An Algorithm for the Solution of Tree Equations

Sabrina Mantaci[1]* and Daniele Micciancio[2]**

[1] Dipartimento di Matematica ed Applicazioni, Università di Palermo
via Archirafi, 34 - 90123 Palermo - ITALY
(e-mail: `sabrina@altair.math.unipa.it`)

[2] Laboratory for Computer Science, Massachusetts Institute of Technology
545 Technology Square - Cambridge, MA 02139 - USA
(e-mail: `miccianc@theory.lcs.mit.edu`)

Abstract. We consider the problem of solving equations over k-ary trees. Here an equation is a pair of labeled α-ary trees, where α is a function associating an arity to each label. A solution to an equation is a morphism from α-ary trees to k-ary trees that maps the left and right hand side of the equation to the same k-ary tree.
This problem is a generalization of the word unification problem posed by A. Markov in the fifties, which corresponds to the case k=1, (in this case also the arity function α must be identically equal to 1, and equations are pairs of words). The word unification problem was solved in two steps. First in 1976 Makanin proved the decidability of the existence of a solution to a word equation, and more recently in 1990 Jaffar gave an algorithm that finds the set of all principal solutions to a word equation when this set is finite.
In this paper we solve the α-ary tree equation problem for all other $k > 1$. We describe an efficient unification algorithm that on input an α-ary tree equation, computes a most general (α-ary) solution to the equation if the equation is solvable and reports failure otherwise. This also proves that any satisfiable α-ary tree equation has a most general solution. All k-ary solutions to the equation can be easily obtained from the α-ary solution output by our algorithm.

1 Introduction

The theory of word equations constitutes an important chapter in combinatorics, and appears in several fields of mathematics and theoretical computer science. This theory was first introduced in the fifties by A. A. Markov, who posed in [5] the problem of satisfiability of equations on the free monoid. This has been an open problem until 1976, when G. S. Makanin proved the decidability of the satisfiability problem for equations on words (cf. [3]). More recently, in 1990, J. Jaffar (cf. [2]) designed an algorithm to find the set of all the principal solutions to a word equation (when this set is finite).

* Partially supported by the Italian Ministry of Universities and Scientific Research MURST 40% *Efficienza di Algoritmi e Progetto di Strutture Informative*.
** Partially supported by DARPA grant SDCS DABT63-96-C-0018.

Starting from the notion of equation on words, and the theory of tree-codes introduced by Nivat in [7], a notion of equation on trees was introduced in [4], where the satisfiability problem for word equations is generalized to equations between ordered trees.

A tree equation, as defined in [4], is a pair (τ_1, τ_2) of ordered trees whose nodes are labeled with symbols from an alphabet X. A k-ary solution to a tree equation (τ_1, τ_2) consists of:

1. a function α assigning to each symbol x in X an arity $\alpha(x)$;
2. a pair of α-ary trees t_1 and t_2 whose associated ordered trees are respectively τ_1 and τ_2;
3. a tree morphism φ from the set of α-ary trees to the set of k-ary trees over A such that $\varphi(t_1) = \varphi(t_2)$.

Notice that in the case of word equations considered in [3] the first two step are trivially solved because all symbols must have arity one. It is not known whether the satisfiability problem for tree equations is decidable.

In the present paper, we consider the subproblem of solving equations between α-ary trees corresponding to step 3 above. In other words, we assume the arity function α and the two α-ary trees t_1 and t_2 to be known, and look for a tree morphism φ such that $\varphi(t_1) = \varphi(t_2)$. We show that if (t_1, t_2) is satisfiable, then it has a most general solution. Furthermore, this solution (or that none exists) can be efficiently determined. We give an efficient algorithm, inspired to the Martelli-Montanari unification algorithm (cf. [6]), for the solution of (α-ary) tree equations over k-ary trees for any $k > 1$.

Notice the fundamental difference between the tree equations we consider here and the first order unification problem: in a tree equation the label of internal nodes may be variable symbols, while in the first order unification problem (reformulated in terms of equations between trees) only the leaves may be labeled with variables.

The rest of the paper is organized as follows. In Section 2 we introduce some notation and terminology. In Section 3 we formally define tree equations and the satisfiability problem for such equations. In Section 4 we present an algorithm to solve sets of equations containing exclusively variable symbols. In Section 5 we prove the correctness of the algorithm and analyze its running time. We prove that the number of iterations performed by the algorithm is linear in the size of the problem. In Section 6 we show how to extend our algorithm to solve tree equations with both constants and variables. Finally in Section 7 we conclude by summarizing the content of this paper and by proposing some open problem.

