INFORMATIK

Third-Order Matching in $\lambda\!\rightarrow$-$Curry$ is Undecidable

Sergei Vorobyov

MPI–I–97–2–006 May 1997

FORSCHUNGSBERICHT RESEARCH REPORT

MAX-PLANCK-INSTITUT
FÜR
INFORMATIK

Im Stadtwald 66123 Saarbrücken Germany

Author's Address

Max-Planck Institut für Informatik, Im Stadtwald, D-66123, Saarbrücken, Germany

Publication Notes

May 28, 1997

Acknowledgements

Harald Ganzinger suggested important improvements to the statement of the problem, motivations, and presentation. Roger Hindley suggested the simple theories of type assignment, explained their classification and interrelation.

Abstract

Given closed untyped λ-terms $\lambda x_1 \dots x_k.s$ and t, which can be assigned some types $\sigma_1 \to \dots \to \sigma_k \to \tau$ and τ respectively in the Curry-style systems of type assignment (essentially due to R. Hindley) $\lambda{\to}$-*Curry* [Bar92], $\lambda_t^{\to}$ [Mit96], TA_λ [Hin97], it is *undecidable* whether there exist closed terms $s_1, \dots, s_k$ of types $\sigma_1, \dots, \sigma_k$ such that $s[s_1/x_1, \cdots, s_k/x_k] =_{\beta\eta} t$, even if the orders of σ_i's do not exceed 3.

This undecidability result should be contrasted to the decidability of the third-order matching in the Church-style simply typed lambda calculus with a single constant base type [Dow94].

The proof is by reduction from the recursively inseparable sets of unsatisfiable and finitely satisfiable sentences of the first-order theory of binary relation [Tra53, Vau60].

Keywords

Typed lambda calculus, Higher-order matching problem, Decidability, Recursive inseparability, First-order theory of binary relation, Full type hierarchy

Contents

1 Introduction **2**

2 Simple Theory of Type Assignment **4**

3 Recursive Inseparability **6**

4 Transformation Sequence **7**

5 Language L_1. Relativization: Translation of L_0 into L_1 **7**

6 Reduction to $\exists^*\forall^*$-Sentences, Language L_2 **9**

7 Transformation to the Pure Prenex $\exists^*\forall^*$-Form **11**

8 Getting Rid of Boolean Connectives and Predicates **11**

9 Transformation to a Higher-Order Matching Problem **12**

10 Faithfulness of the Reduction **13**

1 Introduction

Unification, i.e., solving equations $e(\overline{x}) \stackrel{?}{=} e'(\overline{x})$ in different calculi is at the heart of the theorem-proving procedures. Matching (or half-unification) is a particular case of unification $e(\overline{x}) \stackrel{?}{=} c$, where only the left-hand side $e(\overline{x})$ contains instantiable variables. Matching is important for parameter passing, rewriting, and, more generally, everywhere where a transformation applicability is expressed by a pattern.

In the first-order languages both unification and matching are easily decidable [PW78]. Unification in the third-order languages was proved undecidable by G. Huet [Hue73]. Later W. Goldfarb proved undecidability of unification in the second-order languages [Gol81, Far91]. But Goldfarb's result, unlike Huet's, applies to second-order languages *with constants*, which turn out to be *third-order* after the canonical transformation to the constant-free form [Sta81] (pp. 330–331). Huet and Lang [HL78] proved decidability of the second-order pattern matching (with constants), which should be contrasted to Goldfarb's undecidability result. Today decidability of matching is known for languages (with constants) of orders up to 4 [Dow94, Pad95] and unknown beyond. All the results above hold for the languages with *explicit typing*, which corresponds to the Church-style simply typed lambda calculus with a single constant type o and no variable types.

The *higher-order matching* problem in the simply typed lambda calculus Λ due to G. Huet [Hue76] is formally stated as follows: given an equation $s(x_1, \ldots, x_n) \stackrel{?}{=}_{\beta\eta} t$, where s, t are λ-terms of *fixed* simple types, and t does not contain free variables, do there exist simply typed terms $s_1, \ldots, s_n$ of appropriate types such that $s[s_1/x_1, \ldots, s_n/x_n] = t$ modulo $\beta\eta$-equality? Here the *simply typed lambda calculus* Λ refers to the Church-style system, where terms are annotated with types generated from a single constant type o by a single type constructor $\rightarrow$[1].

Despite all efforts it still remains open whether the above problem is decidable or not. Several approximation results are as follows. Third-order [Dow94] and fourth-order matchings [Pad95] are decidable. Higher-order matching in extensions of Λ with dependent types and type constructors is undecidable [Dow91]. It is also undecidable in the polymorphic second-order system $\lambda 2$ and in Gödel's system T [Dow93]. On the other hand, third-order matching may be decidable or undecidable in Λ extended with type constructors [Spr95]. R. Statman showed [Sta82] that the decidability of the Plotkin-

[1]This should be contrasted to the *simply typed lambda calculus* as defined, e.g., in [Bar92] (Sections 5.1, 3.2) or in [TS96] (Section 1.2), which do have variable types. All systems in Barendregt's λ-cube have variable types.

Statman's definability conjecture would have implied the decidability of the higher-order matching, but R. Loader has proved [Loa93] the undecidability of the lambda definability. D. Wolfram [Wol93] has suggested a higher-order matching algorithm and conjectured its finite termination. The lower complexity bound for the higher-order matching problem in Λ is the tower $2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}$ of height $c \cdot n/\log(n)$, where n is the length of equation and $c > 0$ [Vor97]. Note that the higher-order rewriting community, without waiting until the ultimate solution for the higher-order matching problem in Λ, switched to the so-called *higher-order patterns* [Mil91, Nip91, Nip93] of very restricted linear form. The abovementioned lower bound justifies such a decision.

In this paper we address the higher-order matching problem in the *simple theories of type assignment* (essentially due to R. Hindley), the so-called systems $\lambda{\rightarrow}$-*Curry* [Bar92], $\lambda_t^{\rightarrow}$ [Mit96], TA_λ [Hin97], closely related to the simply typed lambda calculus Λ and typing the same set of terms (modulo type erasure). The importance of the problem stems from the fact that the simple theories of type assignment form a core of the *ML type inference system* $\lambda^{\rightarrow,\Pi,\mathtt{let}}$ [Mit96] (Section 11.3), which extends $\lambda_t^{\rightarrow}$ by adding type abstraction, type application, and polymorphic `let` declaration. The main result of our paper implies that given two simple ML terms (without type abstractions, applications, and polymorphic `let`'s), it is undecidable whether one is an instance of the other obtained by substituting some terms for its free variables. Our undecidability result has direct consequences on the ML-style pattern matching, or more generally, on all higher-order pattern matching languages based on implict typing. It should be noted, however, that for practical reasons and to avoid complications the designers of ML defined ML-patterns to be *linear*, i.e., with each variable occurring at most once. Now this wise but somehow arbitrary solution gets a theoretical justification.

