LECTURE NOTES IN COMPUTER SCIENCE

71

# Automata, Languages and Programming

**Sixth Colloquium**

**July 16-20, 1979**

**Edited by Hermann A. Maurer**

# Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis



# Automata, Languages and Programming

Sixth Colloquium, Graz, Austria, July 16–20, 1979



Edited by Hermann A. Maurer

Springer-Verlag
Berlin Heidelberg New York 1979

**Editor**
Hermann A. Maurer
Institut für Informations-
verarbeitung
Technische Universität Graz
Steyrergasse 17
8010 Graz/Austria

AMS Subject Classifications (1970): 68-XX
CR Subject Classifications (1974): 4.1, 4.2, 5.2, 5.3

ISBN 3-540-09510-1 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-09510-1 Springer-Verlag New York Heidelberg Berlin

Library of Congress Cataloging in Publication Data.
Colloquium on Automata, Languages and programming, 6th, Graz, 1979. Automata, languages and programming. (Lecture notes in computer science ; 71) Includes index.
1. Sequential machine theory--Congresses. 2. Formal languages--Congresses,
3. Programming languages (Electronic computers)--Congresses. I. Maurer,
Hermann A., 1941- II. Title. III. Series.
QA267.5S4C63 1979 001.6'42 79-15859



Printing and binding: Beltz Offsetdruck, Hemsbach/Bergstr.
2145/3140-543210

## PREFACE



The Sixth Colloquium on Automata, Languages and Programming (ICALP 79) was preceded by similar colloquia in Paris (1972), Saarbrücken (1974), Edinburgh (1976), Turku (1977) and Udine (1978), all sponsored by EATCS (European Association for Theoretical Computer Science).

Of a total of 139 papers submitted to ICALP 79, fifty papers were selected. Together with three invited presentations they are contained in this volume.

The program committee of ICALP 79 consisted of G.Ausiello, W.Brauer, K.Culik II, J. de Bakker, E.Engeler, S.Even, M.Harrison, I.M.Havel, J.Hopcroft, G.Hotz, W.Kuich, H.Maurer (chairman), M.Nivat, M.Paterson, Z.Pawlak, A.Salomaa, D.Wood, H.Zima.

As conference chairman, I would like to thank the members of the program committee for their hard work in evaluating the submitted papers. Special thanks are also due to the following referees who helped in the refereeing process: A.Aho, J.Albert, G. Andrews, K.R.Apt, J.Archer, E.A.Ashcroft, L.Banachowski, G.Baron, C.Batini, J.C.Beatty, J.Becvar, D.Bini, A.Blikle, C.Böhm, R.Book, S.Breidbart, A. de Bruin, J.Brzozowski, R.Cartwright, L.M.Chirica, R.S.Cohen, A.B.Cremers, P.Dembinski, K.Ecker, H.Ehrig, M. Furst, G.Gati, G.Goos, J.Gorski, M.Grabowski, D.Gries, J.Gruska, J.Grzymala-Busse, V. Haase, M.Hofri, M.Jazayeri, J.Karhumäki, O.Kariv, M.Karpinski, C.Keleman, B.Konilcowska, A.Krecmar, H.P.Kriegel, F.Krieger, M.Lao, R.Leipälä, M.Linna, F.Luk, G.Mahr, T.S.E. Maibaum, J.Maluszynski, A.Marchetti-Spaccamela, A.Mazurkiewicz, L.G.L.T.Meertens, R. Milner, A.Moura, T.Müldner, K.Müller, E.J.Neuhold, A.Obtulowicz, J.Opatrny, Th.Ottmann, D.M.R.Park, A.Paz, M.Penttonen, A.Pettorossi, F.Plâsil, H.Prodinger, V.Rajlich, P.Raulefs, J.C.Reynolds, J.L.Richier, M.Rodeh, W.P.de Roever, F.Romani, D.Rotem, P. Ruzicka, A.Salwicki, G.Schlageter, F.Schneider, E.Shamir, J.Simon, M.Steinby, W.Stucky, S.Termini, J.W.Thatcher, F.J.Urbanek, V.K.Vaishnavi, P.van Emde Boas, J.van Leeuwen, J.Weglarz, L.Wegner, K.Weihrauch, J.Winkowski, C.K.Yap.

Finally, the support of the Austrian Federal Ministry for Science and Research, the Province of Styria, the City of Graz, the Research Center Graz, IBM Austria, Sperry Univac, the Institut f. Angewandte Informatik und Formale Beschreibungsverfahren - Universität Karlsruhe and the Technical University of Graz is gratefully acknowledged. Last not least, I want to thank the members of the organizing committee and my secretary Mrs. A.Kleinschuster for their help in organizing the conference, and Springer-Verlag for excellent cooperation concerning the publication of this volume.