2 Notation

Let $\mathbb{N} = \{0, 1, \ldots\}$ be the set of natural numbers. We will often use $\mathbb{N}$ as a numerable set of symbols. A word over $\mathbb{N}$ is a finite sequence of natural numbers. $\mathbb{N}^*$ denotes the set of words over $\mathbb{N}$. For any two words $v, u \in \mathbb{N}^*$, the concatenation of v with u is denoted by $v \cdot u$. We say that v is a prefix of u

if there exists a word w such that $u = v \cdot w$. A set of words S is *prefix closed* iff for all words v and w, if $v \cdot w \in S$ then also $v \in S$.

Definition 1. A *graded alphabet* $\mathcal{A} = (A; \alpha)$ is a set of letters A endowed with an arity function $\alpha: A \longrightarrow \mathbb{N}$ assigning a natural number to each letter in A.

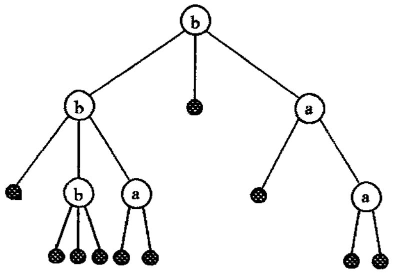
Given two graded alphabets $\mathcal{A} = (A; \alpha)$ and $\mathcal{B} = (B; \beta)$, we say that $\mathcal{B}$ extends $\mathcal{A}$, written $\mathcal{A} \subseteq \mathcal{B}$, if $A \subseteq B$ and $\alpha = \beta|_A$. If $\mathcal{A} = (A; \alpha)$ and $\mathcal{B} = (B; \beta)$ are two graded alphabets and A and B are disjoint, then we denote by $\mathcal{A} \cup \mathcal{B}$ the graded alphabet $(A \cup B; \alpha \vee \beta)$, where $(\alpha \vee \beta)(x) = \alpha(x)$ if $x \in A$, $(\alpha \vee \beta)(x) = \beta(x)$ if $x \in B$.

Definition 2. Let $\mathcal{A} = (A; \alpha)$ be a graded alphabet. A labeled tree over $\mathcal{A}$ is a partial mapping $\tau: \mathbb{N}^* \longrightarrow A$ such that the domain $dom(\tau)$ is a finite and prefix closed subset of $\mathbb{N}^*$ and, for all $v \cdot i \in dom(\tau)$, $i \leq \alpha(\tau(v))$. The elements in $dom(\tau)$ are called *nodes*.

The set of all trees over $\mathcal{A}$ is denoted by $\mathcal{A}^\# = (A; \alpha)^\#$. Ω denotes the empty tree, that is, the tree with no nodes. The *size* of a tree τ is the number of its nodes and it is denoted by $|\tau|$. The set $\mathrm{fr}^+(\tau) = \{u \cdot i \mid u \in dom(\tau),\ u \cdot i \notin dom(\tau)\}$ is called the *outer frontier* of τ. For any tree τ and word $v \in \mathbb{N}^*$, $v^{-1}\tau$ denotes the subtree $\tau': w \mapsto \tau(v \cdot w)$.

Notice that if the arity function α is constant, i.e., $\alpha(a) = k$ for all $a \in A$, then the definition of tree over $(A; \alpha)$ reduces to the traditional notion of k-ary tree.

Example 1. If $A = \{a, b\}$ and $\alpha(a) = 2$, $\alpha(b) = 3$, then the tree



is an α-ary tree over A, whose domain is the set $\{\epsilon, 1, 3, 12, 13, 32\}$ and its outer frontier is the set $\mathrm{fr}^+(\tau) = \{2, 11, 31, 121, 122, 123, 131, 132, 321, 322\}$.

Remark. Let $\mathcal{A} = (A; \alpha)$ be a graded alphabet. It can be easily proved by induction that for all trees τ over $\mathcal{A}$

$$|\mathrm{fr}^+(\tau)| = \left(\sum_{a \in A} (\alpha(a) - 1) n_a \right) + 1$$

where $n_a = |\tau^{-1}(a)|$ denotes the number of nodes labeled with a in the tree τ.

