Proceedings of the 7th Annual IEEE Symposium on Logic in Computer Science

Proceedings of the

Seventh Annual IEEE Symposium on

Logic in Computer Science

Santa Cruz, California
June 22 - 25, 1992

Sponsored by: IEEE Computer Society Technical Committee on Mathematical Foundations of Computing
in cooperation with ACM SIGACT
Association for Symbolic Logic
Europe Association for Theoretical Computer Science

IEEE Computer Society Press
Los Alamitos, California

Washington • Brussels • Tokyo

The papers in this book comprise the proceedings of the meeting mentioned on the cover and title page. They reflect the authors' opinions and, in the interests of timely dissemination, are published as presented and without change. Their inclusion in this publication does not necessarily constitute endorsement by the editors, the IEEE Computer Society Press, or the Institute of Electrical and Electronics Engineers, Inc.

Published by the
IEEE Computer Society Press
10662 Los Vaqueros Circle
PO Box 3014
Los Alamitos, CA 90720-1264

IEEE Computer Society Press Order Number 2735
Library of Congress Number 91-78307
IEEE Catalog Number 92CH3127-8
ISBN 0-8186-2735-2 (paper)
ISBN 0-8186-2736-0 (microfiche)
ISBN 0-8186-2737-9 (case)

Additional copies can be ordered from

IEEE Computer Society Press
Customer Service Center
10662 Los Vaqueros Circle
PO Box 3014
Los Alamitos, CA 90720-1264

IEEE Service Center
445 Hoes Lane
PO Box 1331
Piscataway, NJ 08855-1331

IEEE Computer Society
13, avenue de l'Aquilon
B-1200 Brussels
BELGIUM

IEEE Computer Society
Ooshima Building
2-19-1 Minami-Aoyama
Minato-ku, Tokyo 107
JAPAN

Editorial Production: Anne Copeland
Printed in the United States of America by Braun-Brumfield, Inc.

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

Preface

The LICS Symposium aims to attract high quality original papers covering theoretical and practical issues in computer science that relate to logic in a broad sense, including algebraic, categorical and topological approaches.

Representative topics mentioned in this year's call for papers include: abstract data types, automated deduction, concurrency, constructive mathematics, data base theory, finite model theory, knowledge representation, lambda and combinatory calculi, logical aspects of computational complexity, logics in artificial intelligence, logic programming, modal and temporal logics, program logic and semantics, rewrite rules, software specification, type systems, and verification.

The 42 contributed papers in this volume were selected by the program committee from a total of 174 submissions; several additional submissions arrived too late to be considered. Selection criteria included originality, quality, and relevance to computer science. Each extended abstract was mailed to six members of the program committee. Some members of the committee chose to consult additional reviewers whose names are listed on the following page. Reviews, whenever available, were sent to all submitting authors.

Although LICS submissions were read carefully, conference selection is not a formal refereeing process. Many of the papers describe ongoing research, and it is anticipated that authors will publish more polished and complete versions in scientific journals.

On behalf of the program committee, I thank all authors who chose to submit their papers to LICS '92. Many excellent submissions could not be accepted because of size limitations on the symposium. I would also like to thank members of the program committee and the additional reviewers for their untiring efforts in reading and evaluating the large number of excellent submissions received this year. I would like to thank Dale Miller for his advice in selecting the primary reviewers and his help in organizing electronic data. I would also like to thank the Institute for Research in Cognitive Science for providing me with office space and secretarial help. Further, I wish to thank Chris Sandy, Janet Burns, and Renee Zawacki for managing information generated in cataloging and evaluating so many papers.

Andre Scedrov
1992 Program Chair

Foreword

This volume is the Proceedings of the Seventh Annual IEEE Symposium on Logic in Computer Science (LICS). The symposium encourages international participation of computer scientists influenced by mathematical logic and of logicians influenced by computer science. Previous LICS symposia were held in Cambridge, Massachusetts; Ithaca, New York; Edinburgh, Scotland; Asilomar, California; Philadelphia, Pennsylvania, and Amsterdam, The Netherlands–each time attracting several hundred enthusiastic participants. The Eighth LICS is scheduled for June 20-23, 1993, in Montreal, Canada.

LICS'92 is cosponsored by the IEEE-TC on Mathematical Foundations of Computing, and the University of California at Santa Cruz in cooperation with the Association for Computing Machinery-SIGACT, the Association for Symbolic Logic, and the European Association for Theoretical Computer Science.

LICS'92 has been subsidized by

Institutional Sponsors

Cornell University
Cornell-Xerox Design Research Institute
IBM Research
University of California at Santa Cruz

Donations by these Sponsors make it possible for the LICS Organizers to subsidize student attendance, student author awards, invited speakers, and attendance by researchers without other travel grants.

