ON WORD EQUATIONS AND MAKANIN'S ALGORITHM

Habib Abdulrab, Jean-Pierre Pécuchet
Laboratoire d'Informatique de Rouen et LITP.
Faculté des Sciences, B.P. 118, 76134 Mont-Saint-Aignan Cedex †
E.m.: mcvax!inria!geocub!abdulrab mcvax!inria!geocub!pecuchet

ABSTRACT. — We give a short survey of major results and algorithms in the field of solving word equations, and describe the central algorithm of Makanin.

Introduction

An algebra equipped with a single associative law is a **semigroup**. It is a **monoid** when it has a unit. The free monoid generated by the set A (also called **alphabet**) is denoted by A^*. Its elements are the **words** written on the alphabet A, the neutral element being the empty word denoted by 1. The operation is the concatenation denoted by juxtaposition of words. The **length** of a word w (the number of letters composing it) is denoted by $|w|$. For a word $w = w_1 \dots w_n$, with $|w| = n$, we denote by $w[i] = w_i$ the letter at the ith position. The number of occurrences of a given letter $a \in A$ in a word w, will be denoted by $|w|_a$.

In this terminology, the term algebra (in the sense of [Fag Hue], [Kir]) built on a set of variables V, a set C of constants, and a set of operators constituted of an associative law, is nothing else than the free monoid $T = (V \bigcup C)^*$ over the alphabet of **letters** $L = V \bigcup C$.

A unifier of two terms $e_1, e_2 \in T$ is a monoid morphism $\alpha : T \longrightarrow T$ (i.e. a mapping satisfying $\alpha(mm') = \alpha(m)\alpha(m')$ and $\alpha(1) = 1$), leaving the constants invariant (i.e. satisfying $\alpha(c) = c$ for every $c \in C$) and satisfying the equality $\alpha(e_1) = \alpha(e_2)$.

The pair of words $e = (e_1, e_2)$ is called an **equation** and the unifier α is a **solution** of this equation.

A solution $\alpha : T \longrightarrow T'$ **divides** a solution $\beta : T \longrightarrow T''$ if there exists a **continuous** morphism $\theta : T' \longrightarrow T''$ (i.e. satisfying $\theta(x) \neq 1$ for every x) such as $\beta = \alpha\theta$. We also say that α is **more general** than β. A solution α is said to be **principal** (or **minimal**) when it is divided by no other but itself (or by an **equivalent** solution, i.e. of the form $\alpha' = \alpha\theta$ with θ, an isomorphism).

The two main problems concerning systems of equations are the existence of a solution, and the computation of the set of minimal solutions (denoted by μCSU_A in [Fag Hue]). All these problems reduce to the case of a single equation, as by [Alb Law] every infinite system of equations is equivalent to one of its finite subsystems, and a finite system can be easily encoded in a single equation [Hme].

---

† This work was also supported by the Greco de Programmation du CNRS and the PRC Programmation Avancée et Outils pour l'Intelligence Artificielle.

The study of properties and structure of the set of solutions of a word equation was initiated by Lentin and Schützenberger ([Len Sch], [Len]) in the case of constant-free equations ($C = \emptyset$).

In particular, Lentin shows that every solution is divided by a unique minimal one and gives a procedure (known as the **pig-pug**) allowing to enumerate the set of minimal solutions. This procedure extends without difficulty to the general case of an equation with constants (cf. [Plo], [Pec1]). The minimal solutions are obtained as labels of some paths of a graph. When this graph is finite, as in the case when no variable appears more than twice, we obtain a complete description of <u>all</u> solutions.

The problem of the existence of a solution was first tackled by Hmelevskii who solved it in the case of three variables [Hme], then by Makanin who solved the general case [Mak1]. He gave an algorithm to decide whether a word equation with constants has a solution or not.

This paper is divided into two parts. The first one will be devoted to a brief presentation of the pig-pug method which gives, for simple cases, the most efficient unification algorithm. The rest of this paper will be devoted to Makanin's Algorithm [Mak1] as it is implemented by Abdulrab [Abd1]. In order to keep a reasonable size to this paper, most of the proofs will be omitted.

1. The pig-pug

In the remaining part of this paper, we assume without loss of generality, that the alphabets of variables $V = \{v_1 \dots v_n\}$ and of constants $C = \{c_1 \dots c_m\}$ are finite and disjoint. We make the convention to represent the variables by lower-case letters, as $x, y, z \dots$, and the constants by upper-case letters as $A, B, C \dots$. We call **length** of an equation $e = (e_1, e_2)$ the integer $d = |e_1 e_2|$.

The **projection** of an equation e over a subset Q of V is the equation obtained by "erasing" all the occurrences of $V \setminus Q$. Consequently, an equation has 2^n projections $(\Pi_Q e_1, \Pi_Q e_2)$ where $\Pi_Q : (V \bigcup C)^* \longrightarrow (Q \bigcup C)^*$ is the projection morphism.

One easily proves the following proposition which reduces the research of a solution to that of a continuous one.

Proposition 1.1 *An equation e has a solution iff one of its projections has a continuous solution.* ∎

The pig-pug method consists in searching for a continuous solution α in the following manner: it visits the lists $e_1[1], \dots, e_1[|e_1|]$ and $e_2[1], \dots, e_2[|e_2|]$ of symbols of e from left to right and at the same time, one tries to guess how their images can overlap. At each step, one makes a non deterministic choice for the relative lengths of the images of the first two symbols $e_1[1]$ et $e_2[1]$. According to the choice made :

$$|\alpha(e_1[1])| < |\alpha(e_2[1])|, \quad |\alpha(e_1[1])| = |\alpha(e_2[1])|, \quad |\alpha(e_1[1])| > |\alpha(e_2[1])|$$

one applies to the equation one of the three substitutions to variables :

$$e_2[1] \leftarrow e_1[1]e_2[1], \quad e_2[1] \leftarrow e_1[1], \quad e_1[1] \leftarrow e_2[1]e_1[1].$$