Notice that, for a fixed $\mathcal{A}$, the value of expression $(\sum_i(\alpha(a_i)-1)n_i)+1$ may not range over the whole set of positive integers. Therefore the set $\mathcal{A}^\#$ might not contain any tree with outer frontier of a given cardinality.

In particular for any k-ary tree τ of size $|\tau| = n$ the cardinality of the outer frontier is $|\mathrm{fr}^+(\tau)| = (k-1)n+1$. Notice how the size of the outer frontier uniquely determines the size of the tree when $k > 1$.

If $k = 1$ (i.e., in the case of words) the outer frontier has cardinality 1 independently from the size of the word, and in this case the size of the outer frontier gives no information about the size of the word.

If $k > 1$, there are trees τ such that $|\mathrm{fr}^+(\tau)| = n$ if and only if $k-1$ divides $n-1$. In particular, for binary trees we have $|\mathrm{fr}^+(\tau)| = |\tau|+1$ and there are trees with $|\mathrm{fr}^+(\tau)| = n$ for any positive integer n.

We now define a fundamental operation over trees.

Definition 3. Let τ_1 and τ_2 be two trees over a graded alphabet $\mathcal{A}$ and let $b \in \mathrm{fr}^+(\tau_1)$ be a node in the outer frontier of τ_1. The concatenation of τ_2 to τ_1 at b is the tree $\tau_1(b)\,\tau_2$ with domain $dom(\tau_1\,(b)\tau_2) = dom(\tau_1) \cup b \cdot dom(\tau_2)$ defined by

$$(\tau_1\,(b)\,\tau_2)(u) = \begin{cases} \tau_1(u) \text{ if } u \in dom(\tau_1) \\ \tau_2(v) \text{ if } bv = u \textit{ and } v \in dom(\tau_2) \end{cases}$$

Let τ be a tree with outer frontier $\mathrm{fr}^+(\tau) = \{b_1, b_2, \ldots, b_n\}$ and let $\tau_1, \ldots, \tau_n$ be a sequence of n trees. We denote by $\tau\langle\tau_1, \tau_2, \ldots, \tau_n\rangle$ the result of concatenating the trees $\tau_1, \tau_2, \ldots, \tau_n$ to τ at $\{b_1, b_2, \ldots, b_n\}$, i.e.,

$$\tau\langle\tau_1, \tau_2, \ldots, \tau_n\rangle = (\ldots((\tau(b_1)\tau_1)(b_2)\tau_2)\ldots(b_n)\tau_n).$$

Definition 4. Given two graded alphabets $\mathcal{A} = (A;\alpha)$ and $\mathcal{B} = (B;\beta)$, a *morphism* of $\mathcal{A}^\#$ into $\mathcal{B}^\#$ is an application $\varphi: \mathcal{A}^\# \to \mathcal{B}^\#$ such that, for any $a \in A$, $|\mathrm{fr}^+(\varphi(a))| = \alpha(a)$ and φ preserves concatenation, i.e., for any tree τ with $|\mathrm{fr}^+(\tau)| = n$ and for any sequence of trees $\tau_1, \tau_2, \ldots, \tau_n$

$$\varphi(\tau\langle\tau_1, \tau_2, \ldots, \tau_n\rangle) = \varphi(\tau)\langle\varphi(\tau_1), \varphi(\tau_2), \ldots, \varphi(\tau_n)\rangle$$

Proposition 5. *Any function φ from $\mathcal{A}$ to $\mathcal{B}^\#$ such that $|\mathrm{fr}^+(\varphi(a))| = \alpha(a)$ for all $a \in A$, can be extended to a unique morphism $\tilde{\varphi}: \mathcal{A}^\# \to \mathcal{B}^\#$.*

We write $[\tau/x]$ to denote the morphism φ such that $\varphi(x) = \tau$ and $\varphi(y) = y$ for any other $y \neq x$. We say that a morphism $\varphi: \mathcal{A}^\# \to \mathcal{B}^\#$ is *non erasing* if $\varphi(\tau) = \Omega$ implies $\tau = \Omega$. For example if $A = \{a, b\}$ where a has arity one the morphism $\varphi: \mathcal{A} \to \mathcal{A}$ such that $\varphi(a) = \Omega$ and $\varphi(b) = b$ is not non erasing.