On behalf of the Organizing Committee and all the LICS'92 participants, I sincerely thank these sponsors for their donations. I also thank the Program Chair, Andre Scedrov, the Conference Chair, Phokion Kolaitis, and the Publicity Chair, Daniel Leivant, for their many months of effort. We look forward to another fruitful symposium.

Robert L. Constable
LICS General Chair

Ithaca, NY
April 1992

CONFERENCE ORGANIZATION

General Chair:

Prof. Robert L. Constable
Cornell University

1992 Program Chair:

Prof. Andre Scedrov
University of Pennsylvania

Publicity Chair:

Prof. Daniel Leivant
Indiana University

Organizing Committee:

Martin Abadi
Sergei Artemov
Jon Barwise
Manuel Blum
Samuel Buss
Edmund Clarke
Robert L. Constable (Chair)
Erwin Engeler
Jean Gallier
Ursula Goltz
Yuri Gurevich
Susumu Hayashi
Gerard Huet
Giles Kahn
Deepak Kapur
Rao Kosaraju
Jan-Willem Klop
Phokion Kolaitis
Daniel Leivant
Albert R. Meyer
Grigori Mints
John Mitchell
Yiannis Moschovakis
Rohit Parikh
Andrew Pitts
Gordon D. Plotkin
Simona Ronchi Della Rocca
Grezgorz Rozenberg
Andre Scedrov
Dana Scott
Jerzy Tiuryn
Moshe Vardi
Roel de Vrijer

Program Committee:

Egon Boerger
Rance Cleaveland
Steeve Cook
Nachum Dershowitz
Jean-Yves Girard
Rob von Glabbeek
Susumu Hayashi
John Hughes
Neil Jones
Jean-Louis Lassez
Eugenio Moggi
Anil Nerode
Fernando Pereira
Andre Scedrov (Chair)
Dana Scott
Andrezej Tarlecki
Moshe Vardi

ADDITIONAL REFEREES

Martin Abadi
Serge Abiteboul
Samson Abramsky
Luca Aceto
Miklos Ajtai
Nils Andersen
Peter Andrews
Krzysztof Apt
Lennart Augustsson
Leo Bachmair
Marek Bednarczyk
C. Beierle
Marcin Bialasik
H. Blair
Andreas Blass
Bard Bloom
Alexander Bockmayr
Anders Bondorf
Alexandre Boudet
Gerard Boudol
Robert S. Boyer
Val Breazu-Tannen
A. Brodsky
Steve Brookes
Antonio Bucciarelli
Peter Buneman
Hans Büning
Samuel Buss
M. Cerioli
Ed Clarke
Hubert Comon
Kevin Compton
G. Costa
Pierre-Louis Curien
Max Dauchet
J. Dix
Peter Dybjer
Hans Dybkjær
H.D. Ebbinghaus
Allen Emerson
Amy Felty
Tim Fernando
Jörg Flum
Nissim Francez
Tim Freeman
Peter Freyd
Jean Gallier
G. Germano
A. Giovini
S. Gnesi
A. Goerdt
E. Grädel
Etienne Grandjean
Tim Griffin
Jeremy Gunawardena
Elsa Gunter
Carl Gunter
Yuri Gurevich
Masami Hagiya
Joe Halpern
John Hannan
Bob Harper
Ryu Hasegawa
Nevin Heintze
Fritz Henglein
Hiromi Hiraishi
Sachio Hirokawa
Alain Hui Bon Hoa
Carsten Holst
Douglas Howe
Jieh Hsiang
Sebastian Hunt
Hans Hüttel
Neil Immerman
David Israel
J. Jaffar
G. Jáhrt
Jesper Jörgesen
Jean-Pierre Jouannaud
C. Jutla
Yukiyoshi Kameyama
Samuel N. Kamin
Bruce Kapron
Shmuel Katz
Henry Kautz
Delia Kesner
Jan Willem Klop
Satoshi Kobayashi
Joost Kok
Beata Konikowska
Kurt Konolige
Sarit Kraus
F. Kröger
Ryszard Kubiak
Ken Kunen
T.K. Lakshman
John Launchbury
Insup Lee
Peter Lee
A. Leitsch
Daniel Leivant
Xavier Leroy
Vladimír Lifschitz
Fangzhen Lin
Patrick Lincoln
James Lipton
Ewing Lusk
M. Maher
P. Mancarella
Leo Marcus
Witek Marek
K. Marriott
M. Martelli
Simone Martini
Michel Mauny
Brian Mayoh
Dale Miller
John Mitchell
Torben Mogensen
Andy Moran
Hiroshi Nakano
Sin Nisizaki
Mitsuhiro Okada
Hiroakira Ono
C. Palamidessi
Prakash Panangaden
Christine Paulin
Wojtek Penczek
Frank Pfenning

