $\lambda\tau$ the result holds for matching problems of arbitrary order. [1] The property does not hold in $\lambda 2$ in general: see Example 7.11, below.

**Definition 7.1.** Suppose $\Gamma \vdash_{\lambda 2} A : B$. We define $\mathrm{plnf}_\Gamma(A)$, the *pre-$\eta$-long $\beta$-normal form of* $A$, by induction on the pair (length of $A$, length of $B$), ordered lexicographically. Write $B$ as $\Pi y_1{:}B_1 \ldots \Pi y_m{:}B_m.B'$, with $B'$ atomic.

- $A \in \mathcal{C}$. Then $\mathrm{plnf}_\Gamma(A) \equiv A$.

[1] The notion PLNF seems to be the same as Gardner's notion "fully applied" in the context of ELF (see [9], p. 91). She has also given a proof of a result similar to Lemma 7.8.

- $A \equiv \Pi x{:}A_1.A_2$. Then $\mathrm{plnf}_\Gamma(A) \equiv A$.
- $A \equiv \lambda x{:}A_1.A_2$. Then $\mathrm{plnf}_\Gamma(A) \equiv \lambda x{:}A_1.\mathrm{plnf}_{\Gamma,\forall x:A_1}(A_2)$
- $A \equiv A_1 C_1 \ldots C_n$ $(n \geq 0)$. Then $\mathrm{plnf}_\Gamma(A) \equiv$

$$\begin{cases} \lambda y_1{:}B_1 \ldots \lambda y_m{:}B_m.xC'_1 \ldots C'_n y'_1 \ldots y'_m & \text{if } A_1 \equiv x \\ \lambda y_1{:}B_1 \ldots \lambda y_m{:}B_m.\mathrm{plnf}_\Gamma(\lambda x{:}A_2.A_3))C'_1 \ldots C'_n y'_1 \ldots y'_m & \text{if } A_1 \equiv \lambda x{:}A_2.A_3 \end{cases}$$

  Here $C'_i \equiv \mathrm{plnf}_\Gamma(C_i)$ (for $1 \leq i \leq n$) and $y'_i \equiv \mathrm{plnf}_{\Gamma,\forall y_i:B_i}(y_i)$ (for $1 \leq i \leq m$).

  The recursive calls on the variables $y_i$ are justified as in Definition 2.6.

We say that $A$ *is in PLNF* if $\mathrm{plnf}_\Gamma(A) \equiv A$.

**Lemma 7.2.** *Suppose* $\Gamma \vdash_{\lambda 2} A : B$. *If* $A$ *is in* LNF, *then* $A$ *is in* PLNF.

**Proof.** By induction on the LNF-structure of $A$. ⊠

**Lemma 7.3.** *Suppose* $\Gamma \vdash_{\lambda 2} A : B$. *If* $A$ *is in* PLNF *and* $\beta$*-normal then* $A$ *is in* LNF.

**Proof.** By induction on the PLNF-structure of $A$. ⊠

**Lemma 7.4.** *Suppose* $\Gamma \vdash_{\lambda 2} A : \Pi y_1{:}B_1 \ldots \Pi y_m{:}B_m.B : *$, *with* $A$ *in* PLNF *and* $B$ *atomic. Then* $A$ *is of one of the two following forms:*

1. $\lambda y_1{:}B_1 \ldots \lambda y_m{:}B_m.xC_1 \ldots C_n$, *with* $C_1, \ldots, C_n$ *in* PLNF $(n \geq 0)$.
2. $\lambda y_1{:}B_1 \ldots \lambda y_m{:}B_m.(\lambda x{:}A_1.A_2)C_1 \ldots C_n$, *with* $\lambda x{:}A_1.A_2$, $C_1, \ldots, C_n$ *in* PLNF $(n \geq 0)$.

**Proof.** By induction on the structure of $A$. ⊠

**Lemma 7.5.** *Suppose* $\Gamma, Qx : D, \Delta \vdash_{\lambda 2} A : B$ *and* $\Gamma \vdash_{\lambda 2} C : D : *$. *If* $A$ *and* $C$ *are in* PLNF, *then* $A[x := C]$ *is in* PLNF.

**Proof.** By induction on the PLNF-structure of $A$. Note that because $x$ is an object variable, all types remain unaffected by substitution of $C$ for $x$. ⊠

**Lemma 7.6.** *Suppose* $\Gamma, Qx : *, \Delta \vdash_{\lambda 2} A : B$ *and* $\Gamma \vdash_{\lambda 2} C : *$, *where* $C$ *is a universal variable in* $\Gamma$. *Suppose that* $A$ *is in* PLNF. *Then* $A[x := C]$ *is in* PLNF.

**Proof.** By induction on the PLNF-structure of $A$. ⊠

**Lemma 7.7.** *Suppose* $\Gamma \vdash_{\lambda 2} A : B$*, where* $A$ *is simple and in* PLNF*. If* $A \twoheadrightarrow_\beta A'$ *then* $A'$ *is in* PLNF*.*

**Proof.** If $B \equiv *$, then $A$ is $\beta\eta$-normal so the result trivially follows. So assume that $B \not\equiv *$. Then $\Gamma \vdash_{\lambda 2} B : *$. Write $B \equiv \Pi y_1{:}B_1 \ldots \Pi y_m{:}B_m.B'$, with $B'$ atomic. The proof proceeds by induction on the generation of $\twoheadrightarrow_\beta$. If $A \equiv A'$ or there exists a term $A''$ such that $A \twoheadrightarrow_\beta A'' \twoheadrightarrow_\beta A'$, then the result follows easily. (Recall that by Lemma 5.6, the property of being simple is preserved under $\beta$-reduction.) For the $\beta$-step, use Lemma 7.6 and Lemma 7.5. (That these lemmas cover all possible cases follows from simplicity of $A$.) Next come the compatibility cases. By Lemma 7.4, $A$ can be of two forms. If $A \equiv \lambda y_1{:}B_1 \ldots \lambda y_m{:}B_m.xC_1 \ldots C_n$ and for some $1 \leq i \leq n$, $C_i \to_\beta C_i'$, then the result follows from the induction hypothesis. Next, assume $A \equiv \lambda y_1{:}B_1 \ldots \lambda y_m{:}B_m.(\lambda x{:}A_1.A_2)C_1 \ldots C_n$. Again, if the contraction of a redex occurs in $A_2$ or in some term $C_i$ $(1 \leq i \leq n)$, then the result follows from the induction hypothesis. So assume $A' \equiv \lambda y_1{:}B_1 \ldots \lambda y_m{:}B_m.(A_2[x := C_1])C_2 \ldots C_n$. If $n = 1$ then we have already treated this case (the $\beta$-step). So assume $n > 1$. By Lemma 7.6 and Lemma 7.5, $A_2[x := C_1]$ is in PLNF. Since $n > 1$ and $\lambda x{:}A_1.A_2$ is in PLNF, $A_2$ and hence $A_2[x := C_1]$ is an abstraction term. By inspecting the definition of plnf_$(-)$, one immediately verifies that $A'$ is in PLNF. ⊠

**Lemma 7.8.** *Suppose* $\Gamma \vdash_{\lambda 2} A : B$*, where* $A$ *is simple and in* PLNF*. The* $\beta$*-normal form of* $A$ *is in* LNF*.*

**Proof.** By Lemma 7.7, the $\beta$-normal form of $A'$ is in PLNF. By Lemma 7.3, it is in LNF. ⊠

**Lemma 7.9.** *Suppose* $\Gamma \vdash_{\lambda 2} A : C$ *and* $\Gamma \vdash_{\lambda 2} B : C$*. Suppose that* $\Gamma$ *contains (pairwise distinct) declarations* $Q_i x_i : C_i$ $(1 \leq i \leq n)$ *and that, for all* $1 \leq i \leq n$*,* $\Gamma_{x_i} \vdash_{\lambda 2} D_i : C_i : *$*,* $ord_{\Gamma_{x_i}}(C_i) \leq 3$ *and for all* $1 \leq j \leq n$*,* $x_i \notin FV(D_j)$*. Assume that all terms mentioned above are in* LNF *and that* $A$ *is simplified w.r.t.* $\{x_1, \ldots, x_n\}$*. If* $A[\vec{x} := \vec{D}] =_{\beta\eta} B$ *then* $A[\vec{x} := \vec{D}] \twoheadrightarrow_\beta B$*.*

**Proof.** It is not difficult to check that $\Gamma \setminus \{x_1, \ldots, x_n\} \vdash_{\lambda 2} A[\vec{x} := \vec{D}] : C$. By Lemma 7.5, $A[\vec{x} := \vec{D}]$ is in PLNF. By Lemma 5.8, $A[\vec{x} := \vec{D}]$ is simple. By Lemma 7.8, we know that the $\beta$-normal form of $A[\vec{x} := \vec{D}]$ is in LNF. Since LNFs are unique the result follows. ⊠

**Corollary 7.10.** *Let* $P = \langle \Gamma\,;\, A\,;\, B \rangle$ *be a simplified third-order matching problem in* $\lambda 2$*. Let* $\sigma$ *be a* $*$*-solution for* $P$*. Then* $\sigma(A) \twoheadrightarrow_\beta B$*.*

**Proof.** By Lemma 7.9. ⊠

We end this section with an example that shows that in Lemma 7.9 we cannot omit the condition that $A$ is simplified. Note that the same example is discussed in Section 4.