Graz, April 1979

Hermann Maurer

# Lecture Notes in Computer Science

Vol. 1: GI-Gesellschaft für Informatik e.V. 3. Jahrestagung, Hamburg, 8.–10. Oktober 1973. Herausgegeben im Auftrag der Gesellschaft für Informatik von W. Brauer. XI, 508 Seiten. 1973.

Vol. 2: GI-Gesellschaft für Informatik e.V. 1. Fachtagung über Automatentheorie und Formale Sprachen, Bonn, 9.–12. Juli 1973. Herausgegeben im Auftrag der Gesellschaft für Informatik von K.-H. Böhling und K. Indermark. VII, 322 Seiten. 1973.

Vol. 3: 5th Conference on Optimization Techniques, Part I. (Series: I.F.I.P. TC7 Optimization Conferences.) Edited by R. Conti and A. Ruberti. XIII, 565 pages. 1973.

Vol. 4: 5th Conference on Optimization Techniques, Part II. (Series: I.F.I.P. TC7 Optimization Conferences.) Edited by R. Conti and A. Ruberti. XIII, 389 pages. 1973.

Vol. 5: International Symposium on Theoretical Programming. Edited by A. Ershov and V. A. Nepomniaschy. VI, 407 pages. 1974.

Vol. 6: B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, Matrix Eigensystem Routines – EISPACK Guide. XI, 551 pages. 2nd Edition 1974. 1976.

Vol. 7: 3. Fachtagung über Programmiersprachen, Kiel, 5.–7. März 1974. Herausgegeben von B. Schlender und W. Friedinghaus. VI, 225 Seiten. 1974.

Vol. 8: GI-NTG Fachtagung über Struktur und Betrieb von Rechensystemen, Braunschweig, 20.–22. März 1974. Herausgegeben im Auftrag der GI und der NTG von H.-O. Leilich. VI, 340 Seiten. 1974.

Vol. 9: GI-BIFOA Internationale Fachtagung: Informationszentren in Wirtschaft und Verwaltung. Köln, 17./18. Sept. 1973. Herausgegeben im Auftrag der GI und dem BIFOA von P. Schmitz. VI, 259 Seiten. 1974.

Vol. 10: Computing Methods in Applied Sciences and Engineering, Part 1. International Symposium, Versailles, December 17–21, 1973. Edited by R. Glowinski and J. L. Lions. X, 497 pages. 1974.

Vol. 11: Computing Methods in Applied Sciences and Engineering, Part 2. International Symposium, Versailles, December 17–21, 1973. Edited by R. Glowinski and J. L. Lions. X, 434 pages. 1974.

Vol. 12: GFK-GI-GMR Fachtagung Prozessrechner 1974. Karlsruhe, 10.–11. Juni 1974. Herausgegeben von G. Krüger und R. Friehmelt. XI, 620 Seiten. 1974.

Vol. 13: Rechnerstrukturen und Betriebsprogrammierung, Erlangen, 1970. (GI-Gesellschaft für Informatik e.V.) Herausgegeben von W. Händler und P. P. Spies. VII, 333 Seiten. 1974.

Vol. 14: Automata, Languages and Programming – 2nd Colloquium, University of Saarbrücken, July 29–August 2, 1974. Edited by J. Loeckx. VIII, 611 pages. 1974.

Vol. 15: L Systems. Edited by A. Salomaa and G. Rozenberg. VI, 338 pages. 1974.

Vol. 16: Operating Systems, International Symposium, Rocquencourt 1974. Edited by E. Gelenbe and C. Kaiser. VIII, 310 pages. 1974.

Vol. 17: Rechner-Gestützter Unterricht RGU '74, Fachtagung, Hamburg, 12.–14. August 1974, ACU-Arbeitskreis Computer-Unterstützter Unterricht. Herausgegeben im Auftrag der GI von K. Brunnstein, K. Haefner und W. Händler. X, 417 Seiten. 1974.

Vol. 18: K. Jensen and N. E. Wirth, PASCAL – User Manual and Report. VII, 170 pages. Corrected Reprint of the 2nd Edition 1976.

Vol. 19: Programming Symposium. Proceedings 1974. V, 425 pages. 1974.

Vol. 20: J. Engelfriet, Simple Program Schemes and Formal Languages. VII, 254 pages. 1974.

Vol. 21: Compiler Construction, An Advanced Course. Edited by F. L. Bauer and J. Eickel. XIV. 621 pages. 1974.