Toni Pitassi
David Plaisted
Amir Pnueli
Vaughan Pratt
Charles Rackoff
I.V. Ramakrishnan
Uday S. Reddy
G. Reggio
Didier Remy
John Reynolds
Michael Richter
Jon Riecke
Mikael Rittri
Piet Rodenburg
Kristoffer Rose
Mads Rosendahl
Dean Rosenzweig

Muli Safra
Vijay Saraswat
P. Schmitt
Peter Patel-Schneider
W. Schönfeld
Helmut Schwichtenberg
S. Seibert
R.C. Sekar
Bart Selman
Peter Sesoft
Wayne Snyder
Harald Söndergaard
Klaus Ambos-Spies
D. Spreen
Marian Srebrny
Eugene Stark
Rick Statman

Mark Stickel
Makoto Tatsuta
Wolfgang Thomas
Rich Tomason
Mads Tofte
Howard Wong-Toi
Yoshihito Toyama
Alasdair Urquhart
Fer-Jan de Vries
Scott Weinstein
Daniel Weise
Benjamin Werner
R. Yap
Mariko Yasugi
Amy Zwarico

Table of Contents

Session XII

SESSION 1:

CHAIR: JOHN MITCHELL

Third Order Matching is Decidable

Gilles Dowek
INRIA-Rocquencourt,
B.P. 105, 78153 Le Chesnay CEDEX, France
dowek@margaux.inria.fr
and
School of Computer Science,
Carnegie Mellon University,
Pittsburgh PA15213-3890, U.S.A.
gdowek@cs.cmu.edu

Abstract

The higher order matching *problem is the problem of determining whether a term is an instance of another in the simply typed λ-calculus, i.e. to solve the equation $a = b$ where a and b are simply typed λ-terms and b is ground. The decidability of this problem is still open. We prove the decidability of the particular case in which the variables occurring in the term a are at most third order.*

Introduction

The *higher order matching* problem is the problem of determining whether a term is an instance of another in the simply typed λ-calculus i.e. to solve the equation $a = b$ where a and b are simply typed λ-terms and b is ground.

Pattern matching algorithms are used to check if a proposition can be deduced from another by elimination of universal quantifiers or by introduction of existential quantifiers. In automated theorem proving, elimination of universal quantifiers and introduction of existential quantifiers are mixed and full unification is required, but in proof-checking and semi-automated theorem proving, these rules can be applied separately and thus pattern matching can be used instead of unification.

The higher order matching is conjectured decidable in [6] and the problem is still open. In [5] [6] [7] Huet has given a semi-decision algorithm and shown that in the particular case in which the variables occurring in the term a are at most second order this algorithm terminates, and thus that second order matching is decidable. In [9] Statman has reduced the conjecture to the λ-definability conjecture and in [10] Wolfram has given an always terminating algorithm whose completeness is conjectured.

We prove in this paper that third order matching is decidable i.e. we give an algorithm that decides if a matching problem, in which all the variables are at most third order, has a solution. The main idea is that if the problem $a = b$ has a solution then it also has a solution whose depth is bounded by some integer s depending only on the problem $a = b$, so a simple enumeration of the substitutions whose depth is bounded by s gives a decision algorithm. This result can also be used to bound the depth of the search tree in Huet's semi-decision algorithm and thus turn it into an always terminating algorithm.

At last we discuss the problems that occur when we try to generalize the proof given here to higher order matching.

0-8186-2735-2/92 $3.00 © 1992 IEEE

1 Trees and Terms

1.1 Trees

Definitions 1 (Following [3]) An *occurrence* is a list of strictly positive integers. A *tree domain* D is a non empty finite set of occurrences such that if $\alpha[n] \in D$ then $\alpha \in D$ and if also $n \neq 1$ then $\alpha[n-1] \in D$. A *tree* is a function from a tree domain D to a set L, called the set of labels of the tree.

If T is a tree and D its domain, the occurrence $[\,]$ is called the *root* of T and the occurrence $\alpha[n]$ is called the n^{th} *son* of the occurrence α. The *number of sons* of an occurrence α is the greatest integer n such that $\alpha[n] \in D$. A *leaf* is an occurrence that has no sons.

Let T be a tree and $\alpha = [s_1;...;s_n]$ an occurrence in this tree, the *path* of α is the set of occurrences $\{[s_1;...;s_p] \mid p \leq n\}$. The number of elements of this path is the length of α plus one.

The *depth* of the tree T is the length of the longest occurrence in D. This occurrence is, of course, a leaf.