Notice that for some $\mathcal{A}$ and $\mathcal{B}$ there might be no morphisms from $\mathcal{A}$ to $\mathcal{B}$. For example if $\mathcal{A} = (A;2)$ is a binary alphabet and $\mathcal{B} = (B;3)$ is a ternary alphabet there are no morphisms from $\mathcal{A}^\#$ to $\mathcal{B}^\#$ because all trees in $\mathcal{B}^\#$ have outer frontier of odd size.

3 Tree Equations

In this section we introduce the notion of equation on graded trees. We remark that we are interested in finding solutions for such equations over the set of k-ary trees over A for some $k \geq 2$ and some alphabet A. Notice that for $k \geq 2$ the only k-ary tree with outer frontier of size one is the empty tree. Therefore it is not restrictive to assume that all variable symbols have arity ≥ 2.

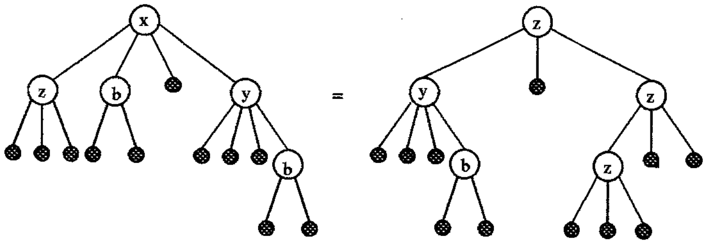
Moreover, under this assumption, for any fixed pair of finite graded alphabets $\mathcal{A}$ and $\mathcal{B}$, there exist at most a finite number of morphisms from $\mathcal{A}$ to $\mathcal{B}$ and all morphisms are non erasing.

Let $\mathcal{X} = (X; \chi)$ and $\mathcal{A} = (A; \alpha)$ be two graded alphabet with arity functions $\chi(x), \alpha(a) \geq 2$ for all $x \in X$ and $a \in A$. For notational convenience, we will consider $\mathcal{A}$ and $\mathcal{X}$ as fixed throughout the rest of the paper. We will also assume that $\mathcal{X}$ contains infinitely many symbols for each arity. We will call $\mathcal{X}$ the set of variables and $\mathcal{A}$ the set of constants.

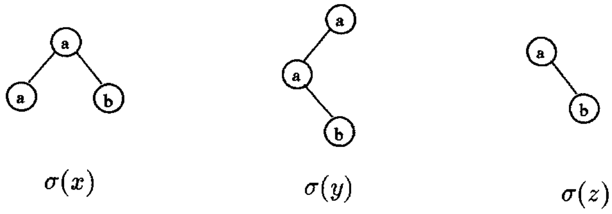
Definition 6. A *tree equation* is a pair of trees (τ_1, τ_2) in $(\mathcal{X} \cup \mathcal{A})^{\#}$.

We say that a tree equation (τ_1, τ_2) admits a solution in $\mathcal{B}^{\#}$, where $\mathcal{A} \subseteq \mathcal{B}$, if there exists a non erasing morphism $\sigma: (\mathcal{X} \cup \mathcal{A})^{\#} \to \mathcal{B}^{\#}$ such that $\sigma(a) = a$ for all $a \in A$, and $\sigma(\tau_1) = \sigma(\tau_2)$. Since the arities of the graded alphabets we are considering are greater then 2, then all our morphisms are non erasing. In the sequel we will take the non erasing property as granted and we will refer to solutions to tree equations simply as "morphisms".

Example 2. We give here an example of a tree equation over the graded alphabet $(\{x, y, z\}; \chi(x) = 4, \chi(y) = 4, \chi(z) = 3)$ and the set of constants $(\{b\}; \alpha(b) = 2)$.



It is easy to verify that the morphism σ defined on the variables as follows:



is a solution in the set of binary trees over $\{a, b\}$

Definition 7. A *tree system* is a finite set of tree equations

$$S = \{(\tau_{i,1}, \tau_{i,2}) \mid i = 1, 2, \ldots, n;\ \tau_{i,1}, \tau_{i,2} \in (\mathcal{X} \cup \mathcal{A})^{\#}\}.$$

A *solution* in $\mathcal{B}$ of a tree system is a morphism $\sigma: (\mathcal{X} \cup \mathcal{A})^{\#} \to \mathcal{B}^{\#}$ that simultaneously solves all equations in S.