**Example 7.11.** Let $\Gamma \equiv \langle \forall C : *, \forall D : *, \forall x : \Pi\alpha{:}*.(\alpha{\to}\alpha){\to}D, \exists f : *{\to}D\rangle$. Let $A \equiv f(C{\to}D)$ and $S \equiv \lambda\alpha{:}*.x\alpha(\lambda y{:}\alpha.y)$. Then $\Gamma \vdash_{\lambda 2} A : D$ and $\Gamma_f \vdash_{\lambda 2} S : *{\to}D$. Note that $ord_\Gamma(*{\to}D) = 3$ and $A$ and $S$ are in LNF, but that $A$ is not simplified. Now $A[f := S]$ $\beta$-reduces in one step to $x(C{\to}D)(\lambda y{:}C{\to}D.y)$, a term which is $\beta$-normal but not in LNF. To reach LNF, an $\eta$-expansion is needed to $x(C{\to}D)(\lambda y{:}C{\to}D\lambda z{:}C.yz)$.

# 8 Encoding restricted third-order matching problems

When a restricted third-order matching problem $P = \langle \Gamma\,;\, A\,;\, B\rangle$ for objects has a solution $\sigma$ we may assume that $\sigma$ is a $*$-solution and hence that $\sigma(A)$ and all $\beta$-reducts of $\sigma(A)$ are simple. This will allow us to encode restricted third-order matching problems in $\lambda 2$ (and their solutions) as third-order matching problems in $\lambda\tau$ (and their solutions).

To this end, we define a translation $[\![ - ]\!]_-$ from terms in $\lambda 2$ to terms in $\lambda\tau$. We will show that it preserves judgements on finite-redexed terms, $\beta$-reduction on simple terms, the property of being a matching problem of finite order, the property of being a solution for a matching problem for types and the property of being a $*$-solution for a simplified third-order matching problem for objects.

We assume the existence of a function $\mathsf{P}$ that assigns to each $\lambda\tau$-type $A$ a variable $p_A$ of type $A$. We will assume that all these variables are mutually distinct. Moreover we will assume that when we translate terms in $\lambda 2$ to terms in $\lambda\tau$ the new variables $p_D$ are distinct from all variables that occur in the $\lambda 2$-terms. We begin with a translation of types in $\lambda 2$ to types in $\lambda\tau$.

**Definition 8.1.** Suppose $\Gamma \vdash_{\lambda 2} A : s$. We define $\ll A \gg_\Gamma$ as follows.

$$\ll A \gg_\Gamma \;=\; \begin{cases} O & \text{if } A \equiv * \\ \ll A_1 \gg_\Gamma \to \ll A_2 \gg_{\Gamma, \forall x:A_1} & \text{if } A \equiv \Pi x{:}A_1.A_2 \\ O & \text{otherwise} \end{cases}$$

Of course, if $\Delta \supseteq \Gamma$ is legal in $\lambda 2$, then $\ll A \gg_\Gamma \equiv \ll A \gg_\Delta$.

We first define $[\![\ ]\!]$ on legal contexts.

**Definition 8.2.** Let $\Gamma$ be legal in $\lambda 2$. We define $[\![\,\Gamma\,]\!]$ by induction on the length of $\Gamma$. We use fresh variables $c^\Pi$, $c^\to$.

$$\begin{aligned} [\![\,\langle\rangle\,]\!] &= \langle c^\Pi : (O{\to}O){\to}O, c^\to : O{\to}O{\to}O\rangle \\ [\![\,\Gamma, Qx : A\,]\!] &= [\![\,\Gamma\,]\!], Qx : \ll A \gg_\Gamma \end{aligned}$$

**Lemma 8.3.** *If $\Gamma \vdash_{\lambda 2} A : s$, then $[\![\,\Gamma\,]\!] \vdash_{\lambda\tau} \ll A \gg_\Gamma : *$.*

**Proof.** By induction on the derivation of $\Gamma \vdash_{\lambda 2} A : s$, using Strengthening for the case of the Product Rule. ⊠

**Lemma 8.4.** *Suppose $\Gamma \vdash_{\lambda 2} A : s$. We have $ord_{[\![\,\Gamma\,]\!]}(\ll A \gg_\Gamma) \leq ord_\Gamma(A)$.*

**Proof.** By Lemma 8.3, $ord_{[\![\,\Gamma\,]\!]}(\ll A \gg_\Gamma)$ is defined. The proof proceeds by induction on the structure of $A$, using Lemma 2.5. ⊠

The following technical lemma will be used in the proof of Lemma 8.14.

**Lemma 8.5.** *Suppose* $\Gamma, Qx : *, \Delta \vdash_{\lambda 2} A : *$ *and* $\Gamma \vdash_{\lambda 2} B : *$*, where* $B$ *is atomic, i.e.* $B$ *is a variable. Then* $\ll A[x := B] \gg_{\Gamma,\Delta} \equiv \ll A \gg_{\Xi}$.

**Proof.** By induction on the structure of $A$, using Lemma 2.5. ⊠

We proceed to define $[\![ - ]\!]_-$ on finite-redexed objects. This function returns pairs of the form $([\![ A ]\!]^1_\Gamma \mid [\![ A ]\!]^2_\Gamma)$, where $[\![ A ]\!]^1_\Gamma$ is a term and $[\![ A ]\!]^2_\Gamma$ is a context containing declarations of variables that are added in the construction of $[\![ A ]\!]^1_\Gamma$ (we use a slightly non-standard comma).

**Definition 8.6.** Suppose $\Gamma \vdash_{\lambda 2} A : B : s$, where $A$ is finite-redexed. We define $[\![ A ]\!]_\Gamma$ by induction on the structure of $A$.

$$
\begin{array}{rcl}
[\![ x ]\!]_\Gamma & = & (x \parallel \langle\rangle) \\
[\![ A_1 A_2 ]\!]_\Gamma & = & \left\{ \begin{array}{l} (p_D y' [\![ C_1 ]\!]^1_\Gamma \ldots [\![ C_n ]\!]^1_\Gamma [\![ A_2 ]\!]^1_\Gamma \mid \\ \langle \forall p_D : D \rangle \sqcup [\![ C_1 ]\!]^2_\Gamma \sqcup \ldots \sqcup [\![ C_n ]\!]^2_\Gamma \sqcup [\![ A_2 ]\!]^2_\Gamma) \\ \text{if } A_1 \equiv y C_1 \ldots C_n, \text{ with } y \text{ a PV} \\ \\ ([\![ A_1 ]\!]^1_\Gamma [\![ A_2 ]\!]^1_\Gamma \parallel [\![ A_1 ]\!]^2_\Gamma \sqcup [\![ A_2 ]\!]^2_\Gamma) \\ \text{if } A_1 \text{ does not start with a PV} \end{array} \right. \qquad (\dagger) \\
[\![ \lambda x{:}A_1.A_2 ]\!]_\Gamma & = & (\lambda x{:} \ll A_1 \gg_\Gamma . [\![ A_2 ]\!]^1_{\Gamma, \forall x:A_1} \mid [\![ A_2 ]\!]^2_{\Gamma, \forall x:A_1}) \\
[\![ \Pi x{:}A_1.A_2 ]\!]_\Gamma & = & \left\{ \begin{array}{ll} (c^\Pi (\lambda x{:}O.[\![ A_2 ]\!]^1_{\Gamma, \forall x:A_1}) \mid \langle\rangle) & \text{if } x \in FV(A_2) \\ (c^\rightarrow [\![ A_1 ]\!]^1_\Gamma [\![ A_2 ]\!]^1_\Gamma \mid \langle\rangle) & \text{otherwise} \end{array} \right.
\end{array}
$$

(†): $y' \equiv \mathrm{Inf}_{[\![ \Gamma ]\!]}(y)$ and $D \equiv \ll \Gamma(y) \gg_\Gamma \rightarrow$
$\ll \tau(\Gamma\,;\, C_1) \gg_\Gamma \rightarrow \cdots \rightarrow \ll \tau(\Gamma\,;\, C_n) \gg_\Gamma \rightarrow \ll \tau(\Gamma\,;\, A_2 \gg_\Gamma \rightarrow \ll \tau(\Gamma\,;\, A_1 A_2) \gg_\Gamma$.