If a is a label and $T_1,...,T_n$ are trees (of domains $D_1,...,D_n$) then the *tree of root a and sons $T_1,...,T_n$* is the tree T of domain $D = \{[\,]\} \cup \bigcup_i \{[i]\alpha \mid \alpha \in D_i\}$ such that

$$T([\,]) = a$$

and

$$T([i]\alpha) = T_i(\alpha)$$

If T is a tree of domain D and α is an occurrence of D, the *subtree* T/α is the tree T' whose domain is $D' = \{\beta \mid \alpha\beta \in D\}$ and such that

$$T'(\beta) = T(\alpha\beta)$$

If T is a tree of domain D, α an occurrence of D and T' a tree of domain D' then the *graft* of T' in T at the occurrence α $(T[\alpha \leftarrow T'])$ is the tree T'' of domain $D'' = D - \{\alpha\beta \mid \alpha\beta \in D\} \cup \{\alpha\beta \mid \beta \in D'\}$ and such that

$$T''(\gamma) = T'(\beta) \text{ if } \gamma = \alpha\beta$$

and

$$T''(\gamma) = T(\gamma) \text{ otherwise}$$

Let T and T' be trees, and a a label such that all the occurrences of a in T are leaves $\alpha_1, ..., \alpha_n$ then the *substitution* of T' for a in T $(T[a \leftarrow T'])$ is defined as $T[\alpha_1 \leftarrow T']...[\alpha_n \leftarrow T']$. Remark that since $\alpha_1, ..., \alpha_n$ are leaves, the order in which the grafts are performed is insignificant.

1.2 Types

Definition 2 Let us consider a finite set $\mathcal{T}$. The elements of $\mathcal{T}$ are called *atomic types*. A *type* is a tree whose labels are either the elements of $\mathcal{T}$ or $\rightarrow$ and such that the occurrences labeled by an element of $\mathcal{T}$ are leaves and the ones labeled by $\rightarrow$ have two sons.

Let T be a type, if the root of T is an atomic type U then T is written U, if the root of T is $\rightarrow$ and its sons are written T_1 and T_2 then T is written $(T_1 \rightarrow T_2)$. By convention $T_1 \rightarrow T_2 \rightarrow T_3$ is an abbreviation for $(T_1 \rightarrow (T_2 \rightarrow T_3))$.

Definition 3 If T is a type, the *order* of T is defined by
- $o(T) = 1$ if T is atomic,
- $o(T_1 \rightarrow T_2) = max\{1 + o(T_1), o(T_2)\}$.

1.3 Typed λ-terms

Definitions 4 For each type T we consider three sets $\mathcal{U}_T$, $\mathcal{L}_T$, $\mathcal{E}_T$. The elements of $\mathcal{U}_T$ are called *universal variables* of type T, those of $\mathcal{L}_T$ *local variables* of type T and those of $\mathcal{E}_T$ *existential variables* of type T. We assume that we have in each atomic type at least a universal variable and that there is a finite number of universal variables i.e. that the set $\bigcup_T \mathcal{U}_T$ is finite. We assume also that we have in each type an infinite number of local and existential variables.

A typed λ-term is a tree whose labels are either *App*, $< Lam, x >$ where x is a local variable or $< Var, x >$ where x is a universal, local or existential variable, such that the occurrences labeled by *App* have two sons, the occurrences labeled $< Lam, x >$ have one son and the occurrences labeled $< Var, x >$ are leaves.

Let t be a term, if the root of t is $< Var, x >$ we write it x, if the root of t is $< Lam, x >$ and its son is written u then we write it $[x : T]u$ where T is the type of x, if the root of t is *App* and its sons are written u and v then we write it $(u\ v)$. By convention $(u\ v\ w)$ is an abbreviation for $((u\ v)\ w)$.

In a term t, an occurrence α labeled by $< Var, x >$ is *bound* if there exists an occurrence β in the path of α labeled by $< Lam, x >$, it is *free* otherwise.

A term is *ground* if no occurrence is labeled by a pair $< Var, x >$ with x existential.

Let t and t' be terms and x be a variable, the *substitution* of t' for x in t $(t[x \leftarrow t'])$ is defined as $t[< Var, x > \leftarrow t']$.

Definition 5 Type of a term

A term t is said to have the type T if either:
- t is a variable (universal, local or existential) of type T.
- $t = (u\ v)$ and u has type $U \rightarrow T$ and v type U for some type U,
- $t = [x : U]u$, the term u has type V and $T = U \rightarrow V$.

A term t is said to be *well-typed* if there exists a type T such that t has type T. In this case T is unique and is called *the type of* t.

Definition 6 The $\beta\eta$-reduction is defined as smallest transitive relation, compatible with term structure such that

$$([x : T]t\ u) \rhd t[x \leftarrow u]$$

$$[x : T](t\ x) \rhd t \quad \text{if } x \text{ is not free in } t$$

We adopt the usual convention of considering terms up to α-conversion (i.e. bound variable renaming) and we consider that bound variables are renamed to avoid capture during substitutions. A rigorous presentation would use de Bruijn indices [2].