As a technical tool we use the reduction from the recursive inseparability of the sets of *unsatisfiable* and *finitely satisfiable* sentences in the first-order language of a binary relation (Trakhtenbrot-Vaught). This makes the proof especially easy, because a sentence satisfied in a finite binary relation can be realized (witnessed) by a third-order λ-definable functionals of order three over projection functions (which is not the case for infinitely satisfiable sentences). Usually undecidability proofs for higher-order matching are based on Goldfarb's result, or on direct reduction from Hilbert's tenth problem [Dow91, Dow93].

We start by defining the calculi and stating the problem. Then we briefly explain the recursive inseparability techniques and proceed to the reduction from sentences of binary relation to instances of the higher-order matching problem.

2 Simple Theory of Type Assignment

The *simple theory of type assignment*[2] refers to a class of Curry-style systems of type assignment to untyped λ-terms, including the systems $\lambda{\rightarrow}$-*Curry* [Bar92] (pp. 148–150), $\lambda_t^{\rightarrow}$ with implicit typing [Mit96] (p. 782), TA_λ [Hin97]. The main characteristic features of these systems are: the presence of type variables and the absence of built-in types in terms[3].

Definition 1 (Types) Simple types *are generated by* $\mathbb{T}_o \equiv o \mid (\mathbb{T}_o \to \mathbb{T}_o)$, *where* o *is a single type constant and* $\to$ *is a single functional type constructor.* Types *are generated by* $\mathbb{T} = o \mid \alpha \mid (\mathbb{T} \to \mathbb{T})$, *where* α *is a single variable. Types are denoted by* σ, τ. *The* order *of a simple type is defined by* $ord(o) = 1$ *and* $ord(\sigma \to \tau) = \max\{1 + ord(\sigma), ord(\tau)\}$.

Remark 2 Our definition and results extend to finitely or infinitely many type constants and/or variables. We confine ourselves to the simplest simple theory of type assignment. H. Barendregt [Bar92] does not use type constants in $\lambda{\rightarrow}$-*Curry*. We also do not need the type constant o if the orders of types are irrelevant. $\square$

Example 3 *Let* $O_0 = o$ *and* $O_{n+1} = (o \to O_n)$. *Every second-order simple type equals* O_n *for some* $n \geq 1$.

Convention 4 *We adopt the usual conventions on omitting parentheses in writing types with* $\to$ *associating to the right. For* $k \in \omega$, $\sigma^k \to \tau$ *abbreviates* $\underbrace{\sigma \to \ldots \to \sigma}_{k} \to \tau$.

Definition 5 (Terms) *Let* $\mathbb{V} = x, y, z, u, v, w, \ldots$ *(probably, with indices) be an infinite set of variables. Untyped* λ*-terms are generated by*

$$\mathcal{T} = \mathbb{V} \mid \mathcal{T}\,\mathcal{T} \mid \lambda x.\mathcal{T}$$

Define the set of free variables *in a term by* $FV(x) = \{x\}$, $FV(MN) = FV(M) \cup FV(N)$, $FV(\lambda x.M) = FV(M) \setminus \{x\}$. *A term* t *is closed if* $FV(t) = \emptyset$.

Remark 6 We do not introduce, although we could, term constants. $\square$

[2] The term suggested by J. Roger Hindley, private communication, May 6, 1997.

[3] It is worth noting that all systems in λ-cube [Bar92] do have variable types.

Definition 7 (Type Assignment Systems $\lambda\!\rightarrow$-*Curry*, $\lambda_t^{\rightarrow}$, $\boldsymbol{TA}_\lambda$) *Let* Γ *denote a* basis *or a* context, *a finite set of type assumptions* $x_i : \sigma_i$ *for different variables* x_i. *The type assignment rules are as follows [Bar92]:*

$$\Gamma \vdash x : \sigma \quad (\textit{if } x : \sigma \in \Gamma) \qquad\qquad (\textit{axiom})$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \rightarrow \tau} \qquad\qquad (\rightarrow\text{-}\textit{introduction})$$

$$\frac{\Gamma \vdash M : \sigma \rightarrow \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash (MN) : \tau} \qquad\qquad (\rightarrow\text{-}\textit{elimination})$$

A term t *has (or can be assigned) type* τ *in context* Γ *iff* $\Gamma \vdash t : \tau$. *A term* t *is typable in a context* Γ *iff there exists a type* τ *such that* $\Gamma \vdash t : \tau$. *A term* t *is typable iff it is typable in the empty context. We write* $\vdash_{\lambda\rightarrow\text{-}Curry}$ *whenever we need to stress typability in* $\lambda\!\rightarrow$-*Curry*. □

This type assignment system is a core of the *ML type inference system* $\lambda^{\rightarrow,\Pi,\mathtt{let}}$ [Mit96] (Section 11.3), which extends $\lambda_t^{\rightarrow}$ by adding type abstraction, type application, and polymorphic `let` declaration.

Definition 8 (Higher-Order Matching Problem in $\lambda\!\rightarrow$-*Curry*) *Given two* λ*-terms* s *and* t *with* $FV(s) = \{x_1, \ldots, x_k\}$ *and* $FV(t) = \emptyset$, *such that* $\lambda x_1 \ldots x_k.s$ *and* t *can be assigned some types* $\sigma_1 \rightarrow \ldots \rightarrow \sigma_k \rightarrow \tau$ *and* τ *respectively, do there exist closed terms* $s_1, \ldots, s_k$ *with types* σ_i $(i = 1, \ldots, n)$ *such that*

$$s[s_1/x_1, \cdots, s_k/x_k] =_{\beta\eta} t\ ?$$

In the positive case the problem is solvable.