Vol. 22: Formal Aspects of Cognitive Processes. Proceedings 1972. Edited by T. Storer and D. Winter. V, 214 pages. 1975.

Vol. 23: Programming Methodology. 4th Informatik Symposium, IBM Germany Wildbad, September 25–27, 1974. Edited by C. E. Hackl. VI, 501 pages. 1975.

Vol. 24: Parallel Processing. Proceedings 1974. Edited by T. Feng. VI, 433 pages. 1975.

Vol. 25: Category Theory Applied to Computation and Control. Proceedings 1974. Edited by E. G. Manes. X, 245 pages. 1975.

Vol. 26: GI-4. Jahrestagung, Berlin, 9.–12. Oktober 1974. Herausgegeben im Auftrag der GI von D. Siefkes. IX, 748 Seiten. 1975.

Vol. 27: Optimization Techniques. IFIP Technical Conference. Novosibirsk, July 1–7, 1974. (Series: I.F.I.P. TC7 Optimization Conferences.) Edited by G. I. Marchuk. VIII, 507 pages. 1975.

Vol. 28: Mathematical Foundations of Computer Science. 3rd Symposium at Jadwisin near Warsaw, June 17–22, 1974. Edited by A. Blikle. VII, 484 pages. 1975.

Vol. 29: Interval Mathematics. Procedings 1975. Edited by K. Nickel. VI, 331 pages. 1975.

Vol. 30: Software Engineering. An Advanced Course. Edited by F. L. Bauer. (Formerly published 1973 as Lecture Notes in Economics and Mathematical Systems, Vol. 81) XII, 545 pages. 1975.

Vol. 31: S. H. Fuller, Analysis of Drum and Disk Storage Units. IX, 283 pages. 1975.

Vol. 32: Mathematical Foundations of Computer Science 1975. Proceedings 1975. Edited by J. Bečvář. X, 476 pages. 1975.

Vol. 33: Automata Theory and Formal Languages, Kaiserslautern, May 20–23, 1975. Edited by H. Brakhage on behalf of GI. VIII, 292 Seiten. 1975.

Vol. 34: GI – 5. Jahrestagung, Dortmund 8.–10. Oktober 1975. Herausgegeben im Auftrag der GI von J. Mühlbacher. X, 755 Seiten. 1975.

Vol. 35: W. Everling, Exercises in Computer Systems Analysis. (Formerly published 1972 as Lecture Notes in Economics and Mathematical Systems, Vol. 65) VIII, 184 pages. 1975.

Vol. 36: S. A. Greibach, Theory of Program Structures: Schemes, Semantics, Verification. XV, 364 pages. 1975.

Vol. 37: C. Böhm, λ-Calculus and Computer Science Theory. Proceedings 1975. XII, 370 pages. 1975.

Vol. 38: P. Branquart, J.-P. Cardinael, J. Lewi, J.-P. Delescaille, M. Vanbegin. An Optimized Translation Process and Its Application to ALGOL 68. IX, 334 pages. 1976.

Vol. 39: Data Base Systems. Proceedings 1975. Edited by H. Hasselmeier and W. Spruth. VI, 386 pages. 1976.

Vol. 40: Optimization Techniques. Modeling and Optimization in the Service of Man. Part 1. Proceedings 1975. Edited by J. Cea. XIV, 854 pages. 1976.

Vol. 41: Optimization Techniques. Modeling and Optimization in the Service of Man. Part 2. Proceedings 1975. Edited by J. Cea. XIII, 852 pages. 1976.

Vol. 42: James E. Donahue, Complementary Definitions of Programming Language Semantics. VII, 172 pages. 1976.

Vol. 43: E. Specker und V. Strassen, Komplexität von Entscheidungsproblemen. Ein Seminar. V, 217 Seiten. 1976.

Vol. 44: ECI Conference 1976. Proceedings 1976. Edited by K. Samelson. VIII, 322 pages. 1976.

Vol. 45: Mathematical Foundations of Computer Science 1976. Proceedings 1976. Edited by A. Mazurkiewicz. XI, 601 pages. 1976.

Vol. 46: Language Hierarchies and Interfaces. Edited by F. L. Bauer and K. Samelson. X, 428 pages. 1976.

Vol. 47: Methods of Algorithmic Language Implementation. Edited by A. Ershov and C. H. A. Koster. VIII, 351 pages. 1977.

Vol. 48: Theoretical Computer Science, Darmstadt, March 1977. Edited by H. Tzschach, H. Waldschmidt and H. K.-G. Walter on behalf of GI. VIII, 418 pages. 1977.

Vol. 49: Interactive Systems. Proceedings 1976. Edited by A. Blaser and C. Hackl. VI, 380 pages. 1976.