Let the *size of a term* $A$ be the number of symbols in $A$. Suppose $\Gamma \vdash_{\lambda 2} B : C$. It is easy to see that there exists a linear function (which we denote by $\mathsf{F}$) such that if $m$ is the size of $[\![ B ]\!]^1_\Gamma$ then the size of $B$ is bounded by $\mathsf{F}(m)$. This will be used in the proof of Theorem 9.6.

Our next aim is to prove that $[\![ - ]\!]_-$ preserves judgements on finite-redexed terms. We need some auxiliary results.

**Lemma 8.7.** *Suppose* $\Gamma \vdash_{\lambda 2} A : B : s$*, where* $A$ *is finite-redexed. Suppose* $\Delta \supseteq \Gamma$ *is legal in* $\lambda 2$*. Then* $[\![ A ]\!]_\Gamma = [\![ A ]\!]_\Delta$.

**Proof.** By induction on the structure of $A$. ⊠

**Lemma 8.8.** *Suppose* $\Gamma \vdash_{\lambda 2} A : *$*. Then* $[\![ A ]\!]^2_\Gamma \equiv \langle\rangle$.

**Proof.** By induction on the structure of $A$, using Lemma 2.5. (Note that $A$ is trivially finite-redexed.) ⊠

**Lemma 8.9.** *Suppose $\Gamma \vdash_{\lambda 2} A : B$, where $A$ is finite-redexed and not an abstraction term. If $B$ is polymorphic, then $A$ is of the form $xA_1 \ldots A_n$, for some terms $A_1, \ldots, A_n$ $(n \geq 0)$ and some $x$ such that $\Gamma(x)$ is polymorphic.*

**Proof.** We can write $A \equiv A' A_1 \ldots A_n$, where $n \geq 0$ and $A'$ is not an application term. Thus either $A'$ is a variable $x$ and the result easily follows, or $A'$ is an abstraction term. But in this case, because $A$ is finite-redexed the type of $A'$ is not polymorphic and *a fortiori* the type of $A$ is not polymorphic, contradicting the assumption. ⊠

**Lemma 8.10.** *Suppose $\Gamma \vdash_{\lambda 2} A : B : s$, where $A$ is finite-redexed. Then*

$$[\![ A ]\!]^2_\Gamma, [\![ \Gamma ]\!] \vdash_{\lambda\tau} [\![ A ]\!]^1_\Gamma : \ll B \gg_\Gamma .$$

**Proof.** By induction on the structure of $A$. In most cases the induction is straightforward. If $A \equiv A_1 A_2$ and $A_1$ starts with PV $y$, use the fact that the type of $\mathrm{lnf}_{[\![ \Gamma ]\!]}(y)$ equals the type of $y$ in $[\![ \Gamma ]\!]$. If $A \equiv \Pi x{:}A_1.A_2$, use Lemma 8.8. Also remember that if $x \in FV(A_2)$, then $A_1 \equiv *$, so $\ll A_1 \gg_\Gamma \equiv O$. The only case worth considering in detail is the case where $A \equiv A_1 A_2$ and $A_1$ does not start with a PV. By Lemma 2.5 (6), there exist terms $C_1$, $C_2$ such that $\Gamma \vdash_{\lambda 2} A_1 : \Pi x{:}C_1.C_2$, $\Gamma \vdash_{\lambda 2} A_2 : C_1$ and $C_2[x := A_2] \equiv B$. By induction hypothesis, $[\![ A_1 ]\!]^2_\Gamma, [\![ \Gamma ]\!] \vdash_{\lambda\tau} [\![ A_1 ]\!]^1_\Gamma : \ll \Pi x{:}C_1.C_2 \gg_\Gamma$ and $[\![ A_2 ]\!]^2_\Gamma, [\![ \Gamma ]\!] \vdash_{\lambda\tau} [\![ A_2 ]\!]^1_\Gamma : \ll C_1 \gg_\Gamma$. Of course we also have $[\![ A_1 ]\!]^2_\Gamma \sqcup [\![ A_2 ]\!]^2_\Gamma, [\![ \Gamma ]\!] \vdash_{\lambda\tau} [\![ A_1 ]\!]^1_\Gamma : \ll \Pi x{:}C_1.C_2 \gg_\Gamma$ and $[\![ A_1 ]\!]^2_\Gamma \sqcup [\![ A_2 ]\!]^2_\Gamma, [\![ \Gamma ]\!] \vdash_{\lambda\tau} [\![ A_2 ]\!]^1_\Gamma : \ll C_1 \gg_\Gamma$. Since $A_1$ is simple and does not start with a PV, we can use Lemma 8.9 to infer that the type of $A_1$ is not polymorphic. Hence $\Pi x{:}C_1.C_2 \equiv C_1 \to C_2$ and $C_2[x := A_2] \equiv C_2 \equiv B$. Moreover, $\ll \Pi x{:}C_1.C_2 \gg_\Gamma \equiv \ll C_1 \to C_2 \gg_\Gamma \equiv \ll C_1 \gg_\Gamma \to \ll C_2 \gg_{\Gamma, \forall x:C_1} \equiv \ll C_1 \gg_\Gamma \to \ll C_2 \gg_\Gamma$. We are done. ⊠

**Lemma 8.11.** *Suppose $\Gamma \vdash_{\lambda 2} A : B : s$, where $A$ is finite-redexed. If $A$ is normal (resp. in* LNF*) then $[\![ A ]\!]^1_\Gamma$ is normal (resp. in* LNF*).*

**Proof.** By induction on the structure of $A$. ⊠

**Definition 8.12.** Let $P = \langle \Gamma \,;\, A \,;\, B \rangle$ be a matching problem of finite order $n$ in $\lambda 2$. Define

$$[\![ P ]\!] = \langle [\![ A ]\!]^2_\Gamma \sqcup [\![ B ]\!]^2_\Gamma, [\![ \Gamma ]\!] \,;\, [\![ A ]\!]^1_\Gamma \,;\, [\![ B ]\!]^1_\Gamma \rangle.$$

By Lemma 8.10, Lemma 8.11 and Lemma 8.4, $[\![ P ]\!]$ is a matching problem of order $n$ in $\lambda\tau$.

Note that if $P = \langle \Gamma \,;\, A \,;\, B \rangle$ is a matching problem for types then we may assume that $\Gamma$ contains no object variables, since, by Lemma 2.5 (3), these variables do not occur in $A$ or in $B$ and need not occur in substitution terms of solutions for $P$. So $[\![ P ]\!]$ is a first-order matching problem and if $[\![ P ]\!]$ has a solution, it has a unique solution.

Now that we have proved that the translation maps matching problems of finite order $n$ in $\lambda 2$ to matching problems of order $n$ in $\lambda\tau$, we proceed to show that the translation preserves $*$-solutions to simplified third-order matching problems. Among other things we have to show that the translation preserves $\beta$-reduction for simple terms. Since simple terms may contain two kinds of redexes we need two substitution lemmas.

**Lemma 8.13.** *Suppose* $\overbrace{\Gamma, Qx : D, \Delta}^{\Xi} \vdash_{\lambda 2} A : B : s$ *and* $\Gamma \vdash_{\lambda 2} C : D : *$, *where* $A$ *and* $C$ *are finite-redexed and* $D$ *is not polymorphic. Then*

$$[\![ A[x := C] ]\!]^1_{\Gamma,\Delta[x:=C]} \equiv [\![ A ]\!]^1_{\Xi}[x := [\![ C ]\!]^1_{\Gamma}].$$

**Proof.** By induction on the structure of $A$. Because $x$ is an object variable, $\Delta[x := C] \equiv \Delta$ and $B[x := C] \equiv B$.