The problem is said to be of order n *if* $\max_{i=1}^{k}(\sigma_k) \leq n$. □

The main result of this paper is the following

Theorem 9 *The higher-order matching in simple type assignment systems* $\lambda\!\rightarrow$-*Curry*, $\lambda_t^{\rightarrow}$, TA_λ *is undecidable, even if terms sought for are of types of order* ≤ 3. □

Remark 10 We prove, in fact, a stronger result: the problem is undecidable even if a variable-free term t on the right-hand side is always of the form $\lambda x_1 \ldots x_p.(\lambda uv.u)$ — it simply ignores its arguments and unconditionally yields *true*. □

3 Recursive Inseparability

A useful method of proving undecidability is by reduction from a pair of recursively inseparable sets. Two disjoint sets A, B are called *recursively inseparable* iff there are no recursive sets C containing A and disjoint with B, i.e., $A \subseteq C$, $B \cap C = \emptyset$. Obviously, each of the recursively inseparable sets A, B is non-recursive.

Proposition 11 *Let A, B be recursively inseparable and f be a total recursive function. Then $f(A)$, $f(B)$ are also recursively inseparable.*

Proof. Suppose $f(A)$, $f(B)$ are separable by a recursive set C. Then A, B are separable by the recursive set $f^{-1}(C)$, the co-image of C. Indeed, $f^{-1}(C)$ is recursive: to test whether $x \in f^{-1}(C)$, compute $f(x)$ and check if $f(x) \in C$.

Moreover, $f^{-1}(C)$ separates A and B. In fact, $f(A) \subseteq C$ implies $A \subseteq f^{-1}(f(A)) \subseteq f^{-1}(C)$. If we suppose that $f^{-1}(C) \cap B \neq \emptyset$ then $C \cap f(B) \neq \emptyset$, and we get a contradiction. $\square$

We will apply the above method as follows. Let $\mathbb{M}$ be the set of all solvable higher-order matching problems (i.e., problems that have solutions). We will take a pair of well-known recursively inseparable sets U and FS and describe a total recursive function f such that $f(U) \subseteq \overline{\mathbb{M}}$ and $f(FS) \subseteq \mathbb{M}$. By Proposition 11, $\mathbb{M}$ cannot be recursive; otherwise $f(U)$ and $f(FS)$ would be recursively separable.

The well-known pair of recursively inseparable sets U and FS is due to B. Trakhtenbrot and R. Vaught. Consider the first-order language L_0 without equality containing just one binary predicate symbol P. Any model for L_0 is called a *binary relation*. A binary relation is *finite* iff its extension (domain) is finite. Let U be the set of all sentences of L_0 false (*Unsatisfiable*) in all binary relations, and FS be the set of all sentences of L_0 satisfied by some *finite* binary relation (*Finitely Satisfiable*).

Theorem 12 (Trakhtenbrot-Vaught) *The sets U and FS are recursively inseparable.* $\square$

Trakhtenbrot proved it for several unary and binary predicate symbols [Tra53], Vaught reduced it to just one binary predicate symbol, [Vau60], Section 4.

4 Transformation Sequence

Given a prenex sentence Δ_0 of the language L_0 (containing just one binary predicate symbol P and no equality) we will describe a recursive function f yielding an instance of the higher-order matching problem $f(\Delta_0)$ with the properties:

$$\Delta_0 \in U \quad \Rightarrow \quad f(\Delta_0) \notin \mathbb{M} \tag{1}$$
$$\Delta_0 \in FS \quad \Rightarrow \quad f(\Delta_0) \in \mathbb{M}, \tag{2}$$

where U is the set of sentences of L_0 false in all binary relations (unsatisfiable), FS is the set of sentences of L_0 true in some finite binary relation (finitely satisfiable), and $\mathbb{M}$ is the set of all higher-order matching problems that have solutions.

By Theorem 12 and Proposition 11, the existence of such a recursive function f implies that the set $\mathbb{M}$ is non-recursive, i.e., the higher-order matching problem is undecidable. It is conceptually easier to divide the description of f in the sequence of transformation steps, including: 1) translation to a higher-order language and relativization, 2) transformation to the relativized and then pure $\exists^*\forall^*$-form (skolemization), 3) elimination of predicates and boolean connectives for the boolean functions, 4) transformation to the pure existential form, and 5) to a higher-order matching problem. All the transformation steps will be done effectively, as described in the following sections.

The central idea behind our proof is that every sentence about a binary relation satisfied by a finite n-element model of binary relation has a λ-definable realization by means of third-order functionals over projection functions $\lambda x_1 \ldots x_n . x_i$ $(i = 1, \ldots, n)$, whereas an unsatisfiable sentence does not have any such realization. A higher-order matching problem we get by transformation is solvable iff such a realization exists.

5 Language L_1. Relativization: Translation of L_0 into L_1

Let L_1 be the language without equality containing variables of type α, the usual boolean connectives, quantifiers, and two distinguished predicate symbols, unary R and binary P, both taking arguments of type α. Define the translation $^@$ of formulas of the language L_0 into formulas of the language L_1 as follows:

$$
\begin{array}{rcl}
(\mathrm{P}(x,y))^{@} & \equiv_{df} & P(x^{\alpha}, y^{\alpha}), \\
(\Phi \wedge \Psi)^{@} & \equiv_{df} & \Phi^{@} \wedge \Psi^{@}, \\
\text{(similarly} & \text{for} & \text{the other boolean connectives),} \\
(\forall x.\Phi)^{@} & \equiv_{df} & \forall x^{\alpha}.[R(x^{\alpha}) \Rightarrow \Phi^{@}], \\
(\exists x.\Phi)^{@} & \equiv_{df} & \exists x^{\alpha}.[R(x^{\alpha}) \wedge \Phi^{@}].
\end{array}
$$

Remark 13 This translation is the usual relativization with respect to the unary predicate R, plus labeling every variable occurrence with type α. Keeping variables explicitly typed will give us additional information on the orders and structure of types for which the problem is undecidable. $\square$

Convention 14 *In the sequel it will be convenient to abbreviate* $\forall x^{\alpha}.[R(x^{\alpha}) \Rightarrow \Phi]$ *by* $\forall^{R} x.\Phi$ *and* $\exists x^{\alpha}.[R(x^{\alpha}) \wedge \Phi]$ *by* $\exists^{R} x.\Phi$.