Vol. 50: A. C. Hartmann, A Concurrent Pascal Compiler for Minicomputers. VI, 119 pages. 1977.

Vol. 51: B. S. Garbow, Matrix Eigensystem Routines – Eispack Guide Extension. VIII, 343 pages. 1977.

Vol. 52: Automata, Languages and Programming. Fourth Colloquium, University of Turku, July 1977. Edited by A. Salomaa and M. Steinby. X, 569 pages. 1977.

Vol. 53: Mathematical Foundations of Computer Science. Proceedings 1977. Edited by J. Gruska. XII, 608 pages. 1977.

Vol. 54: Design and Implementation of Programming Languages. Proceedings 1976. Edited by J. H. Williams and D. A. Fisher. X, 496 pages. 1977.

Vol. 55: A. Gerbier, Mes premières constructions de programmes. XII, 256 pages. 1977.

Vol. 56: Fundamentals of Computation Theory. Proceedings 1977. Edited by M. Karpiński. XII, 542 pages. 1977.

Vol. 57: Portability of Numerical Software. Proceedings 1976. Edited by W. Cowell. VIII, 539 pages. 1977.

Vol. 58: M. J. O'Donnell, Computing in Systems Described by Equations. XIV, 111 pages. 1977.

Vol. 59: E. Hill, Jr., A Comparative Study of Very Large Data Bases. X, 140 pages. 1978.

Vol. 60: Operating Systems, An Advanced Course. Edited by R. Bayer, R. M. Graham, and G. Seegmüller. X, 593 pages. 1978.

Vol. 61: The Vienna Development Method: The Meta-Language. Edited by D. Bjørner and C. B. Jones. XVIII, 382 pages. 1978.

Vol. 62: Automata, Languages and Programming. Proceedings 1978. Edited by G. Ausiello and C. Böhm. VIII, 508 pages. 1978.

Vol. 63: Natural Language Communication with Computers. Edited by Leonard Bolc. VI, 292 pages. 1978.

Vol. 64: Mathematical Foundations of Computer Science. Proceedings 1978. Edited by J. Winkowski. X, 551 pages. 1978.

Vol. 65: Information Systems Methodology, Proceedings, 1978. Edited by G. Bracchi and P. C. Lockemann. XII, 696 pages. 1978.

Vol. 66: N. D. Jones and S. S. Muchnick, TEMPO: A Unified Treatment of Binding Time and Parameter Passing Concepts in Programming Languages. IX, 118 pages. 1978.

Vol. 67: Theoretical Computer Science, 4th GI Conference, Aachen, March 1979. Edited by K. Weihrauch. VII, 324 pages. 1979.

Vol. 68: D. Harel, First-Order Dynamic Logic. X, 133 pages. 1979.

Vol. 69: Program Construction. International Summer School. Edited by F. L. Bauer and M. Broy. VII, 651 pages. 1979.

Vol. 70: Semantics of Concurrent Computation. Proceedings 1979. Edited by G. Kahn. VI, 368 pages. 1979.

Vol. 71: Automata, Languages and Programming. Proceedings 1979. Edited by H. A. Maurer. IX, 684 pages. 1979.

CONTENTS

# SHARING IN NONDETERMINISM

Egidio ASTESIANO - Gerardo COSTA

Istituto di Matematica dell'Università di Genova
Via L.B.Alberti, 4 - 16132 Genova - Italy

*ABSTRACT. We consider a language of typed λ-expressions with primitives including nondeterministic choice operators. Starting from the natural idea that a first order nondeterministic procedure should define a one-many function, we give a reduction system in which ground arguments are shared, in order to avoid some unnatural consequences due to unrestricted application of the copy-rule. This is achieved by extending the language and modifying the usual β-rule. Then we discuss how to define a corresponding denotational semantics, establishing in particular the existence of a model which is fully abstract w.r.t. the operational semantics.*

## 1. INTRODUCTION

Consider the usual language of first order deterministic recursive procedures, enriched with a binary choice operator or and extend the usual evaluation mechanism by letting t or t' evaluate to either t or t' . Then consider the following example /HA1/ where x ranges over the domain of non-negative integers $\mathbb{N}$ :

$$\begin{cases} F(x) \Leftarrow \text{ if } x=0 \text{ then } 0 \text{ else } G(x \text{ or } x-1) \\ G(x) \Leftarrow \text{ if } x=0 \text{ then } 0 \text{ else if } x=1 \text{ then } 1 \text{ else } 2 \end{cases} .$$