Proposition 1 The $\beta\eta$-reduction relation is strongly normalizable and confluent on typed terms, and thus each term has a unique normal form.
Proof See, for instance, [4].

Proposition 2 Let t be a normal well-typed term of type $T_1 \rightarrow ... \rightarrow T_n \rightarrow T$ (T atomic), the term t has the form

$$t = [y_1 : T_1]...[y_m : T_m](x\ u_1\ ...\ u_p)$$

where $m \leq n$ and x is a variable.
Proof The term t can be written in a unique way $t = [y_1 : U_1]...[y_m : U_m]u$ where u is not an abstraction. The term u can be written in a unique way $u = (v\ u_1\ ...\ u_p)$ where v is not an application. The term v is not an application by definition, it is not an abstraction (if $p = 0$ because u is not an abstraction and if $p \neq 0$ because t is normal), it is therefore a variable. Then for type reasons $m \leq n$ and for all i, $U_i = T_i$.

Definition 7 If $t = [y_1 : T_1]...[y_m : T_m](x\ u_1\ ...\ u_p)$ is a term of type $T = T_1 \rightarrow ... \rightarrow T_n \rightarrow T$ (T atomic) $(m \leq n)$ which is in $\beta\eta$-normal form then we define its *β-normal η-long* form as the term

$$t' = [y_1 : T_1]...[y_m : T_m][y_{m+1} : T_{m+1}]...[y_n : T_n] \\ (x\ u'_1\ ...\ u'_p\ y'_{m+1}\ ...\ y'_n)$$

where u_i' is the β-normal η-long form of u_i and y_i' is the β-normal η-long form of y_i.

This definition is by induction on the pair $< c_1, c_2 >$ where c_1 is the number of occurrences in t and c_2 the number of occurrences in T

In the following all the terms are assumed to be on β-normal η-long form.

1.4 Böhm Trees

Definition 8 Böhm Tree

A (finite) *Böhm tree* is a tree whose occurrences are labeled by pairs $< l, x >$ such that l is a list of local variables $[y_1; ...; y_n]$ and x is a variable and the number of sons of an occurrence labeled by $< l, x >$ is the arity of x i.e. the integer p such that the type of x has the form $T_1 \to ... \to T_p \to T$ with T atomic.

Definition 9 Type of a Böhm Tree

Let t be a Böhm tree whose root is labeled by the pair $< [y_1; ...; y_n], x >$ and whose sons are $u_1, ..., u_p$. The Böhm tree t is said to have the type T if the Böhm trees $u_1, ..., u_p$ have type $U_1, ..., U_p$ the variable x has type $U_1 \to ... \to U_p \to U$ and $T = T_1 \to ... \to T_n \to U$ where $T_1, ..., T_n$ are the types of the variables $y_1, ..., y_n$.

A Böhm tree t is said to be *well-typed* if there exists a type T such that t has type T. In this case T is unique and is called *the type of* t.

Definition 10 Let t be a λ-term in normal form. We write $t = [y_1 : T_1]...[y_n : T_n](x\ u_1\ ...\ u_p)$. The *Böhm tree* of t is the tree whose root is the pair $< l, x >$ where $l = [y_1; ...; y_n]$ is the list of the variables bound at the top of this term, x is the head variable of t and sons are the Böhm trees of $u_1, ..., u_p$.

Remark Normal well-typed terms and well-typed Böhm trees are in one-to-one correspondence. Moreover if t is a normal term and $\tilde{t}$ is its Böhm tree then occurrences in t labeled by a variable and occurrences in $\tilde{t}$ are in one-to-one correspondence.

So we will use the following abuse of notation: if α is an occurrence in the Böhm tree of t we write (t/α) for the normal term corresponding to the Böhm tree $(\tilde{t}/\alpha)$ and $t[\alpha \leftarrow u]$ for the term $t[\alpha' \leftarrow u]$ where α' is the variable occurrence in t corresponding to α.

Definition 11 Let t be a term, we write $|t|$ for the depth of the Böhm tree of the normal form of t.

Proposition 3 In each type T there is a ground term t such that $|t| = 0$.

Proof Let $T = U_1 \to ... \to U_n \to U$ with U atomic. Let x be a universal variable of type U. The term $t = [y_1 : U_1]...[y_n : U_n]x$ has type T and its depth is 0.

1.5 Substitution

Definition 12 A *substitution* is a finite set of pairs $< x_i, t_i >$ where x_i is an existential variable and t_i a term of the same type in which no local variable occurs free such that if $< x, t >$ and $< x, t' >$ are both in this set then $t = t'$. The variables x_i are said to be *bound* by the substitution.

Definition 13 If σ is a substitution and t a term then we let

$$\sigma t = t[\alpha_1^1 \leftarrow t_1]...[\alpha_1^{p_1} \leftarrow t_1]...[\alpha_n^1 \leftarrow t_n]...[\alpha_n^{p_n} \leftarrow t_n]$$

where $\alpha_i^1, ..., \alpha_i^{p_i}$ are the occurrences of x_i in t.

