ON REWRITING TECHNIQUES

THESIS

SUBMITTED IN PARTIAL FULFILMENT
OF THE REQUIREMENTS
FOR THE AWARD
OF
THE (M. Sc) DEGREE
(COMPUTER SCIENCE)

By

MOHAMED ESMAIL ALI HAMADA

DEPARTMENT OF MATHEMATICS
FACULTY OF SCIENCE
AIN SHAMS UNIVERSITY

SUPERVISORS

DR. MAHMOUD K. A. KHAIRAT
DEPARTMENT OF MATHEMATICS
FACULTY OF SCIENCE
AIN SHAMS UNIVERSITY

DR. FAYED F. M. GHALEB
DEPARTMENT OF MATHEMATIC
FACULTY OF SCIENCE
AIN SHAMS UNIVERSITY

SUBMITTED TO

50944

004r
M.E

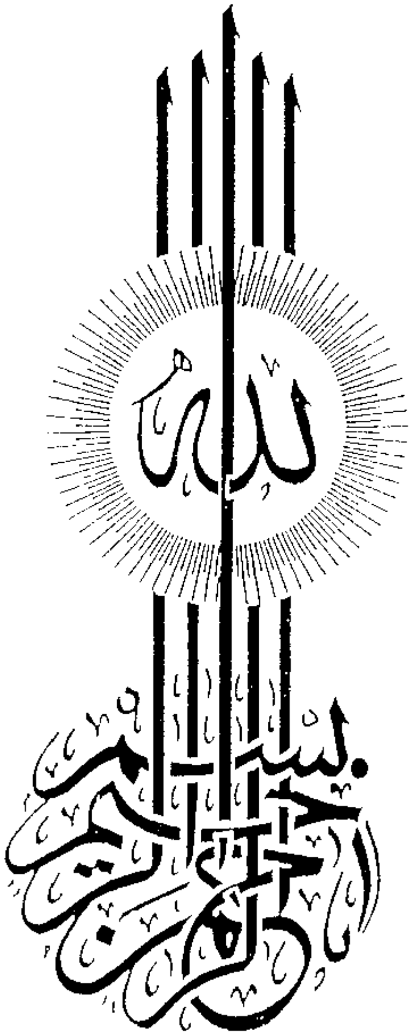
FACULTY OF SCIENCE
AIN SHAMS UNIVERSITY
CAIRO

1992

ACKNOWLEDGMENT

The author is deeply grateful to ***Dr. Mahmoud K. A. Khairat***, Mathematics Department, Faculty of Science, Ain Shams University, for his kind supervision, suggesting this line of research, and valuable advices.

The author is greatly indebted to ***Dr. Fayed F. M. Ghaleb***, Mathematics Department, Faculty of Science, Ain Shams University, for his kind supervision and systematic guidance. With his generous and fruitful discussions this work has been completed.


بسم الله الرحمن الرحيم

CONTENTS

Page

SUMMARY

The thesis is divided into five chapters, appendix, and glossary.

Chapter one contains a summary of the important types of rewriting systems and its recent applications in many areas. Also a discussion of the concept of "Modularity" of rewriting is given.

In chapter two we give the basic definitions and properties of term rewriting systems "TRS". In this chapter we introduce a general type of induction called "Global induction" and from it we deduce the principle of noethrian induction and the principle of structural induction.

In Chapter three we study in some details the termination property of term rewriting systems which is one of the most important properties of term rewriting systems.

In Chapter four we mainly study the critical pair completion technique "CPC" which has important applications in universal algebra. The CPC technique depends mainly on the Church-Rosser property, the termination property of TRSs, and on the unification of first order terms. Due to this fact an essential part of this chapter is devoted to study the Church-Rosser property of TRSs and the unification of first order terms.

In chapter five we introduce our attempt to implement the unification algorithm, the normal form algorithm, and the critical pair algorithm, using the ADA language under DOS.

We enclose a disk containing the executable file TRS.EXE of the implemented algorithms.

Most of the definitions of rewriting and the different usage by different authors is given in a glossary at the end of the thesis.

Appendix A contains a complete proof of a lemma concerned with a relation between the numbers of occurrence of operator symbols of different degrees in a first order term. This lemma is used in Chapter 3 (Theorem 3.1) to prove that the KB ordering on pure terms is a well-ordering.

CHAPTER I

PREFACE

∮1.1 INTRODUCTION

Rewriting systems have many types according to the structure of the used terms. In Section 1.2 we desscuse some important types of rewriting systems and give illustrative examples. We will mainly study term rewriting systems and its properties for first order terms. In Section 1.3 we give a short summary of some recent works using rewriting techniques. In Section 1.4 we define "Modularity", one of the important properties in computer programming and illustrate its relation to term rewriting systems.

∮1.2 SOME TYPES OF REWRITING

In this section we survey some types of rewriting techniques. The most important types of rewriting systems are defined in the glossary.