Evaluation of F(1), using an outermost strategy, yields, as possible values, 0, 1 and 2. Hennessy and Ashcroft point out that in a functional model corresponding to this evaluation strategy F and G cannot denote functions f and g from $\mathbb{N}$ into $2^{\mathbb{N}}$ . Indeed we would have: $f(1) = g(\{0,1\}) = g(0) \cup g(1) = \{0,1\}$ , in contrast with the operational result. This motivates in /HA1,HA2/ the choice of a model in which the meanings for F and G are functions from $2^{\mathbb{N}} \setminus \{\emptyset\}$ into itself. From a mathematical point of view, however, it seems perfectly reasonable to take a different approach: the equation for G is deterministic and by itself defines a function from $\mathbb{N}$ into $\mathbb{N}$ , it should then define the "same" (see below) function when considered together with the equation for F. In general, it is mathematically sound to say that nondeterministic recursive procedures define one-many functions. In this framework, deterministic procedures, like the one for G, define one-singleton functions and the result of applying a function f to a set A is given by $\cup \{f(a), a \in A\}$ . This is the approach we shall take here (the same choice is made in /AN1,AN2/, though in a different setting).

Looking for an operational semantics corresponding to this (intuitive) model, we remark that the point in the above example is that G makes two copies of its argument,

which then behave independently; hence we have to inhibit this possibility. There are essentially three solutions.

a) Innermost derivations (innermost w.r.t. choices as well as w.r.t. unknown function symbols.

b) Derivations in which choices have the precedence over replacement of unknown function symbols (this implies the problem of detecting "implicit choices", see below).

c) Outermost derivations together with sharing techniques.

In the example above, the three strategies yield the same result (e.g. F(1) evaluates to 0 or 1) but this is not true in general. For instance, consider the following system, where x and y range over $\mathbb{N}$ :

$$\begin{cases} F(x,y) \Leftarrow G(x,K(y)) \\ G(x,y) \Leftarrow \text{if } x=0 \text{ then } 0 \text{ else } G(x-1,H(y)) \\ H(x) \Leftarrow H(x) \text{ or } H(x-1) \\ K(x) \Leftarrow 1 + K(x) \end{cases}$$

According to a) : F(m,n) obviously diverges, any m and n.

According to b) : F(0,n) evaluates to 0, any n, but F(m+1,n) evaluates to G(m,H(K(n))), which diverges, any m and n, because of H(K(n)); notice that we have to derive H(K(n)) first, because H implies a choice.

According to c) : F(m,n) evaluates to 0, any m and n.

Actually, there is a strict hierarchy: strategy a) is less powerful then strategy b) which is less powerful then strategy c); where "less powerful" means that the computed functions are less defined. Moreover it seems that there are some difficulties in defining precisely an operational semantics based on strategy b) and the corresponding functional model. This has been attempted in /HA2/ for this language of first order recursive procedures over flat domains (the evaluation mechanism is called "call-time choice"), but the results are rather unsatisfactory.

Strategy c) seems the one which better corresponds to our purpose of preventing duplication of arguments, without introducing unwanted non-terminations. Therefore it is the one we analyze here, using a nondeterministic language of typed $\lambda$-expressions, NDLS, derived from PCF (see /P2/ and also /HA1,AC2/).

It turns out that to be consistent with the mathematical model we have in mind, we have to consider sharing only w.r.t. arguments of ground type. Indeed, consider an higher type version of the relevant part of our first example.

$\theta(X)(n) \Leftarrow$ if $X(n)=0$ then 0 else if $X(n)=1$ then 1 else 2 ;

where n and X range over $\mathbb{N}$ and $\mathbb{N} \to 2^{\mathbb{N}}$, respectively, and so we want $\theta$ to denote a function $\theta$ in $(\mathbb{N} \to 2^{\mathbb{N}}) \to \mathbb{N} \to 2^{\mathbb{N}}$. Now, in a call like $\theta(F \text{ or } G)(1)$, where F and G are the procedures $F(n) \Leftarrow n$, $G(n) \Leftarrow n-1$, we regard F or G as a procedure itself; hence it must denote a function, say h, from $\mathbb{N}$ into $2^{\mathbb{N}}$ and, quite naturally, h is defined by $h(n) = \{n,n+1\}$. So we get:

$\theta(h)(1)$ = if $h(1)=0$ then 0 else if $h(1)=1$ then 1 else 2 = $\{0,1,2\}$ (by definition of natural extension to sets of one-one functions such as = and if-then-else).

It is clear that this result is obtained, operationally, with the usual evaluation mechanism (call-by-name with copy-rule), assuming that F or G may evaluate to either F or G.