Remark that since the α_i^j are leaves, the order in which the grafts are performed is insignificant.

Definition 14 Let σ and τ be two substitutions the substitution $\tau \circ \sigma$ is defined by

$$\tau \circ \sigma = \{< x, \tau t > \mid < x, t > \in \sigma\} \cup \{< x, t > \mid < x, t > \in \tau \text{ and } x \text{ not bound by } \sigma\}$$

Proposition 4 Let σ and τ be two substitutions and t is a term, we have

$$(\tau \circ \sigma)t = \tau(\sigma t)$$

Proof By decreasing induction on the depth of an occurrence α in t we prove that we have

$$(\tau \circ \sigma)(t/\alpha) = \tau(\sigma(t/\alpha))$$

2 Pattern Matching

Definition 15 Matching Problem

A *matching problem* is a pair of well-typed terms $< a, b >$ where a and b have the same type and b is ground.

Definition 16 Third Order Matching Problem

A *third order matching problem* is a matching problem $< a, b >$ such that the types of the existential variables that occur in a are of order at most three.

Definition 17 Solution

Let $a = b$ be a matching problem. A substitution σ is a *solution* of this problem if and only if the normal forms of the terms σa and b are identical up to α-conversion.

Remark Usual unification terminology distinguishes *variables* (here existential variables) and *constants* (here universal variables). The need for local variables comes from the fact that we want to transform the problem $[y : T]x = [y : T]y$ (where x is an existential variable of type T) into the problem $x = y$ by dropping the common abstraction. The symbol y cannot be an existential variable because it cannot be instanciated by a substitution, it cannot be a universal variable because, if it were, we would have the solution to the second problem $x \leftarrow y$ which is not a solution to the first. So we let y be a local variable and the solution $x \leftarrow y$ is now forbidden in both problems because no local variable can occur free in the terms substituted to variables in a substitution.

In Huet's unification algorithm [5] [6] these local variables are always kept in the head of the terms in common abstractions. In Miller's mixed prefixes terminology [8], these local variables are universal variables declared to the right of all the existential variables.

Definition 18 Ground Solution

Let $a = b$ be a problem and σ a solution to $a = b$. The solution σ is *ground* if for each existential variable that has an occurrence in a, the term σx is ground.

Proposition 5 If a matching problem has a solution then it has a ground solution.

Proof Let $a = b$ be a matching problem and σ a solution. Let $y_1 : T_1, ..., y_n : T_n$ the existential variables occurring in the σx for x existential variable of a. Let $u_1, ..., u_n$ be ground terms of the types $T_1, ..., T_n$. Let $\tau = \{< y_1, u_1 >, ..., < y_n, u_n >\}$, and $\sigma' = \tau \circ \sigma$. Obviously, for each existential variable x of a, the term $\sigma' x$ is ground. And $\sigma' a = \tau\sigma a = \tau b = b$. So the problem $a = b$ has a ground solution.

Definition 19 Complete Set of Solutions

Obviously if σ is a solution to a problem $a = b$ then $\tau \circ \sigma$ is one too. A set S of solutions to a problem $a = b$ is said to be *complete* if for every substitution θ solution to this problem there exists a substitution $\sigma \in S$ and a substitution τ such that $\theta = \tau \circ \sigma$.

Lemma 1 Some problems have no finite complete set of solutions.

Proof (Example 1) Consider an atomic type T and an existential variable $x : T \to (T \to T) \to T$. The problem

$$[a : T](x\ a\ [z : T]z) = [a : T]a$$

has an infinite number of independent minimal solutions

$$x \leftarrow [c : T][s : T \to T](s\ldots.\ (s\ c)\ \ldots\ )$$

So in contrast with second order matching [6] [7] there is no (always terminating) algorithm that enumerates all the independent minimal solutions of a third order matching problem.

We consider now algorithms that take as an input a matching problem and either give *one* solution to it or fail if it does not have any.

3 A Bound on the Depth of Solutions

The main idea in this paper is that when we have a matching problem $a = b$ and x is an existential variable occurring in a and t is the term substituted to x by some solution to the problem then the depth of the Böhm tree of t can be bounded by an integer s depending only on the problem $a = b$. Of course the previous example shows that a matching problem may have solutions of arbitrary depth, but to design a decision algorithm we do not need to prove that *all* the solutions are bounded by s but only that *at least one* is.

To show this result we take a problem $a = b$ that has a solution σ (by proposition 5, we can consider without loss of generality that this solution is ground) and we build another solution σ' whose depth is bounded by an integer s depending only on the problem $a = b$.