- $A \equiv y$. Easy.
- $A \equiv \lambda y{:}A_1.A_2$. This case follows easily from the induction hypothesis.
- $A \equiv \Pi y{:}A_1.A_2$. Easy because $x \notin FV(A)$ and $x \notin \{c^{\rightarrow}, c^{\Pi}\}$.
- $A \equiv A_1A_2$. If $A_1$ starts with a PV the claim follows from the induction hypothesis plus the fact that substitution of objects does not affect types. If $(A_1A_2)[x := C]$ does not start with a PV then $A_1A_2$ does not begin with a PV and the result follows from the induction hypothesis. Suppose that $A_1A_2$ does not start with a PV, but that $(A_1A_2)[x := C]$ does start with a PV. Then $A_1 \equiv xA_1'$ and $C$ starts with a PV. Thus $[\![ A_1A_2 ]\!]^1_{\Xi} \equiv x[\![ A_1' ]\!]^1_{\Xi}[\![ A_2 ]\!]^1_{\Xi}$. Since $D$ is not polymorphic we can write $D \equiv \tau(A_1' ; \Xi) \rightarrow \tau(A_2 ; \Xi) \rightarrow B \equiv \tau(A_1' ; \Xi) \rightarrow \tau(A_2 ; \Xi) \rightarrow B_1 \rightarrow \cdots \rightarrow B_n \rightarrow B'$. Write $C \equiv yC_1 \ldots C_m$. Then $[\![ C ]\!]^1_{\Gamma} \equiv p_E \mathrm{lnf}_{[\![ \Gamma ]\!]}(y)[\![ C_1 ]\!]^1_{\Gamma} \ldots [\![ C_m ]\!]^1_{\Gamma}$, where
$E \equiv \ll \Gamma(y) \gg_{\Gamma} \rightarrow \ll \tau(\Gamma ; C_1) \gg_{\Gamma} \rightarrow \cdots \rightarrow \ll \tau(\Gamma ; C_m) \gg_{\Gamma} \rightarrow \ll D \gg_{\Gamma}$.

  Note that $([\![ \Gamma, \Delta ]\!])(y) \equiv \ll (\Gamma, \Delta)(y) \gg_{\Gamma,\Delta} \equiv \ll \Gamma(y) \gg_{\Gamma} \equiv ([\![ \Gamma ]\!])(y)$. So $\mathrm{lnf}_{[\![ \Gamma,\Delta ]\!]}(y) \equiv \mathrm{lnf}_{[\![ \Gamma ]\!]}(y)$. Also $\ll B \gg_{\Gamma} \equiv \ll B \gg_{\Gamma,\Delta}$ and, for all $1 \leq i \leq m$, $\ll \tau(\Gamma ; C_i) \gg_{\Gamma} \equiv \ll \tau(\Gamma, \Delta ; C_i) \gg_{\Gamma,\Delta}$. For $T \in \mathcal{T}$, write $T^* \equiv T[x := C]$. Then we have
$\ll \tau(\Gamma, \Delta ; A_1'^*) \gg_{\Gamma,\Delta} \equiv \ll \tau(\Xi ; A_1') \gg_{\Gamma,\Delta} \equiv \ll \tau(\Xi ; A_1') \gg_{\Gamma}$ (and similarly for $A_2$). We may conclude that we write

$$\begin{aligned} F \equiv\ & \ll (\Gamma, \Delta)(y) \gg_{\Gamma,\Delta} \rightarrow \\ & \ll \tau(\Gamma, \Delta ; C_1) \gg_{\Gamma,\Delta} \rightarrow \cdots \rightarrow \ll \tau(\Gamma, \Delta ; C_m) \gg_{\Gamma,\Delta} \rightarrow \\ & \ll \tau(\Gamma, \Delta, A_1'^*) \gg_{\Gamma,\Delta} \\ & \ll \tau(\Gamma, \Delta ; A_2^*) \gg_{\Gamma,\Delta} \rightarrow \\ & \ll B \gg_{\Gamma,\Delta} \end{aligned}$$

  then we have $F \equiv E$. By Lemma 8.7 we have, for all $1 \leq i \leq m$, $[\![ C_i ]\!]^1_{\Gamma} \equiv [\![ C_i ]\!]^1_{\Gamma,\Delta}$. We compute:

$$\begin{aligned} & [\![ (A_1A_2)^* ]\!]^1_{\Gamma,\Delta} \equiv \\ & [\![ (xA_1'A_2)^* ]\!]^1_{\Gamma,\Delta} \equiv \\ & [\![ CA_1'^*A_2^* ]\!]^1_{\Gamma,\Delta} \equiv \\ & p_F(\mathrm{lnf}_{[\![ \Gamma,\Delta ]\!]}(y))[\![ C_1 ]\!]^1_{\Gamma,\Delta} \ldots [\![ C_m ]\!]^1_{\Gamma,\Delta}[\![ A_1'^* ]\!]^1_{\Gamma,\Delta}[\![ A_2^* ]\!]^1_{\Gamma,\Delta} \equiv \\ & p_E(\mathrm{lnf}_{[\![ \Gamma ]\!]}(y))[\![ C_1 ]\!]^1_{\Gamma} \ldots [\![ C_m ]\!]^1_{\Gamma}[\![ A_1'^* ]\!]^1_{\Gamma,\Delta}[\![ A_2^* ]\!]^1_{\Gamma,\Delta} \equiv \qquad \text{(IH)} \\ & p_E(\mathrm{lnf}_{[\![ \Gamma ]\!]}(y))[\![ C_1 ]\!]^1_{\Gamma} \ldots [\![ C_m ]\!]^1_{\Gamma}([\![ A_1' ]\!]^1_{\Xi}[x := [\![ C ]\!]^1_{\Gamma}])([\![ A_2 ]\!]^1_{\Xi}[x := [\![ C ]\!]^1_{\Gamma}]) \equiv \\ & ([\![ xA_1'A_2 ]\!]^1_{\Xi})[x := [\![ C ]\!]^1_{\Gamma}], \end{aligned}$$

  as required.

⊠

**Lemma 8.14.** *Suppose* $\overbrace{\Gamma, Qx : *, \Delta}^{\Xi} \vdash_{\lambda 2} A : B : s$ *and* $\Gamma \vdash_{\lambda 2} C : *$*, where* $A$ *is finite-redexed and* $C$ *is a universal variable in* $\Gamma$*. Then*

$$[\![ A[x := C] ]\!]^1_{\Gamma,\Delta} \equiv [\![ A ]\!]^1_{\Xi}[x := [\![ C ]\!]^1_{\Gamma}].$$

**Proof.** By induction on the structure of $A$.

- $A \equiv y$. Easy.
- $A \equiv \lambda x{:}A_1.A_2$. By Lemma 8.5, $\ll A_1[x := C] \gg_{\Gamma,\Delta} \equiv \ll A_1 \gg_{\Xi}$. So the result follows easily from the induction hypothesis.
- $A_1A_2$. Since $C$ is atomic we know that every variable declared in $\Xi$ has a polymorphic type in $\Xi$ iff it has a polymorphic type in $\Gamma, \Delta[x := C]$. Thus $A_1$ starts with a PV in $\Xi$ iff $A_1[x := C]$ starts with a PV in $\Gamma, \Delta[x := C]$ (of course, neither $x$ nor $C$ is a polymorphic variable). Assume that $A_1$ starts with a PV; the remaining case is treated similarly. Write $A \equiv yC_1 \ldots C_n$. The result is a consequence of the following facts. By Lemma 8.5, $\ll \Xi(y) \gg_{\Xi} \equiv \ll (\Xi(y))[x := C] \gg_{\Gamma,\Delta} \equiv \ll (\Gamma, \Delta[x := C])(y) \gg_{\Gamma,\Delta}$. So $\mathrm{lnf}_{[\![ \Xi ]\!]}(y) \equiv \mathrm{lnf}_{[\![ \Gamma,\Delta[x:=C] ]\!]}(y)$. By the same lemma and the Substitution Lemma we get that for all $1 \leq i \leq n$, $\ll \tau(\Gamma, \Delta\,;\, C_i[x := C]) \gg_{\Gamma,\Delta} \equiv \ll \tau(\Xi\,;\, C_i) \gg_{\Xi}$. Another application of Lemma 8.5 yields that $\ll B \gg_{\Xi} \equiv \ll B[x := C] \gg_{\Gamma,\Delta}$. Finally, we know by the induction hypothesis that, for all $1 \leq i \leq n$, $[\![ C_i[x := C] ]\!]^1_{\Gamma,\Delta[x:=C]} \equiv [\![ C_i ]\!]^1_{\Xi}[x := [\![ C ]\!]^1_{\Gamma}]$.
- $A \equiv \Pi y{:}A_1.A_2$. Note that $y \in FV(A_2)$ iff $y \in FV(A_2[x := C])$. For simplicity, assume that $y \notin FV(A_2)$; the case where $y \in FV(A_2)$ is treated similarly. Then

$$\begin{array}{ll}
[\![ (\Pi y{:}A_1.A_2)[x := C] ]\!]^1_{\Gamma,\Delta} \equiv & \\
[\![ \Pi y{:}(A_1[x := C]).(A_2[x := C]) ]\!]^1_{\Gamma,\Delta} \equiv & \\
c^{\rightarrow}[\![ A_1[x := C] ]\!]^1_{\Gamma,\Delta}[\![ A_2[x := C] ]\!]^1_{\Gamma,\Delta} \equiv & \text{(IH)} \\
c^{\rightarrow}([\![ A_1 ]\!]^1_{\Xi}[x := [\![ C ]\!]^1_{\Gamma}])([\![ A_2 ]\!]^1_{\Xi}[x := [\![ C ]\!]^1_{\Gamma}]) \equiv & (x \notin \{c^{\rightarrow}, c^{\Pi}\}) \\
(c^{\rightarrow}[\![ A_1 ]\!]^1_{\Xi}[\![ A_2 ]\!]^1_{\Xi})[x := [\![ C ]\!]^1_{\Gamma}] \equiv & \\
([\![ \Pi y{:}A_1.A_2 ]\!]^1_{\Xi})[x := [\![ C ]\!]^1_{\Gamma}]. &
\end{array}$$