Definition 15 (Interpretation of L_1) *The language* L_1 *is interpreted in the full type hierarchy over the base domain* ω *of natural numbers. Define*

$$\mathbf{T}_{\omega} \ \equiv_{df} \bigcup_{\tau \in \mathbb{T}_o} \mathbf{H}_{\tau},$$

where $\mathbf{H}_o = \omega$ *and* $\mathbf{H}_{\sigma \to \tau} = \{$ *the set of all functions from* $\mathbf{H}_{\sigma}$ *to* $\mathbf{H}_{\tau}\}$.

Let α *be an arbitrary type from* $\mathbb{T}_o$. *Fix interpretations of the predicate symbols* R, P *as arbitrary subsets* $\mathbf{R} \subseteq \mathbf{H}_{\alpha}$ *and* $\mathbf{P} \subseteq \mathbf{H}_{\alpha} \times \mathbf{H}_{\alpha}$. *The validity relation* $\langle \mathbf{T}_{\omega},\ \mathbf{R},\ \mathbf{P} \rangle \models \Phi$ *for a sentence* Φ *of* L_1 *is defined in the usual manner.*

We will now relate validity of sentences of L_0 and L_1.

Lemma 16 *Let* Δ_0 *be a sentence of* L_0 *and* Δ_1 *in* L_1 *be its* $^{@}$*-translation.*

1. *If for some type* $\alpha \in \mathbb{T}_o$ *and some interpretations* $\mathbf{R} \subseteq \mathbf{H}_{\alpha}$, $\mathbf{P} \subseteq \mathbf{H}_{\alpha} \times \mathbf{H}_{\alpha}$ *of predicate symbols* R, P *one has* $\langle \mathbf{T}_{\omega},\ \mathbf{R},\ \mathbf{P} \rangle \models \Delta_1$, *then* Δ_0 *is true in some model of binary relation, or, equivalently,* $\Delta_0 \notin U$.

2. *If the sentence* Δ_0 *is true in some finite model* $\langle \{a_1, \ldots, a_n\};\ \mathbf{B} \rangle$ *(where* $\mathbf{B} \subseteq \{a_1, \ldots, a_n\}^2$*) of binary relation, then* $\langle \mathbf{T}_{\omega},\ \mathbf{R},\ \mathbf{P} \rangle \models \Delta_1$ *for the type* $\alpha \equiv o^n \to o$ *and for the finite interpretations* $\mathbf{R} \subseteq \mathbf{H}_{\alpha}$, $\mathbf{P} \subseteq \mathbf{H}_{\alpha} \times \mathbf{H}_{\alpha}$ *of predicate symbols* R, P *defined by* $\mathbf{R} = \{g_1, \ldots, g_n\}$ *(where* $g_i \equiv \lambda x_1^o \ldots \lambda x_n^o.x_i$ *of type* $\alpha \equiv o^n \to o$ *for* $i = 1, \ldots, n$*) and* $\mathbf{P} = \{\langle g_i, g_j \rangle \mid \langle a_i, a_j \rangle \in \mathbf{B}\}$.

Proof. 1) A model $\langle \mathbf{T}_{\omega},\ \mathbf{R},\ \mathbf{P} \rangle \models \Delta_1$ can be straightforwardly transformed into a model of binary relation satisfying Δ_0. 2) Obvious. $\square$

6 Reduction to $\exists^*\forall^*$-Sentences, Language L_2

Consider the sentences of L_1 resulting from the ${}^{@}$-translation of the prenex sentences of L_0. Let us call these sentences *quasi-prenex*, since according to our convention they have form $Q^R x_1^\alpha \dots Q^R x_n^\alpha\ \Phi$, where $Q_i \in \{\forall, \exists\}$ and Φ is quantifier-free.

The language L_2 is like L_1, but allows variables of types of the form $\alpha^k \to \alpha$, and quantification over these variables. In this section we show that every such quasi-prenex sentence is equivalent to an $\exists^*\forall^*$-quasi-prenex sentence of the language L_2 of the form $\exists w_1^{\sigma_1} \dots \exists w_m^{\sigma_m} \forall^R u_1^\alpha \dots \forall^R u_p^\alpha\ \Phi$, where Ψ is a quantifier-free formula of L_2 and the types σ_i are of the form $\alpha^k \to \alpha$. The transformation we use is the usual "skolemization". Notice that the full type hierarchy $\mathbf{T}_\omega$ contains all skolem functions.

Lemma 17 *Every quasi-prenex sentence of* L_1

$$\Delta_1 \equiv Q^R x_1^\alpha \dots Q^R x_n^\alpha\ \Phi, \tag{3}$$

where $Q_i \in \{\forall, \exists\}$ *and* Φ *quantifier-free, can be transformed into an* $\exists^*\forall^*$*-quasi-prenex sentence of* L_2 *of the form*

$$\Delta_2 \equiv \exists w_1^{\sigma_1} \dots \exists w_p^{\sigma_p} \forall^R y_1^\alpha \dots \forall^R y_q^\alpha\ \Psi, \tag{4}$$

such that Ψ *is a quantifier-free formula of* L_2*, the types* $\sigma_i \equiv \alpha^{r_i} \to \alpha$ *(for* $1 \le i \le p$ *and* $r_i \in \omega$*), and the following properties hold:*

1. *If for some type* $\alpha \in \mathbb{T}_o$ *and some interpretations* $\mathbf{R} \subseteq \mathbf{H}_\alpha$ *for* R *and* $\mathbf{P} \subseteq \mathbf{H}_\alpha \times \mathbf{H}_\alpha$ *for* P *one has* $\langle \mathbf{T}_\omega, \mathbf{R}, \mathbf{P} \rangle \models \Delta_2$*, then for these interpretations one also has* $\langle \mathbf{T}_\omega, \mathbf{R}, \mathbf{P} \rangle \models \Delta_1$*.*

2. *Let* $\langle \mathbf{T}_\omega,\ \mathbf{R},\ \mathbf{P} \rangle \models \Delta_1$ *for some type* α *of the form* $o^n \to o$ *(*$n \in \omega$*) and finite interpretations* $\mathbf{R} \subseteq \mathbf{H}_\alpha$*,* $\mathbf{P} \subseteq \mathbf{R} \times \mathbf{R}$ *of predicate symbols* R*,* P *such that (cf., Lemma 16.2)* $\mathbf{R} = \{p_1, \dots, p_n\}$ *(where* $p_i = \lambda x_1^o \dots \lambda x_n^o . x_i$*,* $i = 1, \dots, n$*) and* $\mathbf{P} \subseteq \mathbf{R} \times \mathbf{R}$*.*

   Then $\langle \mathbf{T}_\omega,\ \mathbf{R},\ \mathbf{P} \rangle \models \Delta_2$*, and, moreover, the witness functions for the existentially quantified variables* $w_1^{\sigma_1} \dots w_p^{\sigma_p}$ *in (4) may be chosen from the finite sets of functions* $(P_n)^k \to P_n$*, where* $P_n = \{\lambda x_1^o \dots \lambda x_n^o . x_i \mid i = 1, \dots, n\}$*.*

Proof. If the formula (3) is already in the form (4), nothing has to be done. Otherwise repeatedly select the rightmost maximal $\forall^*\exists^*$-blocks (with nonempty $\forall^*$- and $\exists^*$-parts) and transform them into $\exists^*\forall^*$-blocks as described below. This repeated transformation always terminates, since the number of such $\forall^*\exists^*$-blocks reduces at each step.