3.1 Key Lemma

Definition 20 Let $c = [z_1 : U_1]...[z_p : U_p]d$ be a normal term and i an integer, $i \leq p$. We say that c is *relevant* in its i^{th} argument if z_i has an occurrence in the term d.

Lemma 2 (Key Lemma) Let us consider a normal term u, a variable y of type T of order at most two and a normal ground term c of type T.

(1) If y has an occurrence in u then $|c| \leq |u[y \leftarrow c]|$.

(2) If α is an occurrence in the Böhm tree of u such that no occurrence in the path of α is labeled by y, then α is also an occurrence in the normal form of $u[y \leftarrow c]$ and has the same label in the Böhm tree of u and in the Böhm tree of the normal form of $u[y \leftarrow c]$.

(3) If $\alpha = [s_1; ...; s_n]$ is an occurrence in the Böhm tree of u such that for each occurrence $\beta = [s_1; ...; s_k]$ in the path of α, $\beta \neq \alpha$, labeled by y, the term c is relevant in its r^{th} argument where r is the position of the son of β in the path of α i.e. $r = s_{k+1}$, then there exists an occurrence α' of the Böhm tree of the normal form of $u[y \leftarrow c]$ such that all the labels occurring in the path of α, except y, occur in the path of α' and the number of times they occur in the path of α' is greater or equal to the number of times they occur in the path of α.

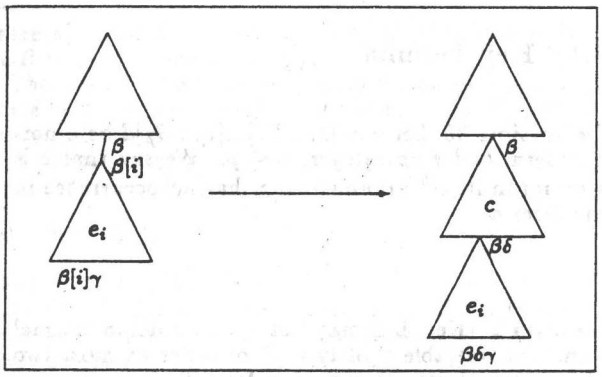
(4) Moreover if $|c| \neq 0$ then the length of α' is greater or equal to the length of α.

Proof By induction on the number of occurrences of y in u. We substitute these occurrences one by one and we normalize the term. Let β be the occurrence in the Böhm tree of u corresponding to the occurrence of y in u we substitute. Let us write

$$c = [z_1 : U_1]...[z_p : U_p]d$$

The term (u/β) has the form $(y\ e_1\ \ldots\ e_p)$. When we substitute y by the term c in $(y\ e_1\ \ldots\ e_p)$ we get $(c\ e_1\ \ldots e_p)$ and when we normalize this term we get the term $d[z_1 \leftarrow e_1, ..., z_p \leftarrow e_p]$ which is normal because the type of the e_i are first order.

Let us consider the occurrences in the Böhm tree of u, while substituting the occurrence of y corresponding to β, we have removed all the occurrences $\beta[i]\gamma$ where i is an integer $(i \leq p)$ and γ is an occurrence in the Böhm tree of e_i. We have added all the occurrences $\beta\delta$ where δ is an occurrence of the Böhm tree of c labeled by a variable different from $z_1, ..., z_p$ and all the occurrences $\beta\delta\gamma$ where δ is a leaf occurrence in the Böhm tree of c labeled by a z_i and γ is an occurrence of the Böhm tree of e_i.



(1) Let β be an outermost occurrence of y in the Böhm tree of u. For each occurrence δ in the Böhm tree of c, $\beta\delta$ is an occurrence in the Böhm tree of the normal form of $u[y \leftarrow c]$. So $|c| \leq |u[y \leftarrow c]|$.

(2) When an occurrence β of y is substituted by c all the occurrences removed have the form $\beta[i]\gamma$. So if no occurrence in the path of α is labeled by y, the occurrence α remain in the normal form of $u[y \leftarrow c]$.

(3) If the occurrence β is not in the path of α then the occurrence α is still an occurrence in the normal form of $u[y \leftarrow c]$, we take $\alpha' = \alpha$.

If $\beta = \alpha$ then the occurrence β is an occurrence of the Böhm tree of the normal form of $u[y \leftarrow c]$. We take $\alpha' = \beta = \alpha$.

If β is in the path of α and $\beta \neq \alpha$, $\beta = [s_1, ..., s_k]$ then let r be the position of the son of β in the path of α i.e. $r = s_{k+1}$. Let γ such that $\alpha = \beta[r]\gamma$. By hypothesis z_r has an occurrence in d, let δ be such an occurrence. The occurrence $\beta\delta\gamma$ is an occurrence in the Böhm tree of the normal form of $u[y \leftarrow c]$. We take $\alpha' = \beta\delta\gamma$.