It is evident that for this kind of equations (systems) the problem of the existence of solutions is decidable. In fact, as remarked before, there exists only a finite number of morphisms between two sets of trees over two graded alphabets, and then it is possible to test all of them. In this case we are faced to an algorithmic problem. We have to find a fast algorithm to obtain the set of the solutions to a tree equation (system). This will be the content of next section.

Notice that a tree system is always equivalent to a tree equation. Namely, the system $S = \{(\tau_{i,1}, \tau_{i,2}) \mid i = 1, 2, \ldots, n\}$ is equivalent to the single tree equation

$$(x\langle \tau_{1,1}, \tau_{1,2}, \ldots \tau_{1,n}\rangle, x\langle \tau_{2,1}, \tau_{2,2}, \ldots \tau_{2,n}\rangle)$$

where x is a new variable with arity $\chi(x) = n$.

Let $\mathcal{V}$ be a finite subalphabet of $\mathcal{X}$ and let $\mathcal{A} = (A; \alpha)$, $\mathcal{B} = (B; \beta)$ be two arbitrary graded alphabets such that $\mathcal{A} \subseteq \mathcal{B}$.

Definition 8. A *variable assignment* σ is a function from $\mathcal{V}$ to $\mathcal{B}^{\#}$ such that $|\mathrm{fr}^{+}(\sigma(x))| = \chi(x)$ for all x in V. Any assignment $\sigma: \mathcal{V} \to \mathcal{B}^{\#}$ can be extended to a unique morphism $\tilde{\sigma}: (\mathcal{V} \cup \mathcal{A})^{\#} \to \mathcal{B}^{\#}$ such that $\tilde{\sigma}(a) = a$ for all $a \in A$. A *solution* to a system of tree equations S over $\mathcal{V} \cup \mathcal{A}$ is a variable assignment $\sigma: \mathcal{V} \to \mathcal{B}^{\#}$ such that $\tilde{\sigma}(\tau_1) = \tilde{\sigma}(\tau_2)$ for all $(\tau_1, \tau_2) \in S$.

Definition 9. Let $\sigma: \mathcal{V} \to \mathcal{A}^{\#}$ and $\rho: \mathcal{V} \to \mathcal{B}^{\#}$ be two assignments. We say that σ is *more general than* ρ, written $\sigma \sqsubseteq \rho$, iff there exists a morphism $\mu: \mathcal{A}^{\#} \to \mathcal{B}^{\#}$ such that $\sigma \circ \mu = \rho$.

The relation $\sqsubseteq$ defines a preorder on the set of solutions.

Definition 10. A solution σ to a system of equations S is a *most general solution* iff σ is a solution to S, and σ is more general than any other solution ρ to S.

Theorem 11. *Let S be a finite system of equations over $\mathcal{V}$. If S is solvable, then it has a most general solution.*

The proof of the above theorem is constructive, i.e. we will give an algorithm that on input a system of equations outputs a most general solution, if one exists, and reports failure otherwise. Moreover, the algorithm is efficient, i.e., it is polynomial.

4 The Algorithm

In this section we describe an algorithm to solve systems of constant-free tree equations, i.e. equations between trees containing only variable symbols. We will then show (see Section 6) that the problem of solving tree equations with both constants and variables can be easily reduced to the constant-free problem.

First of all we will establish a few facts about tree equations.

Lemma 12. *Every assignment $\sigma: \mathcal{V} \rightarrow \mathcal{A}^{\#}$ is a solution to the equation (Ω, Ω).*

Lemma 13. *If $\mathrm{fr}^{+}(\tau_1) \neq \mathrm{fr}^{+}(\tau_2)$ then the equation (τ_1, τ_2) has no solutions.*

Lemma 14. *Let τ be a tree over the variables $\mathcal{V}$, and x a variable not in V. The assignment $\sigma: \mathcal{V} \cup \{x\} \rightarrow \mathcal{A}^{\#}$ is a solution to the equation (x, τ) if and only if $\sigma = [\tau/x] \circ \rho$ for some variable assignment $\rho: \mathcal{V} \rightarrow \mathcal{A}^{\#}$.*

The above lemmas describe the set of solutions to the equations (τ_1, τ_2) where either $|\mathrm{fr}^{+}(\tau_1)| \neq |\mathrm{fr}^{+}(\tau_2)|$ or one of the trees is either punctual or empty. Otherwise we start by associating to each of the trees τ_1 and τ_2 an equivalence relation over the set $\{1, \ldots, n\}$ where $n = |\mathrm{fr}^{+}(\tau_1)| = |\mathrm{fr}^{+}(\tau_2)|$.