Using sharing at higher level (apart from the technical problems, see /W/) would correspond to an intuitive setting quite different from the one we have outlined here. For example, it seems that the meaning of F or G should be a set of functions; this would pose several problems, like finding a suitable powerdomain structure. We think, however, that this point deserves further investigation.

The discussion above explains also why in our language NDLS we do not have or symbols of higher type (they are not needed as we can define them in terms of ground type or's) and why in our models we must consider domains of sets at ground level only.

Having defined the intuitive setting of our work, we are faced with two problems: the first one is that of finding a suitable description for the sharing of ground arguments (suitable w.r.t. both the operational and the denotational semantics); the second concerns the mathematical models.

In the language of the first two examples, sharing can be described very naturally by using graphs or labeled terms /V, PMT/; but also in a neater way (neater mathematically speaking) using derived algebras (see for instance /AC1/) or magmoids /A/. In $\lambda$-calculus the description is much more involved. The classical approach is that of Wadsworth /W/ using graphs. Another approach /L/ makes use of a precise notion of duplication of redexes and of simultaneous contraction of duplicated redexes; but an evaluation mechanism corresponding to this strategy has not yet been given.

The use of graphs has two drawbacks in our case. First of all, we want our operational semantics to be given by a formal reduction system and this seems rather difficult to obtain on graphs. Secondly, when both an operational and a denotational semantics are given it is useful (mainly in proving invariance of the semantics through evaluation) to associate a denotational meaning not only to programs and results, but also to each entity which represents an intermediate step in the evaluation process. Now if these intermediate steps are represented by graphs, one is faced with the problem of transferring the sharing relations into the denotational semantics.

The solution we propose is that of extending the language of $\lambda$-expressions by introducing terms of the form $M\{N/x\}$ and modifying the $\beta$-rule by having $(\lambda x M)N$ reduce to $M\{N/x\}$, when x is of ground type. In other words we suspend the substitution of N in M and keep x bound to its actual value; the idea is similar to that of the association list of the LISP interpreter (but suitable renamings of bound variables prevent from the fluid variables phenomenon).

Modifying the $\beta$-rule requires that we give an explicit reduction algorithm. We do this by a reduction system which is monogenic (but for the choice rule) and which follows a kind of call-by-need strategy /W,V/. In this system the concept of critical variable is central: roughly speaking, a variable is critical in a term if we need to know the value of the actual argument associated to it to proceed in our evaluation

(notice the obvious connection with the concept of sequentiality /V,B/; at this point we rely on the kind of interpretations we consider).

As for the functional semantics of the terms M {N/x} , it is given following the usual style of the denotational semantics (this does not came to light here, since proofs have been omitted; see however section 7).

The second problem concerns the models for our language. It is rather easy to give a model, $\mathcal{S}$ , for NDLS by modifying the usual Scott-Milner continuous functions model. However what one finds is a model which is "too large" and therefore not intrinsicly fully abstract (i.f.a. for short) in the sense of /M2/. We show in /AC3/ that the technique used by Milner in /M2/ can be adapted to obtain an i.f.a. model for typed $\lambda$-calculi with sharing at ground level. Here we simply state the main result and apply it to show the (constructive) existence of an i.f.a. model, $\overline{\mathcal{M}}$ , for NDLS. Finally, following a pattern already outlined in /AC2/, we discuss the relationships between the operational semantics defined by the reduction system and the denotational semantics associated to the models $\mathcal{S}$ and $\overline{\mathcal{M}}$ . In particular, the semantics defined by $\overline{\mathcal{M}}$ is shown to be equivalent to the operational one (i.e. $\overline{\mathcal{M}}$ is fully abstract w.r.t. the operational semantics, according to the well known definition in /M1/).

## 2. THE LANGUAGE NDLS

We consider two ground types, $o$ and $\iota$ ; then T, $\sigma$ and $\tau$ , $\kappa$ will denote, respectively: the set of functional types generated from $o$ and $\iota$, arbitrary types, a general ground type. We shall call first order type any type of the form $\kappa_1 \to \kappa_2 \to \dots \kappa_n \to \kappa$ for $n \geq 1$.

The set of terms of the language NDLS is the set generated, by using typed $\lambda$-abstraction and application, from :

1) a set of ground constant symbols (typical element c) namely: tt, ff, of type $o$ : 0, 1, ..., n,.... of type $\iota$ ;
2) a set of first order constant symbols, namely: Z , of type $\iota \to o$ ; (+1) and (-1) , of type $\iota \to \iota$ ; $IF_\kappa$ , of type $o \to \kappa \to \kappa \to \kappa$ ;
3) the special combinators $or_\kappa$ , of type $\kappa \to \kappa \to \kappa$ ;
4) the sets $VAR = \{ X_i^\sigma , \sigma \in T , i \geq 1 \}$ and $FIX = \{ Y_\sigma , \sigma \in T \}$ , where $X_i^\sigma$ and $Y_\sigma$ have type $\sigma$ and $(\sigma \to \sigma) \to \sigma$ respectively.