⊠

**Lemma 8.15.** *Suppose* $\Gamma \vdash_{\lambda 2} A : B : s$*, where* $A$ *is simple. If* $A \twoheadrightarrow_{\beta} A'$*, then* $[\![ A ]\!]^1_{\Gamma} \twoheadrightarrow_{\beta} [\![ A' ]\!]^1_{\Gamma}$.

**Proof.** By induction on the generation of $\twoheadrightarrow_{\beta}$.

- $A \equiv A'$: trivial.
- There exists a term $C$ such that $A \twoheadrightarrow_{\beta} C$ and $C \twoheadrightarrow_{\beta} A'$. By Lemma 5.6, $C$ is simple. Thus the result follows from the induction hypothesis.
- $A \rightarrow_{\beta} A'$. We treat the two interesting cases.

– The $\beta$-step: $A \equiv (\lambda x{:}A_1.A_2)A_3$ and $A' \equiv A_2[x := A_3]$. Then

$$
\begin{array}{ll}
[\![(\lambda x{:}A_1.A_2)A_3]\!]^1_\Gamma \equiv & \\
[\![(\lambda x{:}A_1.A_2)]\!]^1_\Gamma [\![A_3]\!]^1_\Gamma \equiv & \\
(\lambda x{:} \ll A_1 \gg_\Gamma . [\![A_2]\!]^1_{\Gamma,\forall x:A_1})[\![A_3]\!]^1_\Gamma \to_\beta & \\
([\![A_2]\!]^1_{\Gamma,\forall x:A_1})[x := [\![A_3]\!]^1_\Gamma] \equiv & (\dagger) \\
[\![A_2[x := A_3]]\!]^1_\Gamma &
\end{array}
$$

($\dagger$): If $A_1 \not\equiv *$, use Lemma 8.13. Since $A$ is finite-redexed, the type of $A_3$ is not polymorphic. If $A_1 \equiv *$, use Lemma 8.14. Since $A$ is APR, $A_3$ is a universal variable in $\Gamma$.

– $A \equiv A_1A_2$, $A' \equiv A_1A_2'$ and $A_2 \to_\beta A_2'$. In most cases, the result follows immediately from the induction hypothesis. If $A_1$ starts with a PV, then one also needs that, by Subject Reduction, $A_2'$ has the same type as $A_2$.

– $A \equiv A_1A_2$, $A' \equiv A_1'A_2$ and $A_1 \to_\beta A_1'$. The interesting subcase is where $A_1 \equiv (\lambda x{:}C_1.C_2)C_3$, $A_1' \equiv C_2[x := C_3]$ and $A_1'$ starts with a PV. There are two subsubcases to consider: the case where $C_2$ starts with a PV and the case where $C_2$ is of the form $xE_1 \ldots E_m$ (for some $m \geq 0$) and $C_3$ begins with a PV. We only consider the first case; the second case is treated in a similar way. Write $C_2 \equiv yD_1 \ldots D_n$, with $y$ a PV in $\Gamma$ (by simplicity of $A$, $y \not\equiv x$). For $T \in \mathcal{T}$, write $T^* \equiv T[x := C_3]$. We have:

$$
\begin{array}{ll}
[\![(\lambda x{:}C_1.yD_1 \ldots D_n)C_3A_2]\!]^1_\Gamma \equiv & \\
(\lambda x{:} \ll C_1 \gg_\Gamma . [\![yD_1 \ldots D_n]\!]^1_{\Gamma,\forall x:C_1})[\![C_3]\!]^1_\Gamma [\![A_2]\!]_\Gamma \to_\beta & \\
(([\![yD_1 \ldots D_n]\!]^1_{\Gamma,\forall x:C_1})[x := [\![C_3]\!]^1_\Gamma])[\![A_2]\!]^1_\Gamma \equiv & (\star) \\
([\![yD_1 \ldots D_n]\!]^1_{\Gamma,\forall x:C_1}[\![A_2]\!]^1_\Gamma)[x := [\![C_3]\!]^1_\Gamma] \equiv & (\maltese) \\
([\![yD_1 \ldots D_n]\!]^1_{\Gamma,\forall x:C_1}[\![A_2]\!]^1_{\Gamma,\forall x:C_1})[x := [\![C_3]\!]^1_\Gamma] \equiv & (\dagger) \\
([\![yD_1 \ldots D_nA_2]\!]^1_{\Gamma,\forall x:C_1})[x := [\![C_3]\!]^1_\Gamma] \equiv & (\ddagger) \\
[\![(yD_1 \ldots D_n)^*A_2^*]\!]^1_\Gamma \equiv & (\star) \\
[\![(yD_1 \ldots D_n)^*A_2]\!]^1_\Gamma. &
\end{array}
$$

($\star$) Note that $x \notin FV(A_2)$ and $x \notin FV([\![A_2]\!]^1_\Gamma)$.

($\maltese$): By Lemma 8.7.

($\dagger$): We need to check that $yD_1 \ldots D_nA_2$ is typable in $\Gamma, \forall x : C_1$. By Subject Reduction and the fact that $(\lambda x{:}C_1.yD_1 \ldots D_n)C_3A_2$ has type $B$ in $\Gamma$, we know that $yD_1^* \ldots D_n^*A_2$ has type $B$ in $\Gamma$. By Lemma 5.2, we know that $yD_1 \ldots D_n$ has the same type in $\Gamma, \forall x : C_1$ as $yD_1^* \ldots D_n^*$ in $\Gamma$. So $yD_1 \ldots D_nA_2$ is typable in $\Gamma, \forall x : C_1$.

($\ddagger$): By Lemma 8.13 and Lemma 8.14. That the conditions of these lemmas are satisfied is verified as before.

$\boxtimes$

Next we extend the $[\![-]\!]_-$ translation to substitutions. In principle this is straightforward: we just apply the translation to substitution contexts and substitution terms. Some care is needed, though. Consider a context $\Gamma$, legal in $\lambda 2$, and a substitution $\sigma$, well-typed in $\Gamma$.

While translating a substitution term of a triple in $\sigma$ we may have to add auxiliary variables $p_D$ to the substitution context of that triple. Because our function $\mathsf{P}$ produces precisely one variable for each $\lambda\tau$-type, such a variable should be added to a substitution context at most once in the process of translating $\sigma$ as a whole. Otherwise the application of the translated substitution to $[\![\,\Gamma\,]\!]$ yields an illegal context; namely a context which contains two declarations of a single variable. Thus the translated substitution would not be well-typed in $[\![\,\Gamma\,]\!]$. An additional problem arises when we want to translate a solution $\sigma$ for a matching problem $\langle\Gamma\,;\,A\,;\,B\rangle$. We must take care that the variables added to the substitution contexts are not already added to $[\![\,\Gamma\,]\!]$ in the construction of $[\![A\,]\!]^1_\Gamma$ and $[\![B\,]\!]^1_\Gamma$.

**Definition 8.16.** Let $P = \langle\Gamma\,;\,A\,;\,B\rangle$ be a matching problem of finite order in $\lambda 2$. Let $\sigma$ be a substitution, well-typed in $\Gamma$, and $dom(\sigma) \subseteq dom(\Gamma)$. We define $[\![\sigma\,]\!]_P$ as follows. First we define an ordering on $dom(\Gamma)$: for $x, y \in dom(\Gamma)$ we say that $x <_\Gamma y$ iff $x \in dom(\Gamma_y)$. In other words, $x <_\Gamma y$ if $x$ is declared to the left of $y$ in $\Gamma$. This induces an ordering on $dom(\sigma)$. Let $\langle x\,;\,\gamma\,;\,S\rangle$ be a triple in $\sigma$. We translate this triple to a triple $\langle x\,;\,\gamma', \gamma''\,;\,S'\rangle$, and we put $EV(x) = dom(\gamma'')$, where

- $S' \equiv [\![S\,]\!]^1_{\sigma(\Gamma_x),\gamma}$. By Lemma 8.11, $S'$ is in LNF.
- $\gamma'$ consist of declarations in the set $\{\exists y : \ll T \gg_{\sigma(\Gamma_x),\gamma_y} \mid \exists y : T \in \gamma\}$ ordered by $x <_{\gamma'} y$ iff $x <_\gamma y$.
- $\gamma'' \equiv Ex([\![S\,]\!]^2_{\sigma(\Gamma_x),\gamma} \setminus (dom([\![A\,]\!]^2_\Gamma) \cup dom([\![B\,]\!]^2_\Gamma) \cup \bigcup_{z <_\Gamma x} EV(z)))$.