1. *Abstract reduction system* ARS

An ARS is a structure $(A, (\longrightarrow_i)_{i\in I})$, consisting of a set A and a sequence of binary relations $\longrightarrow_i$ on A.

In abstract reduction systems each element of A^* is called an abstract term. If the structure of the term take the form of first order terms (i.e. terms that are built up from a set of operation symbols and a countable set of variables see Defination 2.1) then the abstract reduction system is called term rewriting system TRS, if the structure of the term takes the form of a tree then the abstract reduction system is called tree replacement system TRS ([33]), and if the term takes the form of a graph then the abstract reduction system is called graph rewriting system GRS. For a survey of abstract reduction system see [26].

2. *Term Rewriting Systems* TRS ([23], [11], [26])

Let T be a set of first order terms defined over a finite set F of operation symbols and a (countable) set V of variables. The set $R \subset T \times T$, $R = \bigcup_{i \geq 1} \{(s_i, t_i)\}$ such that $\forall_{i \geq 1}$ $v(t_i) \subseteq v(s_i)$ and s_i is nontrivial term is called a term rewriting system *TRS*. (where v(t) denotes the set of variables that occurs in the term t).

3. *String rewriting systems* SRS (called Thue-system [7])

Given an (finite) alphabet Σ, an SRS R is a binary relation over Σ^* (Σ^* is the set of all strings over Σ), R subset of $\Sigma^* x \Sigma^*$. The rewriting of a given string $w \in \Sigma^*$ is performed by replacing some occurrence of string v in w by string u [7].

Example 1

Let $\Sigma = \{r, w, b\}$ (r=red, w=white, b=black)

and $R = \{$ 1. wr $\longrightarrow$ rw,
2. bw $\longrightarrow$ wb,
3. br $\longrightarrow$ rb,
4. rr $\longrightarrow$ r,
5. bb $\longrightarrow$ b,
6. ww $\longrightarrow$ w $\}$.

Applying this SRS R on (for example) the string bwrrrbwwbrbbrw we shall get rwb :

$$\mathrm{bw\underline{rr}rbwwbrbbrw} \xrightarrow{4} \mathrm{bw\underline{rr}bwwbrbbrw} \xrightarrow{4} \mathrm{bwrb\underline{ww}brbbrw} \xrightarrow{6}$$

$$\mathrm{bwrbwbr\underline{bb}rw} \xrightarrow{5} \mathrm{\underline{bw}rbwbrbrw} \xrightarrow{2} \mathrm{w\underline{br}bwbrbrw} \xrightarrow{3}$$

$$\mathrm{w\underline{bb}rwbrbrw} \xrightarrow{5} \mathrm{w\underline{br}wbrbrw} \xrightarrow{3} \mathrm{\underline{wr}bwbrbrw} \xrightarrow{1} \mathrm{rw\underline{bw}brbrw} \xrightarrow{2}$$

$$\mathrm{r\underline{wwbb}rbrw} \xrightarrow{5,6} \mathrm{rw\underline{br}brw} \xrightarrow{3} \mathrm{rwr\underline{bb}rw} \xrightarrow{5} \mathrm{rwr\underline{br}w} \xrightarrow{3}$$

$$\mathrm{rw\underline{rr}bw} \xrightarrow{4} \mathrm{r\underline{wr}bw} \xrightarrow{1} \mathrm{\underline{rr}wbw} \xrightarrow{4} \mathrm{rw\underline{bw}} \xrightarrow{2} \mathrm{r\underline{ww}b} \xrightarrow{6} \mathrm{rwb}.$$

(red white black).

Example 2

The above example can be extended to define a conditional rewrite system which always gives rwb even if the length of the given string was one or two.

$\Sigma = \{r, w, b\}$ (r=red, w=white, b=black),

R = { 1. wr $\longrightarrow$ rw,

2. bw $\longrightarrow$ wb,

3. br $\longrightarrow$ rb,

4. rr $\longrightarrow$ r,

5. bb $\longrightarrow$ b,

6. ww $\longrightarrow$ w,

7. If length of the string = 1 then r|w|b $\longrightarrow$ rwb,

8. If length of the string = 2 then rw $\longrightarrow$ rwb,

wb|rb $\longrightarrow$ rw}.

In this system the rule 7 will be applied only on strings of length one and the rule 8 will be applied only on strings of length two. This extension permits strings of length$\geq$1 to be rewritten to the rwb (red, white, black).

4. Graph Rewriting systems GRS

A graph rewriting systems is represented by an initial graph and a set of rewrite rules. Each rewrite rule consists of a left-hand--side graph and a right-hand-side graph [16].