Consider the rightmost maximal $\forall^*\exists^*$ block (with nonempty $\forall$ and $\exists$ parts):

$$\underbrace{\ldots\exists^R u_q^\alpha}_{(A)}\underbrace{\forall^R x_1^\alpha\ldots\forall^R x_m^\alpha}_{(B)}\underbrace{\exists y_1^{\sigma_1}\ldots\exists y_n^{\sigma_n}}_{(C)}\underbrace{\left[\forall^R z_1^\alpha\ldots\forall^R z_p^\alpha\ \Phi,\right]}_{(D)}. \tag{5}$$

The following *invariant* properties hold at every transformation step. They are immediately checked by induction on the number of transformations.

- Part (A) is either empty or ends with an existential quantifier.
- Part (D) may contain only universal quantifiers.
- All quantifiers in parts (A), (B), (D) are R-relativized, and all quantified variables in (A), (B), (D) are of type α.
- All quantified variables in part (C) are of types σ_i of the form $\alpha^{r_i}\to\alpha$ for $r_i\in\omega$.

Let $\overline{w}$ be the vector of all variables universally quantified in parts (A) and (B) (all these variables are of type α). The transformation step consists in replacement of the $\forall^*\exists^**$-quantifier block (B)(C) in (5) with the quantifier block $\exists^*\forall^*$ (C′)(B′), and modifying the part (D) into (D′) as shown below:

$$\underbrace{\ldots\exists^R u_q^\alpha}_{(A)}\underbrace{\exists f_1^{\tau_1}\ldots\exists f_n^{\tau_n}}_{(C')}\underbrace{\forall^R x_1^\alpha\ldots\forall^R x_m^\alpha}_{(B')}\underbrace{\left[\forall^R z_1^\alpha\ldots\forall^R z_p^\alpha\ \Phi[f_i(\overline{w})/y_i]_{i=1}^n,\right]}_{(D')}, \tag{6}$$

where $\tau_i=\alpha^{p+m}\to\sigma_i$ for $1\le i\le n$ and some $m\ge 0$.

This is an equivalence transformation in the full type hierarchy $\mathbf{T}_\omega$, which contains all possible (skolem) functions: if for every x there exists a y such that $\phi(x,y)$, then there exists a function h such that $\forall x.\phi(x,h(x))$. [In principle, the equivalence claim here requires the Axiom of Choice. However, in Lemma 17.1 we do not need equivalence, just a weakening $\exists\forall\Rightarrow\forall\exists$. For the claim of Lemma 17.2 the Axiom of Choice is unneeded.]

The claims of the lemma are obtained by induction on the number of transformations described above. □

7 Transformation to the Pure Prenex $\exists^*\forall^*$-Form

After the previous transformation the sentence Δ_2 in (4) may contain the relativized quantifiers. Transform it to the pure prenex sentence Δ_3, see (7) below, by applying the usual logical equivalences for $Q \in \{\exists, \forall\}$ to push quantifiers outwards:

$$\begin{array}{rcl} \exists x(R(x) \wedge Qy\Phi(x,y)) & \Leftrightarrow & \exists x Qy(R(x) \wedge \Phi(x,y)), \\ \forall x(R(x) \Rightarrow Qy\Phi(x,y)) & \Leftrightarrow & \forall x Qy(R(x) \Rightarrow \Phi(x,y)). \end{array}$$

8 Getting Rid of Boolean Connectives and Predicates

The sentence Δ_3, equivalent to Δ_2 in (4), we obtained so far is of the form

$$\Delta_3 \equiv \exists \overline{y}^{\overline{\sigma}} \forall \overline{x}^{\overline{\alpha}} \Phi, \tag{7}$$

with Φ quantifier-free built of variables of types $\alpha^k \to \alpha$, boolean connectives, and predicate symbols R (unary), P (binary) taking arguments of type α.

Transform Δ_3 into Δ_4 by getting rid of boolean connectives and predicate symbols R, P as follows. Let **bool** be the type $o \to o \to o$ of *booleans* with $\mathbf{t} \equiv \lambda x^o y^o.x$ (true) and $\mathbf{f} \equiv \lambda x^o y^o.y$ (false), both of type **bool**. Let (by slightly abusing notation) R, P be variables of types $\alpha \to \mathbf{bool}$ and $\alpha \to \alpha \to \mathbf{bool}$ respectively, and

$$\Delta_4 \equiv \exists R^{\alpha\to\mathbf{bool}} \exists P^{\alpha\to\alpha\to\mathbf{bool}} \exists \overline{y}^{\overline{\sigma}} \forall \overline{x}^{\overline{\alpha}} (\Phi^* = \mathbf{t}), \tag{8}$$

where the * transformation is defined as follows:

1. $R(t)^* = R(t)$ and $P(s,t)^* = P(s,t)$ (with predicate symbols R, P on the left and function variables R, P of types $\alpha \to \mathbf{bool}$, $\alpha \to \alpha \to \mathbf{bool}$ on the right),

2. $(\neg\Phi)^* = \mathit{Cond}\ (\Phi^*)\ \mathbf{f}\ \mathbf{t}$,

3. $(\Phi \wedge \Psi)^* = \mathit{Cond}\ (\Phi^*)\ (\mathit{Cond}\ (\Psi^*)\ \mathbf{t}\ \mathbf{f})\ \mathbf{f}$,

4. $(\Phi \vee \Psi)^* = \mathit{Cond}\ (\Phi^*)\ \mathbf{t}\ (\mathit{Cond}\ (\Psi^*)\ \mathbf{t}\ \mathbf{f})$,

   where $\mathit{Cond} = \lambda x^{\mathbf{bool}} \lambda y^{\mathbf{bool}} \lambda z^{\mathbf{bool}}.(\lambda a^o \lambda b^o.x(yab)(zab))$

   (one can easily check that $\mathit{Cond}\ \mathbf{t}\ x\ y = x$ and $\mathit{Cond}\ \mathbf{f}\ x\ y = y$).