Definition 15. Let τ be a non empty tree with $(b_1, \ldots, b_n)$ the (lexicographically) ordered sequence of nodes in the outer frontier of τ. We define the equivalence relation $\sim_\tau$ by $i \sim_\tau j$ iff b_i and b_j begin with the same symbol. In other words, i and j are in relation $\sim_\tau$ iff the i^{th} and j^{th} node of the outer frontier of τ are in the same first-level subtree of τ.

When we consider the equivalence classes $(C_1, \ldots, C_k)$ of a relation $\sim_\tau$ we will always assume that they are ordered in the obvious way: if $i < j$ then $i' < j'$ for all $i' \in C_i$ and $j' \in C_j$.

Let x and y be the labels of the roots of τ_1 and τ_2 and let $X_1, \ldots, X_{\chi(x)}$, $Y_1, \ldots, Y_{\chi(y)}$ be the equivalence classes of $\sim_{\tau_1}$ and $\sim_{\tau_2}$. Notice that $\sim_{\tau_1} \cup \sim_{\tau_2}$ is an equivalence relation if and only if, for any X_i $(1 \leq i \leq \chi(x))$ and any Y_j $(1 \leq j \leq \chi(y))$, $X_i \cap Y_j$ is either the empty set, or equals one of the two sets X_i and Y_j. This corresponds to the fact that in a tree any two subtrees are either disjoint or one a subtree of the other.

Lemma 16. *If the equation (τ_1, τ_2) is satisfiable then $(\sim_{\tau_1} \cup \sim_{\tau_2})$ is an equivalence relation.*

Now assume that $\sim_{\tau_1} \cup \sim_{\tau_2}$ is an equivalence relation and let $(V_1, \ldots, V_m)$ be its equivalence classes. Notice that for all $k \in \{1, \ldots, m\}$ we have $V_k = X_i \cup X_{i+1} \cup \cdots \cup X_{i+i'} = Y_j \cup Y_{j+1} \cup \cdots \cup Y_{j+j'}$ for some i, i', j, j' such that either $i' = 0$ or $j' = 0$ (or both). Let v be a new variable of arity $\chi(v) = m$. For all $k = 1, \ldots, m$ we introduce a fresh variable w_k of arity $\chi(w_k) = i' + j' + 1$ and define the equation

$$E_k = (t_k^x \langle i^{-1}\tau_1, \ldots, (i+i')^{-1}\tau_1 \rangle, t_k^y \langle j^{-1}\tau_2, \ldots, (j+j')^{-1}\tau_2 \rangle)$$

where t_k^x (resp. t_k^y) is the empty tree Ω if $i' = 0$ (resp. $j' = 0$), or is the punctual tree w_k otherwise.

Lemma 17. *Let (τ_1, τ_2) be an equation such that $n = |\mathrm{fr}^+(\tau_1)| = |\mathrm{fr}^+(\tau_2)| \geq 2$ and $\sim_{\tau_1} \cup \sim_{\tau_2}$ is an equivalence relation. The assignment σ is a solution to (τ_1, τ_2) if and only if $\sigma = \pi \circ \rho$ where $\pi = [v\langle t_1^x, \ldots, t_m^x\rangle/x, v\langle t_1^y, \ldots, t_m^y\rangle/y]$ and ρ is a solution to the system $\pi(\{E_1, \ldots, E_m\})$.*

An algorithm for the solution of tree equations is obtained by combining the results of previous lemmas. Let S be a finite set of equations over $\mathcal{V}$. The algorithm generates a sequence of triples $(\mathcal{V}_i, S_i, \sigma_i)$ where

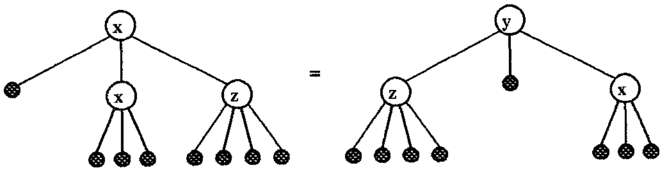
- $\mathcal{V}_i$ is a finite subalphabet of $\mathcal{X}$,
- S_i is a finite system of equations over $\mathcal{V}_i$,
- σ_i is a variable assignment from $\mathcal{V}$ to $\mathcal{V}_i^{\#}$.