  By Lemma 8.8, $\gamma'' \equiv \langle\rangle$ if $\Gamma(x) \equiv *$.

Finally we define $[\![\sigma\,]\!]_P$ as the collection of the translation of the triples in $\sigma$.

**Lemma 8.17.** *Let $P = \langle\Gamma\,;\,A\,;\,B\rangle$ be a matching problem in of finite order in $\lambda 2$. Let $\sigma$ be a solution for $P$. (It is no restriction to assume that $dom(\sigma) \subseteq dom(\Gamma)$.) Then $[\![\sigma\,]\!]_P$ is well-typed in $\Delta \equiv [\![A\,]\!]^2_\Gamma, [\![B\,]\!]^2_\Gamma, [\![\,\Gamma\,]\!]$.*

**Proof.** This follows essentially from the fact that $[\![-\,]\!]_-$ preserves judgements (Lemma 8.10) and the fact that we have added auxiliary variables only when they were not already present. Details are left to the reader. ⊠

**Theorem 8.18.** *Let $\langle\Gamma\,;\,A\,;\,B\rangle$ be a third-order matching problem for types in $\lambda 2$ and let $\sigma$ be a solution for $P$. (It is no restriction to assume that $dom(\sigma) \subseteq dom(\Gamma)$.) Then $[\![\sigma\,]\!]_P$ is a solution for $[\![\,P\,]\!]$.*

**Proof.** By Lemma 8.17, $[\![\sigma\,]\!]_P$ is well-typed in $[\![A\,]\!]^2_\Gamma, [\![B\,]\!]^2_\Gamma, [\![\,\Gamma\,]\!]$. Write $\sigma = \{\langle x_i\,;\,\gamma_i\,;\,S_i\rangle \mid 1 \leq i \leq n\}$. We are done if we can show that

$$[\![A\,]\!]^1_\Gamma[x_1 := [\![S_1\,]\!]^1_{\sigma(\Gamma_{x_1}),\gamma_1}, \ldots, x_n := [\![S_n\,]\!]^1_{\sigma(\Gamma_{x_n}),\gamma_n}] \equiv [\![B\,]\!]^1_\Gamma.$$

For convenience we introduce the context $\Xi$ which is the result of inserting, for all $1 \leq i \leq n$, $\gamma_i$ immediately to the left of $x_i$ in $\Gamma$. Note that $\Xi \vdash_{\lambda 2} A : C$ and that $\Xi_{x_i} \vdash_{\lambda 2} S_i : \Gamma(x_i)$. Now we calculate:

$A[x_1 := S_1, \ldots, x_n := S_n] \equiv B$
$[\![A[x_1 := S_1, \ldots, x_n := S_n]\,]\!]^1_{\sigma(\Gamma)} \equiv [\![B\,]\!]^1_{\sigma(\Gamma)}$ so by Lemma 8.7
$[\![A[x_1 := S_1, \ldots, x_n := S_n]\,]\!]^1_{\sigma(\Gamma)} \equiv [\![B\,]\!]^1_{\Gamma}$ so by Lemma 8.13:
$[\![A\,]\!]^1_{\Xi}[x_1 := [\![S_1\,]\!]^1_{\sigma(\Gamma_{x_1}),\gamma_1}, \ldots, x_n := [\![S_n\,]\!]^1_{\sigma(\Gamma_{x_n}),\gamma_n}] \equiv [\![B\,]\!]^1_{\Gamma}$ so by Lemma 8.7
$[\![A\,]\!]^1_{\Gamma}[x_1 := [\![S_1\,]\!]^1_{\sigma(\Gamma_{x_1}),\gamma_1}, \ldots, x_n := [\![S_n\,]\!]^1_{\sigma(\Gamma_{x_n}),\gamma_n}] \equiv [\![B\,]\!]^1_{\Gamma}$.

⊠

**Theorem 8.19.** *Let $\langle \Gamma\,;\, A\,;\, B\rangle$ be a simplified third-order matching problem for objects in $\lambda 2$ and let $\sigma$ be a $*$-solution for $P$. (It is no restriction to assume that $dom(\sigma) \subseteq dom(\Gamma)$.) Then $[\![\sigma\,]\!]_P$ is a solution for $[\![\,P\,]\!]$.*

**Proof.** The proof is the same as the proof of the previous theorem, except for the calculation. By Corollary 7.10, we know that $A[x_1 := S_1, \ldots, x_n := S_n] \twoheadrightarrow_\beta B$. By Lemma 5.8, we know that $A[x_1 := S_1, \ldots, x_n := S_n]$ is simple. We calculate:
$A[x_1 := S_1, \ldots, x_n := S_n] \twoheadrightarrow_\beta B$ so by Lemma 8.15
$[\![A[x_1 := S_1, \ldots, x_n := S_n]\,]\!]^1_{\sigma(\Gamma)} \twoheadrightarrow_\beta [\![B\,]\!]^1_{\sigma(\Gamma)}$ so by Lemma 8.7
$[\![A[x_1 := S_1, \ldots, x_n := S_n]\,]\!]^1_{\sigma(\Gamma)} \twoheadrightarrow_\beta [\![B\,]\!]^1_{\Gamma}$ so by Lemma 8.13:
$[\![A\,]\!]^1_{\Xi}[x_1 := [\![S_1\,]\!]^1_{\sigma(\Gamma_{x_1}),\gamma_1}, \ldots, x_n := [\![S_n\,]\!]^1_{\sigma(\Gamma_{x_n}),\gamma_n}] \twoheadrightarrow_\beta [\![B\,]\!]^1_{\Gamma}$ so by Lemma 8.7
$[\![A\,]\!]^1_{\Gamma}[x_1 := [\![S_1\,]\!]^1_{\sigma(\Gamma_{x_1}),\gamma_1}, \ldots, x_n := [\![S_n\,]\!]^1_{\sigma(\Gamma_{x_n}),\gamma_n}] \twoheadrightarrow_\beta [\![B\,]\!]^1_{\Gamma}$.

⊠

# 9 Decidability of restricted third-order matching

We begin this section by stating, in terms of the $[\![-\,]\!]_-$ translation, sufficient (and necessary) conditions for a substitution to be a solution for a restricted third-order matching problem in $\lambda 2$. We need the following lemma, which, despite its simplicity, is crucial. It tells us that the $[\![-\,]\!]_-$ translation is injective on terms that are of the same type and in LNF.

**Lemma 9.1.** *Suppose $\Gamma \vdash_{\lambda 2} A_1 : B$ and $\Gamma \vdash_{\lambda 2} A_2 : B$. If $A_1$ and $A_2$ are in* LNF *and $[\![A_1\,]\!]^1_\Gamma \equiv [\![A_2\,]\!]^1_\Gamma$, then $A_1 \equiv A_2$.*

**Proof.** Induction on the length of $A_1$. Use the fact that domains are equal since $A_1$ and $A_2$ have the same type $B$. Also use the fact that on polymorphic arguments, $[\![-\,]\!]^1_-$ is injective.