Lemma 18 *1. If for some type $\alpha \in \mathbb{T}_o$ one has $\mathbf{T}_\omega \models \Delta_4$, then for some interpretations $\mathbf{R} \subseteq \mathbf{H}_\alpha$ and $\mathbf{P} \subseteq \mathbf{H}_\alpha \times \mathbf{H}_\alpha$ for predicate symbols R and P one has $\langle \mathbf{T}_\omega, \mathbf{R}, \mathbf{P} \rangle \models \Delta_3$.*

2. Let $\langle \mathbf{T}_\omega,\ \mathbf{R},\ \mathbf{P} \rangle \models \Delta_3$ for some type α of the form $o^n \to o$ ($n \in \omega$) and finite interpretations $\mathbf{R} \subseteq \mathbf{H}_\alpha$ and $\mathbf{P} \subseteq \mathbf{H}_\alpha \times \mathbf{H}_\alpha$ of predicate symbols R, P such that (cf., Lemmas 16.2 and 17.2) $\mathbf{R} = \{p_1, \ldots, p_n\}$ (where $p_i = \lambda x_1^o \ldots \lambda x_n^o . x_i$, $i = 1, \ldots, n$) and $\mathbf{P} \subseteq \mathbf{R} \times \mathbf{R}$.

Then $\mathbf{T}_\omega \models \Delta_4$, and, moreover, the witness functions for the existentially quantified variables in (8) may be chosen from the finite sets of functions $(P_n)^k \to P_n$ or $(P_n)^k \to \mathbf{bool}$, where $P_n = \{\lambda x_1^o \ldots \lambda x_n^o . x_i \mid i = 1, \ldots, n\}$.

Proof. 1) Let for some type $\alpha \in \mathbb{T}_o$ one has $\mathbf{T}_\omega \models \Delta_4$. Then for some functions $R \in \mathbf{H}_{\alpha\to\mathbf{bool}}$, $P \in \mathbf{H}_{\alpha\to\alpha\to\mathbf{bool}}$ one has $\mathbf{T}_\omega \models \exists \overline{y}^{\overline{\sigma}} \forall \overline{x}^{\overline{\alpha}} (\Phi^* = \mathbf{t})$. Define the interpretations $\mathbf{R} \subseteq \mathbf{H}_\alpha$ and $\mathbf{P} \subseteq \mathbf{H}_\alpha \times \mathbf{H}_\alpha$ for the predicate symbols R and P as $\mathbf{R} = \{u \mid Ru = \mathbf{t}\}$ and $\mathbf{P} = \{\langle u, v \rangle \mid Puv = \mathbf{t}\}$. Then induction on the definition of the * translation shows that for all values of free parameters of Φ^* (except R, P) one has: Φ is true iff $\Phi^* = \mathbf{t}$.

2) Straightforward. $\square$

9 Transformation to a Higher-Order Matching Problem

Transform the sentence Δ_4 in (8) we obtained so far into a higher-order matching problem by: 1) omitting the leading existential quantifiers, 2) eliminating the trailing universal quantifiers by rewriting $\forall \overline{x}^{\overline{\alpha}} (\Phi^* = \mathbf{t})$ into $\lambda \overline{x}^{\overline{\alpha}} \Phi^* = \lambda \overline{x}^{\overline{\alpha}} \mathbf{t}$, and 3) erasing all type information from the last equation, thus getting

$$\Delta_5 \ \equiv_{df} \ \lambda \overline{x} \Phi^{*\prime} = \lambda \overline{x} (\lambda uv.u), \tag{9}$$

where $\Phi^{*\prime}$ is obtained from Φ^* by erasing all types. We have the following

Lemma 19 *Let the matching problem Δ_5 in (9) be obtained from the sentence Δ_4 in (8) by the transformation described above.*

1. If the matching problem Δ_5 has a solution, then $\mathbf{T}_\omega \models \Delta_4$ for some type $\alpha \in \mathbb{T}_o$.

2. *Let* $\mathbf{T}_\omega \models \Delta_4$ *for some type* α *of the form* $o^n \to o$ $(n \in \omega)$ *and, moreover, the witness functions for the existentially quantified variables in (8) may be chosen from the finite sets of functions (cf., Lemmas 16.2, 17.2, and 18.2)* $(P_n)^k \to P_n$ *or* $(P_n)^k \to \mathbf{bool}$*, where* $P_n = \{\lambda x_1^o \ldots \lambda x_n^o.x_i \mid i = 1, \ldots, n\}$*. Then the matching problem* Δ_5 *has a solution of order* ≤ 3.

Proof. 1) Obvious, because for all closed terms $\bar{t}$ one has $\mathbf{T}_\omega \models \Psi[\bar{t}/\bar{x}] \Rightarrow \exists x \Psi$ for all Ψ (logical rule), and $\mathbf{T}_\omega \models \lambda x.s = \lambda x.t$ is equivalent to $\mathbf{T}_\omega \models \forall x(s = t)$.

2) Every function in the finite sets of functions $(P_n)^k \to P_n$ or $(P_n)^k \to \mathbf{bool}$, where $P_n = \{\lambda x_1^o \ldots \lambda x_n^o.x_i \mid i = 1, \ldots, n\}$ is λ-definable by a closed term. In fact, every $\lambda x_1^o \ldots \lambda x_n^o.x_i$ is λ-definable, and the equality between two elements of P_n is λ-definable ([HKM97], Section 2.4) by

$$EQ_n \equiv_{df} \lambda p^{o^n \to o} q^{o^n \to o} \lambda u^o v^o.p(qu\underbrace{v \ldots v}_{n-1})(qvu\underbrace{v \ldots v}_{n-2}) \ldots (q\underbrace{v \ldots v}_{n-1}u).$$

$(EQ_n(\lambda x_1 \ldots x_n.x_i)(\lambda x_1 \ldots x_n.x_j)$ normalizes to $\mathbf{t}$ if $i = j$ and to $\mathbf{f}$ otherwise.)