We adopt the usual conventions for suppressing redundant parenthesis in writing terms and we shall often omit subscripts in primitive symbols and variables. We use letters M and N to denote arbitrary terms, adding sometimes subscripts to indicate their type. Closed terms of ground type will be called programs.

We shall need later on $\Omega_\sigma$ and $Y_\sigma^{(n)}$ , defined as follows: $\Omega_\kappa = Y_\kappa(\lambda X_1^\kappa X_1^\kappa)$, $\Omega_{\sigma\to\tau} = \lambda X_1^\sigma \Omega_\tau$ , $Y_\sigma^{(0)} = \Omega_{(\sigma\to\sigma)\to\sigma}$ and $Y_\sigma^{(n+1)} = \lambda X(X(Y_\sigma^{(n)} X))$ , where X stands for $X_1^{\sigma\to\sigma}$.

Finally, $M^{(n)}$, $n \geq 0$, will denote the term obtained from M by replacing all occurrences of $Y_\sigma$ by $Y^{(n)}_\sigma$, for all $\sigma$, and we shall call finite any term which contains Y only in the combination for $\Omega_\kappa$ (e.g. $M^{(n)}$ is finite).

## 3. OPERATIONAL SEMANTICS

We now define the language ENDLS, extension of NDLS, in which we are able to express sharing of ground arguments. We shall define our reduction system w.r.t. this extended language, but we are only interested in giving an operational semantics for programs in NDLS.

Let ENDLS be the language defined by all the rules for NDLS plus the following one: if $M_\sigma$ and $N_\kappa$ are terms, then $(M_\sigma \{N_\kappa / X^\kappa_i\})$ is a term of type $\sigma$, any $\sigma$, $\kappa$, i. For example $(((((\ IF_\iota\ X^o_1)\ X^\iota_1\ )\ \{((+1)\underline{2})\ /\ X^\iota_1\}\ )\ X^\iota_2\ )\ \{\underline{tt}\ /\ X^o_1\}\ )$ is a term of type $\iota$ in ENDLS. We shall extend to terms in ENDLS the conventions on round brackets and subscripts, so the term above would usually be written as $IF_\iota\ X\ X_1\ \{\ (+1)\underline{2}\ /\ X_1\}\ X_2\ \{\underline{tt}\ /\ X\ \}$.

For terms in ENDLS the definition of free variable is obtained by adding to the usual definition the following clause: $Free(M\{N/X\}) = (Free(M) - X\ ) \cup Free(N)$, where Free(M) is the set of variables having a free occurrence in M. Hence, if M[N/X] denotes the result of substituting N for all free occurrences of X in M (with suitable renamings) then, for example, $(X^o\{X^o/X^o\}\ )\ [\underline{tt}/X^o] = X^o\{\underline{tt}/X^o\}$. The definition of open (closed) term is the usual one (but w.r.t. the new definition of free variable).

We now define a reduction algorithm for terms in ENDLS by means of a (partial) relation $\longrightarrow$ between terms, which is given by the rules below.

1) $\underline{or}\ M\ N \longrightarrow M$ ; $\underline{or}\ M\ N \longrightarrow N$ ;

2) $(+1)\underline{n} \longrightarrow \underline{n+1}$ ; $(-1)\underline{n+1} \longrightarrow \underline{n}$ ; $(-1)\underline{0} \longrightarrow \underline{0}$ ;

3) $Z\ \underline{0} \longrightarrow \underline{tt}$ ; $Z\ \underline{n+1} \longrightarrow \underline{ff}$ ;

4) $IF\ \underline{tt}\ M\ N \longrightarrow M$ ; $IF\ \underline{ff}\ M\ N \longrightarrow N$ ;

5) $Y\ M \longrightarrow M(YM)$ ;

6a) $(\lambda X^\sigma\ M)N \longrightarrow M\ [N\ /\ X^\sigma]$ , for non ground $\sigma$

6b) $(\lambda X^\kappa\ M)N \longrightarrow M\ \{N\ /\ X^\kappa\}$ , where we perform in M all the renamings of bound variables which would be required by the application of the usual $\beta$-rule;

7a) $\dfrac{M \longrightarrow M'}{MN \longrightarrow M'N}$ ;

7b) $\dfrac{N \longrightarrow N',\ f \in \{(+1), (-1), Z, IF\}}{fN \longrightarrow fN'}$ ;