⊠

**Lemma 9.2.** *Let $P = \langle \Gamma\,;\, A\,;\, B\rangle$ be a matching problem for types in $\lambda 2$. Let $\sigma$ be a substitution, well-typed in $\Gamma$, $dom(\sigma) \subseteq dom(\Gamma)$. Suppose that $[\![\sigma\,]\!]_P([\![A\,]\!]^1_\Gamma) \equiv [\![B\,]\!]^1_\Gamma$. Then $\sigma$ is a solution for $P$.*

**Proof.** Since $\sigma$ is well-typed in $\Gamma$, we have $\sigma(\Gamma) \vdash_{\lambda 2} \sigma(A) : *$. Trivially, $\sigma(A)$ is in LNF. We also have $\sigma(\Gamma) \vdash_{\lambda 2} B : *$ and (by Lemma 8.7) $[\![B\,]\!]^1_{\sigma(\Gamma)} \equiv [\![B\,]\!]^1_\Gamma$. As in the proof of Theorem 8.18, $[\![\sigma(A)\,]\!]^1_{\sigma(\Gamma)} \equiv [\![\sigma\,]\!]_P([\![A\,]\!]^1_\Gamma)$. By assumption $[\![\sigma\,]\!]_P([\![A\,]\!]^1_\Gamma) \equiv [\![B\,]\!]^1_\Gamma$. So $[\![\sigma(A)\,]\!]^1_{\sigma(\Gamma)} \equiv [\![B\,]\!]^1_{\sigma(\Gamma)}$. Since both terms have the same type, we can use Lemma 9.1 to obtain that $\sigma(A) \equiv B$. Thus $\sigma$ is a solution for $P$. ⊠

**Lemma 9.3.** *Let $P = \langle \Gamma\,;\, A\,;\, B\rangle$ be a simplified third-order matching problem for objects in $\lambda 2$. Let $\sigma$ be a $*$-substitution, well-typed in $\Gamma$, $dom(\sigma) \subseteq dom(\Gamma)$. Suppose that $[\![\sigma]\!]_P([\![A]\!]^1_\Gamma) \twoheadrightarrow_\beta [\![B]\!]^1_\Gamma$. Then $\sigma$ is a solution for $P$.*

**Proof.** Since $\sigma$ is well-typed in $\Gamma$, the $\beta$-normal form of $\sigma(A)$ exists; let's call it $D$. As in Lemma 7.9, $D$ is in LNF. As in the proof of Theorem 8.19, we have $[\![\sigma]\!]_P([\![A]\!]^1_\Gamma) \twoheadrightarrow_\beta [\![D]\!]^1_{\sigma(\Gamma)}$. By assumption $[\![\sigma]\!]_P([\![A]\!]^1_\Gamma) \twoheadrightarrow_\beta [\![B]\!]^1_\Gamma \equiv [\![B]\!]^1_{\sigma(\Gamma)}$. We conclude that $[\![D]\!]^1_{\sigma(\Gamma)} \equiv [\![B]\!]^1_{\sigma(\Gamma)}$. Since both terms have the same type, we can use Lemma 9.1 to obtain that $D \equiv B$. Thus $\sigma$ is a solution for $P$. ⊠

Consider a third-order matching problem $P$ in $\lambda 2$. Then $[\![P]\!]$ is a third-order matching problem in $\lambda\tau$. Dowek showed that third-order matching is decidable in $\lambda\tau$. The crucial lemma is the following.

**Theorem 9.4** *(Dowek [5]). Let $P$ be a third-order matching problem in $\lambda\tau$. There exists an $n \in \mathbb{N}$, that can be effectively computed from $P$ only, such that the following holds. From a solution $\sigma$ for $P$ one can construct a solution $\sigma^\star$ for $P$ such that the size of each substitution term in $\sigma^\star$ is less than $n$. So if $P$ has a solution then it has a solution such that the size of its substitution terms is less than $n$. We denote this $n$ by $bound(P)$.*

Decidability of third-order matching in $\lambda\tau$ follows from this result plus the following observations. First, in a given context there are only finitely many terms of a given type and a given size. Secondly, because the context is assumed to contain a universal variable $e$ of type $O$, there exists, for every $\lambda\tau$-type, a (closed) term of that type and hence there is no need for auxiliary variables declared in substitution contexts. So the context in which the substitution terms are typed is not arbitrarily extended.

For our purpose we have to adapt the transformation of $\sigma$ into $\sigma^\star$ a little bit. We explain why. The transition from an arbitrary solution $\sigma$ for a matching problem in $\lambda\tau$ to $\sigma^\star$ uses the variable $e$ mentioned above. It proceeds by replacing certain "superfluous" subterms of substitution terms in $\sigma$ by terms either of the form $\lambda x_1{:}S_1 \ldots \lambda x_n{:}S_n.e$, or of the form $\lambda x_1{:}S_1 \ldots \lambda x_n{:}S_n.x_j$ (for some $1 \leq j \leq n$ depending on the subterm in question). Now let $P = \langle \Gamma\,;\, A\,;\, B\rangle$ be a third-order matching problem in $\lambda 2$ and $\tau$ a solution for $P$. We mimick this transformation, applied to substitution terms of $[\![\tau]\!]_P$, in the substitution terms of $\tau$; see Lemma 9.5, below. It will be important later on that the transformation commutes with the $[\![-]\!]_-$ translation, i.e. $([\![\tau]\!]_P)^\star = [\![\tau^\star]\!]_P$. Now it is very well possible that we replace two subterms in the translated term by $\lambda x_1{:}S_1 \ldots \lambda x_n{:}S_n.e$ which in the original term would mean replacing a subterm of type $A_1 \to \cdots \to A_n \to A$ and a subterm of type $B_1 \to \cdots \to B_n \to B$ with $A \not\equiv B$. This would mean that we have to use two variables: $e_1$ of type $A$ and $e_2$ of type $B$. For commutation we would need two variables $e_1$ and $e_2$ of type $O$ in the setting of $\lambda\tau$. So we adapt the transformation as follows: for every occurrence of a subterm (of a translated substitution term) that is replaced by a term $\lambda x_1{:}S_1 \ldots \lambda x_n{:}S_n.f$ ($f \notin \{x_1, \ldots, x_n\}$) we use a new variable $f$.

It is not difficult to see that for each substitution term we need at most $bound([\![P]\!])$ new variables. So for the substitution as a whole the number of new variables need not exceed the number of existential variables in $A$ times $bound([\![P]\!])$. Hence we can beforehand fix

a set of variables from which the substitution terms may be built up. It is convenient, but not necessary, to add the new variables to the corresponding substitution context whenever necessary, instead of assuming these variables to be present in the context beforehand. In any case, by taking new variables only from this fixed set we still have the property (for $\lambda\tau$) that there are finitely many substitutions of a given size.

**Lemma 9.5.** *Let $\Gamma \vdash_{\lambda 2} A : B$, where $A$ is in* LNF*. By Lemma 8.10, $[\![ A ]\!]^2_\Gamma, [\![ \Gamma ]\!] \vdash_{\lambda\tau} [\![ A ]\!]^1_\Gamma :\ll B \gg_\Gamma$. Let $xt_1 \ldots t_n$ be a subterm of $[\![ A ]\!]^1_\Gamma$ in* LNF *and of (atomic!) type $C$. Suppose that $x \in FV(A)$ is not a PV or a variable $p$ added in the construction of $[\![ A ]\!]^1_\Gamma$ or in $\{c^\rightarrow, c^\Pi\}$. Write $\Gamma(x) \equiv S_1 \rightarrow \cdots \rightarrow S_n \rightarrow S$. Fix an occurrence of $xt_1 \ldots t_n$.*

1. *Fix $1 \leq i \leq n$. Write $S_i \equiv R_1 \rightarrow \cdots \rightarrow R_m \rightarrow R$ (note that it is possible to write $S_i$ this way). Let $e$ be a fresh variable of type $O$. Let $\Gamma' \equiv \Gamma, \exists e : R$. Let $D$ be the result of replacing $t_i$ by $\lambda y_1{:} \ll R_1 \gg_\Gamma \ldots \lambda y_m{:} \ll R_m \gg_\Gamma .e$. Obviously, $[\![ A ]\!]^2_\Gamma, [\![ \Gamma ]\!], \exists e : O \vdash_{\lambda\tau} D :\ll B \gg_\Gamma$. Moreover there is an occurrence of a subterm $xt'_1 \ldots t'_n$ in $A$ such that when we let $A'$ be the result of replacing $t'_i$ in this occurrence by $\lambda y_1{:}R_1 \ldots \lambda y_m{:}R_m.e$ then $[\![ A' ]\!]^1_{\Gamma'} \equiv D$ and $\Gamma' \vdash_{\lambda 2} A' : B$.*

2. *Suppose that for some $j$, $1 \leq j \leq n$, we have that $S \equiv S_j$ (hence $S_j \equiv C$). Let $D$ be the result of replacing $x$ in this occurrence by $\lambda y_1{:} \ll S_1 \gg_\Gamma \ldots \lambda y_n{:} \ll S_n \gg_\Gamma .y_j$ and $\beta$-normalizing. Obviously, $[\![ A ]\!]^2_\Gamma, [\![ \Gamma ]\!] \vdash_{\lambda\tau} D :\ll B \gg_\Gamma$. Moreover there is an occurrence of a subterm $xt'_1 \ldots t'_n$ in $A$ such that when we let $A'$ be the result of replacing $x$ in this occurrence by $\lambda y_1{:}S_1 \ldots \lambda y_n{:}S_n.y_j$ and $\beta$-normalizing then $[\![ A' ]\!]^1_\Gamma \equiv D$ and $\Gamma \vdash_{\lambda 2} A' : B$.*

**Proof.** By induction on the structure of $A$. ⊠

Let us return to the discussion preceding this lemma. Let $\langle x\,;\, \gamma\,;\, \lambda x_1{:}S_1 \ldots \lambda x_n{:}S_n.yt_1 \ldots t_m \rangle$ be a triple in $\tau$. For some term $C$ we have that $\sigma(\Gamma_x), \gamma, \forall x_1 : S_1, \ldots, \forall x_n : S_n \vdash_{\lambda\tau} yt_1 \ldots t_m : C$. Furthermore, $[\![ \tau ]\!]_P$ contains a triple $\langle x\,;\, \gamma', \gamma''\,;\, [\![ \lambda x_1{:}S_1 \ldots \lambda x_n{:}S_n.yt_1 \ldots t_m ]\!]_{\sigma(\Gamma_x),\gamma} \rangle$.