Using EQ, every function in the finite sets of functions $(P_n)^k \to P_n$ or $(P_n)^k \to \mathbf{bool}$, where $P_n = \{\lambda x_1^o \ldots \lambda x_n^o.x_i \mid i = 1, \ldots, n\}$ is easily lambda-definable by a $Cond$-expression encoding a finite table. The order of any term defining a function in $(P_n)^k \to P_n$ or $(P_n)^k \to \mathbf{bool}$ is ≤ 3. □

10 Faithfulness of the Reduction

Let $f(\Delta_0) = \Delta_5$, where the sequence of transformations $\Delta_0 \rightsquigarrow \Delta_1 \rightsquigarrow \Delta_2 \rightsquigarrow \Delta_3 \rightsquigarrow \Delta_4 \rightsquigarrow \Delta_5$ is described in the previous sections. It is obvious from the description of all transformations that the function f is recursive. We now turn to the proof that f satisfies the properties (1) and (2).

Case $\Delta_0 \in U$. Instead of (1) we prove equivalent $f(\Delta_0) \in \mathbb{M} \Rightarrow \Delta_0 \notin U$. Let the higher-order matching problem $\Delta_5 = f(\Delta_0)$ have a solution.

1. Lemma 19.1 implies that $\mathbf{T}_\omega \models \Delta_4$ for some type $\alpha \in \mathbb{T}_o$.

2. Lemma 18.1 implies that $\langle \mathbf{T}_\omega, \mathbf{R}, \mathbf{P} \rangle \models \Delta_{2,3}$ (Δ_2 and Δ_3 are equivalent) for this type $\alpha \in \mathbb{T}_o$ and some interpretations $\mathbf{R} \subseteq \mathbf{H}_\alpha$, $\mathbf{P} \subseteq \mathbf{H}_\alpha \times \mathbf{H}_\alpha$ for R, P.

3. Lemma 17.1 implies that $\langle \mathbf{T}_\omega,\ \mathbf{R},\ \mathbf{P}\rangle \models \Delta_1$ for this type $\alpha \in \mathbb{T}_o$ and these interpretations $\mathbf{R} \subseteq \mathbf{H}_\alpha$, $\mathbf{P} \subseteq \mathbf{H}_\alpha \times \mathbf{H}_\alpha$ of predicate symbols R and P.

4. Lemma 16.1 implies that Δ_0 is true in some model of binary relation, i.e., $\Delta_0 \notin U$ (recall that U is a set of unsatisfiable sentences about a binary relation).

   Thus, (1) is proved. □

Remark 20 This part of the proof corresponds to the correctness of transformation (the first clauses of Lemmas 16 – 19): if the resulting higher-order matching problem Δ_5 has a solution, then the initial sentence Δ_0 is satisfied by some binary relation. □

Case $\Delta_0 \in FS$. Let a sentence Δ_0 of L_0 be true in a finite model of binary relation $\langle\{a_1, \ldots, a_n\};\ \mathbf{B}\rangle$ (where $\mathbf{B} \subseteq \{a_1, \ldots, a_n\}^2$) of cardinality $n \in \omega$.

1. Lemma 16.2 implies that $\langle \mathbf{T}_\omega,\ \mathbf{R},\ \mathbf{P}\rangle \models \Delta_1$ for the type $\alpha \equiv o^n \to o$ and for the finite interpretations $\mathbf{R} \subseteq \mathbf{H}_\alpha$, $\mathbf{P} \subseteq \mathbf{H}_\alpha \times \mathbf{H}_\alpha$ of predicate symbols R, P defined by $\mathbf{R}_\alpha = \{g_1, \ldots, g_n\}$ (where $g_i = \lambda x_1^o \ldots \lambda x_n^o.x_i$ for $i = 1, \ldots, n$), $\mathbf{P}_\alpha = \{\langle g_i, g_j\rangle \mid \langle a_i, a_j\rangle \in \mathbf{B}\}$.

2. Lemma 17.2 implies that $\langle \mathbf{T}_\omega,\ \mathbf{R},\ \mathbf{P}\rangle \models \Delta_{2,3}$, and, moreover, the witness functions for the existentially quantified variables $w_1^{\sigma_1} \ldots w_p^{\sigma_p}$ in (4) may be chosen from the finite sets of functions $(P_n)^k \to P_n$, where $P_n = \{\lambda x_1^o \ldots \lambda x_n^o.x_i \mid\ i = 1, \ldots, n\}$.

3. Lemma 18.2 implies that $\mathbf{T}_\omega \models \Delta_4$, and, moreover, the witness functions for the existentially quantified variables in (8) may be chosen from the finite sets of functions $(P_n)^k \to P_n$ or $(P_n)^k \to \mathbf{bool}$, where $P_n = \{\lambda x_1^o \ldots \lambda x_n^o.x_i \mid\ i = 1, \ldots, n\}$.

4. Lemma 19.2 implies that the matching problem Δ_5 has a solution of order ≤ 3.

   Thus, (2) and the main claim of this paper are proved. □

Remark 21 This part of the proof corresponds to the *finite completeness* of the transformation (the second clauses of Lemmas 16 – 19): if a sentence Δ_0 is satisfied by a finite binary relation, then the resulting higher-order matching problem Δ_5 has a closed λ-definable solution in the form of functionals (of order 3) over projection functions P_n.

Summarizing, the whole proof works smoothly thanks to the helpful dichotomy in Trakhtenbrot-Vaught's Theorem 12 "*unsatisfiable* vs. *finitely satisfiable*" plus the recursive inseparability. This is because we need not care about the case when a sentence Δ_0 is satisfied by an *infinite* model of binary relation, when we may be unable to find a closed solution to the resulting higher-order matching problem Δ_5. $\square$

References

[Bar92] 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 118–308, Oxford, 1992. Clarendon Press.

[Dow91] G. Dowek. L'indécidabilité du filtrage du troisième ordre dans les calculs avec types dépendants ou constructeurs de types. *Comptes Rendus à l'Académie des Sciences, Paris*, vol. 312 (Série I):951–956, 1991. (*ou* (or) should read *et* (and), cf., Erratum, ibid., vol. 318: 873, 1994).

[Dow93] G. Dowek. The undecidability of pattern matching in calculi where primitive recursive functions are representable. *Theor. Comput. Sci.*, 107:349–356, 1993.

[Dow94] G. Dowek. Third order matching is decidable. *Annals Pure Appl. Logic*, 69:135–155, 1994. Preliminary version in LICS'92.