In all the cases, all the labels occurring in the path of α, except y, occur in the path of α' and the number of times they occur in the path of α' is greater or equal to the number of times they occur in the path of α.

(4) If $\delta = [\,]$ then $c = [z_1 : U_1]...[z_p : U_p]z_r$ and $|c| = 0$. So if $|c| \neq 0$ then $\delta \neq [\,]$ and the length of α' is greater or equal to the length of α.

Corollary Let us consider a normal term u, a variable y of type T of order at most two and a ground term c of type T. If c is relevant in all its arguments and $|c| \neq 0$ then $|u| \leq |u[y \leftarrow c]|$.

Proof We take for α the longest occurrence in the Böhm tree of u. When we substitute one by one the occurrences of y, by part (4) of the key lemma, we get longer occurrences. So there is an occurrence in the Böhm tree of the normal form of $u[y \leftarrow c]$ which is longer than α. So $|u| \leq |u[y \leftarrow c]|$.

3.2 Constraints on the Substitution σ'

First we are going to express some equational constraints that the substitution σ' must verify in order to be a solution to the problem $a = b$. Of course the equation $a = b$ answers the problem, but we need simpler ones: our equations will be on the form $(x\ c_1\ ...\ c_n) = b'$ where x is a variable and $c_1, ..., c_n, b'$ are ground terms.

Definition 21 Let $a = b$ be a problem and σ a (ground) solution to this problem. By induction on the number of occurrences of a we construct a set of equations $\Xi(a = b, \sigma)$.

- If $a = [x : T]d$ then since σ is a solution to the problem $a = b$ we have $b = [x : T]e$ and σ is a solution to the problem $d = e$. We let

$$\Xi(a = b, \sigma) = \Xi(d = e, \sigma)$$

- If $a = (f\ d_1\ ...\ d_n)$ with f universal or local then since σ is a solution to $a = b$ we have $b = (f\ e_1\ ...\ e_n)$ and σ is a solution to the problems $d_i = e_i$. We let

$$\Xi(a = b, \sigma) = \bigcup_i \Xi(d_i = e_i, \sigma)$$

- If $a = (x\ d_1\ ...\ d_n)$ with x existential then for all i such that z has an occurrence in the normal form of the term $(\sigma x\ \sigma d_1\ ...\ \sigma d_{i-1}\ z\ \sigma d_{i+1}\ ...\ \sigma d_n)$ we let $c_i = \sigma d_i$ and $H_i = \Xi(d_i = \sigma d_i, \sigma)$ (obviously σ is a solution to $d_i = \sigma d_i$). Otherwise we let $c_i = z_i$ where z_i is a new local variable and $H_i = \emptyset$. We let

$$\Xi(a = b, \sigma) = \{(x\ c_1\ ...\ c_n) = b\} \cup \bigcup_i H_i$$

Proposition 6 Let $t = (x\ d_1\ ...\ d_n)$ be a term and σ be a substitution. Let $c_i = \sigma d_i$ if z has an occurrence in $(\sigma x\ \sigma d_1\ ...\ \sigma d_{i-1}\ z\ \sigma d_{i+1}\ ...\ \sigma d_n)$ and $c_i = z_i$ new local variable of the same type as d_i otherwise. The variables z_i do not occur in the normal form of $(\sigma x\ c_1\ ...\ c_n)$.

Proof Let us assume that some of these variables have an occurrence in the normal form of $(\sigma x\ c_1\ ...\ c_n)$ and consider outermost occurrence of such a variable z_i in the Böhm tree of the normal form of $(\sigma x\ c_1\ ...\ c_n)$. By part (2) of the key lemma, the variable z_i has also an occurrence in the normal form of term $(\sigma x\ c_1\ ...\ c_n)[z_j \leftarrow \sigma d_j \mid j \neq i]$ i.e. in the normal form of the term $(\sigma x\ \sigma d_1\ ...\ \sigma d_{i-1}\ z_i\ \sigma d_{i+1}\ ...\ \sigma d_n)$, which is contradictory.

Proposition 7 Let $a = b$ be an equation and σ a solution to this equation,

- the substitution σ is a solution to the equations of the set $\Xi(a = b, \sigma)$,
- conversely if σ' is a solution to the equations of $\Xi(a = b, \sigma)$ then σ' is also a solution to the problem $a = b$.

Proof

- By induction on the number of occurrences of a. When a is an abstraction (resp. an atomic term whose head is universal or local) then by induction hypothesis σ is a solution to all the equations of the set $\Xi(d = e, \sigma)$ (resp. $\Xi(d_i = e_i, \sigma)$), so it is a solution to all the equations of $\Xi(a = b, \sigma)$.

When $a = (x\ d_1\ ...\ d_n)$ then by induction hypothesis σ is a solution to all the equations of the $H_i's$ and