8) $\dfrac{M \longrightarrow M'}{M\{N/X\} \longrightarrow M'\{N/X\}}$ ;

9) $\gamma\ \{N/X\} \longrightarrow \gamma$ , where $\gamma$ is a primitive symbol (i.e. $\underline{c}$, $(+1)$, Y, $\underline{or}$,...);

10) $(\underline{or}\ M)\ \{N/X\} \longrightarrow \underline{or}\ (M\ \{N/X\}\ )$ ;

11a) $(IF\ \underline{b})\ \{N/X\} \longrightarrow IF\ \underline{b}$

11b) $(IF\ \underline{b}\ M)\ \{N/X\} \longrightarrow (IF\ \underline{b})(M\ \{N/X\})$ , for $\underline{b} = \underline{tt}, \underline{ff}$ ;

12) $(\lambda X_i^\sigma M)\ \{N/X_j^\kappa\} \longrightarrow \lambda \tilde{X}_i^\sigma\ (\tilde{M}\ \{N/X_j^\kappa\})$ , where $\tilde{}$ indicates renamings, to be performed if $\sigma = \kappa$ and $i = j$, or if $X_i^\sigma$ is free in N;

13a) $\dfrac{X \in CR(M)}{M\ \{c/X\} \longrightarrow M\ [c/X]}$ ;  13b) $\dfrac{X \in CR(M)\ ,\ N \longrightarrow N'}{M\ \{N/X\} \longrightarrow M\{N'/X\}}$ ;

where CR(M) is defined as follows (in a PASCAL-like language):

```
CR(M) := case M of
           X_i^K          : {X_i^K} ;
           M1 M2          : case M1 of
                              (+1),(-1),Z,IF : CR(M2) ;
                              "else"         : CR(M1) ;
                            end
           M1{M2/X}       : if X ∈ CR(M1) then CR(M2) else CR(M1);
           "else"         : ∅
         end
```

Notice that CR(M) is a subset of Free(M) and it is either a singleton or the empty set; we call it the critical set of M and if $CR(M) = \{X_i^K\}$ , then we say that $X_i^K$ is critical in M.

LEMMA 2.1 For any term M in ENDLS, if M' in ENDLS exists s.t. $M \longrightarrow M'$, then $CR(M)=\emptyset$. We omit the proof, which can be easily done by induction on the structure of terms. Remark that the reverse of the implication is false (just consider $M = X^{1\to 1}$ ). □

The three examples below should illustrate the non standard features of our rewriting system; the last one embeds the translation of the first example in section 1 and emphasizes the differences between rules 6a) and 6b) in relation to or ( we thank one of the referees for suggesting it). Here we symplify our notation by using x,y,w as names for variables and, in the third example, by using infix form for (+)1, or and IF. Side comments should help understanding why Theorem 2.2 below is true, i.e. no freedom is permitted except when using rule 1.

EX.1 $(( \lambda x\ ( \lambda x\ x\ ))\ M\ )\ (\ or\ 1\ 2\ ) \longrightarrow$ (both x's are ground here)

$(( \lambda y\ y\ )\ \{M/x\}\ )\ (\ or\ 1\ 2\ ) \longrightarrow$ (by rule 12, if y is not free in M)

$(\ \lambda y\ (\ y\ \{M/x\}\ )\ (\ or\ 1\ 2\ ) \longrightarrow$

$(\ y\ \{M/x\}\ )\ \{or\ 1\ 2\ /y\} \longrightarrow$ ( $CR(y\{M/x\}) = CR(y) = \{y\}$ )

$y\{M/x\}\{n/y\} \longrightarrow n\{M/x\} \longrightarrow n$ ( n is either 1 or 2 ).

EX.2 $(\ \lambda x\ (((\ \lambda x\ (IF_0\ x))x)\ tt\ x)\ )\ (or\ tt\ ff) \longrightarrow$

$(((\ \lambda x\ (IF_0\ x))\ x)\ tt\ x)\ \{or\ tt\ ff\ /x\} \longrightarrow$

$((IF_0\ x)\{x/x\}\ tt\ x)\ \{or\ tt\ ff\ /x\} \longrightarrow$ ($CR(IF_0\ x\{x/x\}tt\ x)=CR(IF_0\ x\{x/x\})=\{x\}$ ; note it is the x above / )

$((IF_0\ x)\ \{x/x\}\ tt\ x)\ \{b/x\} \longrightarrow$ (b is either tt or ff)

$((IF_0\ x)\ \{b/x\}\ tt\ b) \longrightarrow$

$IF_0\ b\ tt\ b \longrightarrow b$ .