<u>Definition</u> Term graph [19]

A term graph is a natural generalization of a labeled tree. Informally, term graphs are rooted, directed, and ordered graphs in which each node is labeled by a function symbol or a variable. Unlike trees, term graphs allow the nodes to be shared and, in particular, may contain cycles. Formally, a term graph (over a set F of function symbols and a countable set V of variables) is a quadruple (N, lab, suc, p) where N is a set of nodes with p in N,

$$lab : N \longrightarrow F \cup V, \text{ and } suc : N \Longrightarrow N^*,$$

such that

$|suc(n)| = arity(lab(n))$ (where $arity(v)=0 \ \forall \ v\in V$) for all n in N. For n in N lab(n) is called the label of n and the **members** of suc(n) are called the successors of n. p is called the root of the term graph.

A graph is <u>*ROOT CONNECTED*</u> if there is a path from the root to each node in the graph. If there is exactly one path from the root of the graph G to each other node of G, then the graph G is a tree.

The triple G = (N, lab, suc) where lab and suc are partial functions on N, is called <u>*OPEN LABELED GRAPH*</u>.

__Definition__ "Graph rewrite rule" [19]

A graph rewrite rule is a pair r = (L, R) where L, R are root connected graphs. In [5] a graph rewrite rule is a triple (G,n,n') where G is an open labeled graph and n, n' are nodes of G, called respectively the left root and the right root of the rule.

__Definition__ "Homomorphisms of graphs" [5]

Given two graphs $G_1=(N_1,lab_1,suc_1,n_1)$ and $G_2=(N_2,lab_2,suc_2,n_2)$ a homomorphism from G_1 to G_2 is a map $f : N_1 \longrightarrow N_2$ such that for all n in N_1,

$$lab_2(f(n)) = lab_1(n)$$

$$suc_2(f(n)) = \bar{f}(suc_1(n))$$

where $\bar{f}$ is defined by $\bar{f}(m_1, \ldots, m_k) = (f(m_1), \ldots, f(m_k))$, that is, homomorphism preserves labels, successors, and their order.

__Definition__ "Redex in term graphs" ([5])

A redex in a term graph G_o is a pair D = (R, f), where R is a graph rewrite rule (G, n, n') and f is a homomorphism from G/n to G_o. (G/n represents the subgraph of G rooted at n). The homomorphism f is called an occurrence of R.

__Definition__ "Graph rewriting" ([19])

Let R be a set of rewrite rules, a graph rewriting of a graph G_0 via R is either

1) a finite sequence $L = \langle(G_o, D_o), \ldots, (G_n, D_n), G_{n+1}\rangle$ such that

$$G_o \xrightarrow{Do} G_1 \xrightarrow{D1} G_2 \xrightarrow{D2} \ldots \xrightarrow{Dn} G_{n+1},$$

or

2) an infinite sequence $L = \langle (G_o, D_o), (G_1, D_1), (G_2, D_2), \ldots \rangle$ such that

$$G_0 \xrightarrow{Do} G_1 \xrightarrow{D1} G_2 \xrightarrow{D2} \ldots ,$$

where G_i is the graph derived from the graph G_{i-1} using the rewriting rule D_{i-1} , $i = 1, 2, \ldots$.

§1.3 APPLICATIONS OF REWRITING

In this section we shall survey some recent applications of rewriting techniques.

1. In 1977 O'Donnel and in 1978 Bakous introduced programming languages in which rewrite rules are used to provide an operational semantics ([33], [2]).

When rewrite rules are used in this way the authors were faced with two obvious questions :

i. Are there too many rules, thus making equivalent two expressions that should be different? (soundness),

ii. Are there enough rules to be able to compute the correct meaning of all expressions? (completeness).

Recently in 1988 J.Y. Halpern, J.H. Williams, E.L. Wimmers ([21]) developed a formal framework which gives precise answers to these soundness and completeness questions. They investigated these issues in the context of an extended version of the functional programming language FP ([2]).

2. In 1988 H. Barendregt, M. Eekelen, J. Glauert, J. Kennaway, M. Plasmeijer and M. Sleep [6] present "Lean", an experimental language based on a graph rewriting model. In their approach they have extended term rewriting systems [34] to a model of general graph rewriting. Such a model make it possible to reason about programs, to prove correctness, and to port programs to different machines. A Lean computation is specified by an initial graph and a set of rules used to rewrite the graph to its final result. The rules contain graph patterns that may match some part of the graph, in this case it can be rewritten according to these rules.

3. In 1988, T. Brus, M. Eekelen, M. Leer, and M. Plasmeijer [8] introduced the language "clean", which is roughly a subset of "lean" intended for functional languages only. In clean, graph representations of terms are used to perform term rewriting more efficiently. The basis of clean is that a computation is represented by an initial graph and a set of rules used to rewrite this graph to its result. Clean describes functional graph rewriting in which only the root of the subgraph matching a pattern may be overwritten. The semantics allow parallel rewriting where candidate rewrites do not interfere. The rewriting process stops if none of the patterns in the rules match any part of the graph which means that the graph is in normal form. See also [17].

4. In 1989, R. Gallimore, D. Coleman, and V. Stavridou [20] defined an algebraic specification language implemented by the UMIST OBJ system. The system implements an executable subset of J.