The adapted standardisation of $[\![ \tau ]\!]_P$ changes this triple in the following way. The operations described in Lemma 9.5 are applied to $[\![ yt_1 \ldots t_m ]\!]_{\sigma(\Gamma_x),\gamma,\forall x_1:S_1,\ldots,\forall x_n:S_n}$ for some occurrences of variables among $\{x_1, \ldots, x_n\}$. As stated there, this transformation can be mimicked in $yt_1 \ldots t_m$. (Note that no $x_i$ is a polymorphic variable or a variable $p$ or in $\{c^\rightarrow, c^\Pi\}$). One can prove that every variable that occurs free in a substitution term of $([\![ \tau ]\!]_P)^\star$ either occurs in $[\![ B ]\!]^1_\Gamma$ or is a new variable (see [15] for a detailed statement of the properties of $([\![ \tau ]\!]_P)^\star$). We let the new substitution context (replacing $\gamma', \gamma''$) consist of the new variables. This can be mimicked in $\tau$. This yields a well-typed substitution $\tau^\star$ such that $[\![ \tau^\star ]\!]_P = ([\![ \tau ]\!]_P)^\star$. Notice that $\tau^\star$ contains at most $n \cdot bound([\![ P ]\!])$ new variables.

To sum up, we have that the following diagram commutes:

$$\begin{array}{ccc}
\sigma & \xrightarrow{\quad\star\quad} & \sigma^\star \\
{\scriptstyle [\![ - ]\!]_P}\Big\downarrow & & \Big\downarrow{\scriptstyle [\![ - ]\!]_P} \\
[\![ \sigma ]\!]_P & \xrightarrow{\quad\star\quad} & ([\![ \sigma ]\!]_P)^\star = [\![ \sigma^\star ]\!]_P
\end{array}$$

**Theorem 9.6.** *It is decidable whether a restricted third-order matching problem in $\lambda 2$ has a solution or not.*

**Proof.** Let $P = \langle \Gamma\,;\, A\,;\, B \rangle$ be given. We present an algorithm which returns a substitution if $P$ has a solution and returns *fail* otherwise.

1. Translate $P$ to $[\![\, P \,]\!]$.
2. Fix a set $\mathcal{E}$ of $n \cdot \mathit{bound}([\![\, P \,]\!])$ fresh variables, where $n$ is the number of existential variables in $A$.
3. Enumerate all substitutions consisting of substitution terms whose size is smaller than $\mathsf{F}(\mathit{bound}([\![\, P \,]\!]))$ and whose free variables are in $\mathit{dom}(\Gamma) \cup \mathcal{E}$. (For $\mathsf{F}$, see below Definition 8.6).
4. For each such substitution $\sigma$, check whether it is a solution for $P$. If so, return $\sigma$ and stop.
5. If none of the substitutions is a solution for $P$, return *fail* and stop.

We have to show that this algorithm is sound and complete and that it always terminates. Soundness is trivial and termination is easy, given the considerations preceding this theorem. As to completeness. Suppose $P$ has solution $\sigma$. If $P$ is a matching problem for types we have by Theorem 8.18 that $[\![\sigma]\!]_P$ is a solution for $[\![\, P \,]\!]$. If $P$ is a matching problem for objects then, since $P$ is restricted, it is no restriction to assume that $\sigma$ is a $*$-substitution. By Theorem 8.19, $[\![\sigma]\!]_P$ is a solution for $[\![\, P \,]\!]$. So in both cases $[\![\sigma]\!]_P$ is a solution for $[\![\, P \,]\!]$. By Theorem 9.4, $([\![\sigma]\!]_P)^\star$ is also a solution for $[\![\, P \,]\!]$. We have shown that the transition from $[\![\sigma]\!]_P$ to $([\![\sigma]\!]_P)^\star$ can be mimicked in $\sigma$, which yields a substitution $\sigma^\star$, that is well-typed in $\Gamma$ and such that $([\![\sigma]\!]_P)^\star = [\![\sigma^\star]\!]_\Gamma$. By Lemma 9.2 and Lemma 9.3, $\sigma^\star$ is a solution for $P$. By Theorem 9.4, the size of the substitution terms in $([\![\sigma]\!]_P)^\star$ is less than $\mathit{bound}([\![\, P \,]\!])$. So the size of the substitution terms in $\sigma^\star$ is less than $\mathsf{F}(\mathit{bound}([\![\, P \,]\!]))$. We have seen that $\sigma^\star$ does not contain more than $n \cdot \mathit{bound}([\![\, P \,]\!])$ new variables. Hence $\sigma^\star$ occurs in the enumeration in (2) of the algorithm. ⊠

Note that we did not so much use the fact that third-order matching is decidable in $\lambda\tau$ as the proof of that fact. To be precise, we have used the existence of a bound to the size of substitution terms and the particular nature of the transformation of solutions into more efficient solutions. So this part of our proof can only be generalized to higher orders if the proofs of decidability of matching in $\lambda\tau$ for the higher orders are similar to the proof of decidability of third-order matching.

**Corollary 9.7.** *It is decidable whether a third-order matching problem in $\lambda 2$ has a solution or not.*

**Proof.** Let $P$ be a third-order matching problem in $\lambda 2$. If $P$ is a matching problem for types then $P$ is restricted and an algorithm to decide whether $P$ has a solution is given by Theorem 9.6. If $P$ is a matching problem for objects, an algorithm to decide whether $P$ has a solution is given by Corollary 6.12 and Theorem 9.6. ⊠

## References


[1] 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.

[2] T. Coquand and G. Huet. The calculus of constructions. *Information and Control*, 76:95–120, 1988.

[3] G. Dowek. L'indécidabilité du filtrage du troisième ordre dans les calculs avec types dépendants ou constructeurs de types. *Compte Rendu à l'Académie des Sciences*, 312, Série I:951–956, 1991.

[4] G. Dowek. A second-order pattern matching algorithm for the cube of typed $\lambda$-calculi. Technical report, INRIA – Rocquencourt, 1991.

[5] G. Dowek. Third order matching is decidable. In *Proceedings* $7^{th}$ *Annual Symposium on Logic in Computer Science,* Santa Cruz, California, 1992.

[6] G. Dowek. A complete proof synthesis method for the cube of type systems. To be published, 1993.

[7] G. Dowek. The undecidability of pattern matching in calculi where primitive recursive functions are representable. *Theoretical Computer Science*, 107:349–356, 1993.

[8] A.J. Field and P.G. Harrison. *Functional Programming.* Addison-Wesley, 1989.

[9] Ph. Gardner. *Representing Logics in Type Theory.* Phd thesis, University of Edinburgh, January 1992.

[10] J.H. Geuvers. The Church-Rosser property for $\beta\eta$-reduction in typed $\lambda$-calculi. In *Proceedings* $7^{th}$ *Annual Symposium on Logic in Computer Science,* Santa Cruz, California, 1992.

[11] J.H. Geuvers and M.-J. Nederhof. Modular proof of strong normalisation for the calculus of constructions. *Journal of Functional Programming*, 1:155–189, 1989.

[12] W.D. Goldfarb. The undecidability of the second-order unification problem. *Theoretical Computer Science*, 13:225–230, 1981.

[13] V. Padovani. Fourth order dual interpolation is decidable. Manuscript, Université Paris VII - C.N.R.S, 1994.

[14] V. Padovani. On equivalence classes of interpolation equations. Manuscript, Université Paris VII - C.N.R.S, 1994.

[15] J. Springintveld. Third-order matching in the presence of type constructors. Technical Report 112, Logic Group Preprint Series, Utrecht University, May 1994.