[Far91] W. Farmer. Simple second-order languages for which unification is undecidable. *Theor. Comput. Sci.*, 87:25–41, 1991.

[Gol81] W. D. Goldfarb. The undecidability of the second-order unification problem. *Theor. Comput. Sci.*, 13:225–230, 1981.

[Hin97] J. R. Hindley. *Basic Lambda Calculus.* Cambridge Tracts in Theoretical Computer Science Series. Cambridge Univ. Press, 1997.

[HKM97] G. D. Hillebrand, P. C. Kanellakis, and H. G. Mairson. Database query languages embedded in the typed lambda calculus. *Information and Computation*, 1997. To appear. Preliminary version in LICS'93, pp. 332–343.

[HL78] G. Huet and B. Lang. Proving and applying program transformations expressed with second-order patterns. *Acta Informatica*, 11:31–55, 1978.

[Hue73] G. Huet. The undecidability of unification in third-order logic. *Information and Computation*, 22:257–267, 1973.

[Hue76] G. Huet. *Résolution d'Équations dans les Langages d'Ordre 1, 2, . . . , ω*. Thèse de Doctorat d'État, Université de Paris VII, 1976.

[Loa93] R. Loader. The undecidability of λ-definability. "Types" electronic forum, 1993.

[Mil91] D. Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. In P. Schroeder-Heister, editor, *Extensions of Logic Programming*, volume 475 of *Lect. Notes Comput. Sci.*, pages 253–281. Springer-Verlag, 1991.

[Mit96] J. C. Mitchell. *Foundations for Programming Languages*. Foundations of Computing Series. MIT Press, 1996.

[Nip91] T. Nipkow. Higher-order critical pairs. In *6th Annual IEEE Symp. on Logic in Computer Science (LICS'91)*, pages 342–349. IEEE, 1991.

[Nip93] T. Nipkow. Functional unification for higher-order patterns. In *8th Annual IEEE Symp. on Logic in Computer Science (LICS'93)*, pages 64–74. IEEE, 1993.

[Pad95] V. Padovani. On equivalence classes of interpolation equations. In *Typed Lambda Calculi and Applications, TLCA'95, Edinburgh, The Netherlands*, volume 902 of *Lect. Notes Comput. Sci.*, pages 335–349. Springer-Verlag, 1995.

[PW78] M.S. Paterson and M.N. Wegman. Linear unification. *J. Comput. Syst. Sci.*, 16:158–167, 1978.

[Spr95] J. Springintveld. Third-order matching in the presence of type constructors. In *Typed Lambda Calculi and Applications, TLCA'95, Edinburgh, The Netherlands*, volume 902 of *Lect. Notes Comput. Sci.*, pages 428–442. Springer-Verlag, 1995.

[Sta81] R. Statman. On the existence of closed terms in the typed λ-calculus II: transformations of unification problems. *Theor. Comput. Sci.*, 15:329–338, 1981.

[Sta82] R. Statman. Completeness, invariance, and λ-definability. *J. Symb. Logic*, 47(1):17–26, 1982.

[Tra53] B. A. Trakhtenbrot. On recursive separability. *Soviet Math. Doklady*, 88:953–956, 1953. In Russian.

[TS96] A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*, volume 43 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge Univ. Press, 1996.

[Vau60] R. L. Vaught. Sentences true in all constructive models. *J. Symb. Logic*, 25(1):39–53, 1960.

[Vor97] S. Vorobyov. The "hardest" natural decidable theory. In G. Winskel, editor, *12th Annual IEEE Symp. on Logic in Computer Science (LICS'97)*. IEEE, June 1997. To appear, available at `http://www.mpi-sb.mpg.de/~sv`.

[Wol93] D. A. Wolfram. *The Clausal Theory of Types*, volume 21 of *Cambridge Tracts in Theoretical Computer Science Series*. Cambridge Univ. Press, 1993.

I N F O R M A T I K

Below you find a list of the most recent technical reports of the research group *Logic of Programming* at the Max-Planck-Institut für Informatik. They are available by anonymous ftp from our ftp server `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Birgit Hofmann
Im Stadtwald
D-66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-97-2-005	L. Bachmair, H. Ganzinger	A Theory of Resolution
MPI-I-97-2-003	U. Hustadt, R.A. Schmidt	On evaluating decision procedures for modal logic
MPI-I-97-2-002	R.A. Schmidt	Resolution is a decision procedure for many propositional modal logics
MPI-I-97-2-001	D.A. Basin, S. Matthews, L. Viganò	Labelled modal logics: quantifiers
MPI-I-96-2-010	A. Nonnengart	Strong Skolemization
MPI-I-96-2-009	D.A. Basin, N. Klarlund	Beyond the Finite in Automatic Hardware Verification
MPI-I-96-2-008	S. Vorobyov	On the decision complexity of the bounded theories of trees
MPI-I-96-2-007	A. Herzig	SCAN and Systems of Conditional Logic
MPI-I-96-2-006	D.A. Basin, S. Matthews, L. Viganò	Natural Deduction for Non-Classical Logics
MPI-I-96-2-005	A. Nonnengart	Auxiliary Modal Operators and the Characterization of Modal Frames
MPI-I-96-2-004	G. Struth	Non-Symmetric Rewriting
MPI-I-96-2-003	H. Baumeister	Using Algebraic Specification Languages for Model-Oriented Specifications
MPI-I-96-2-002	D.A. Basin, S. Matthews, L. Viganò	Labelled Propositional Modal Logics: Theory and Practice
MPI-I-96-2-001	H. Ganzinger, U. Waldmann	Theorem Proving in Cancellative Abelian Monoids
MPI-I-95-2-011	P. Barth, A. Bockmayr	Modelling Mixed-Integer Optimisation Problems in Constraint Logic Programming
MPI-I-95-2-010	D.A. Plaisted	Special Cases and Substitutes for Rigid E-Unification
MPI-I-95-2-009	L. Bachmair, H. Ganzinger	Ordered Chaining Calculi for First-Order Theories of Binary Relations
MPI-I-95-2-008	H.J. Ohlbach, R.A. Schmidt, U. Hustadt	Translating Graded Modalities into Predicate Logic
MPI-I-95-2-007	A. Nonnengart, A. Szalas	A Fixpoint Approach to Second-Order Quantifier Elimination with Applications to Correspondence Theory