The algorithm starts from $(\mathcal{V}_0, S_0, \sigma_0) = (\mathcal{V}, S, \mathrm{id}_{\mathcal{V}})$ and iteratively computes $(\mathcal{V}_{i+1}, S_{i+1}, \sigma_{i+1})$ from $(\mathcal{V}_i, S_i, \sigma_i)$ until either $S_i = \emptyset$ or $S_i = \{False\}$. $(\mathcal{V}_{i+1}, S_{i+1}, \sigma_{i+1})$ is computed from $(\mathcal{V}_i, S_i, \sigma_i)$ as follows. An equation (τ_1, τ_2) is selected from S_i (any selection strategy is good). Then, the computation proceeds by cases corresponding to lemmas 12, 13, 14 16 and 17.

1. If both τ_1 and τ_2 are empty ($\tau_1 = \tau_2 = \Omega$) then $S_{i+1} = S_i \setminus \{(\tau_1, \tau_2)\}$, $\mathcal{V}_{i+1} = \mathcal{V}_i$ and $\sigma_{i+1} = \sigma_i$.
2. If $|\mathrm{fr}^+(\tau_1)| \neq |\mathrm{fr}^+(\tau_2)|$ then $S_{i+1} = \{False\}$, $\mathcal{V}_{i+1} = \mathcal{V}_i$ and $\sigma_{i+1} = \sigma_i$.
3. If τ_1 (resp. τ_2) is the punctual tree x then $\mathcal{V}_{i+1} = \mathcal{V}_i \setminus \{x\}$, $\pi_i = [\tau_2/x]$ (resp. $\pi_i = [\tau_1/x]$), $S_{i+1} = \pi_i(S_i \setminus \{(\tau_1, \tau_2)\})$, and $\sigma_{i+1} = \sigma_i \circ \pi_i$.
4. If none of the previous cases applies, let $n = |\mathrm{fr}^+(\tau_1)| = |\mathrm{fr}^+(\tau_2)| \geq 2$.
   (a) If $\sim_{\tau_1} \cup \sim_{\tau_2}$ is not an equivalence relation then $S_{i+1} = \{False\}$, $\mathcal{V}_{i+1} = \mathcal{V}_i$ and $\sigma_{i+1} = \sigma_i$.
   (b) Otherwise, let $E_1, \ldots, E_m, x, y$ and π be as in Lemma 17 and define $\mathcal{V}_{i+1} = \mathcal{V} \setminus \{x, y\} \cup \{v, w_1, \ldots, w_m\}$, $\pi_i = \pi$, $S_{i+1} = \pi_i(S_i \setminus \{(\tau_1, \tau_2)\} \cup \{E_1, \ldots, E_m\})$, and $\sigma_{i+1} = \sigma_i \circ \pi_i$.

The algorithm terminates as soon as $S_i = \emptyset$ or $S_i = \{False\}$. Upon termination, if S_i is empty the algorithm outputs σ_i, otherwise it reports failure.

Example 3. Consider the equation



where $\chi(x) = \chi(y) = 3$ and $\chi(z) = 4$. A more compact notation for this equation is $x(\cdot, x(\cdot,\cdot,\cdot), z(\cdot,\cdot,\cdot,\cdot)) = y(z(\cdot,\cdot,\cdot,\cdot), \cdot, x(\cdot,\cdot,\cdot))$. Let us run the algorithm over this equation.
Iteration 1. Case 4 applies. Here $n = 8$ and the equivalence classes of $\sim_y$ and $\sim_x$ are given by $X'_1 = \{1\}, X'_2 = \{2,3,4\}, X'_3 = \{5,6,7,8\}, Y'_1 = \{1,2,3,4\}, Y'_2 =$

$\{5\}, Y_3' = \{6,7,8\}$. The union $\sim_y \cup \sim_x$ is an equivalence relation with equivalence classes $V_1 = \{1,2,3,4\}, V_2 = \{5,6,7,8\}$.

So, we generate two new equations, which both happen to be equal to

$$z(\cdot,\cdot,\cdot,\cdot) = w(\cdot, x(\cdot,\cdot,\cdot,\cdot))$$

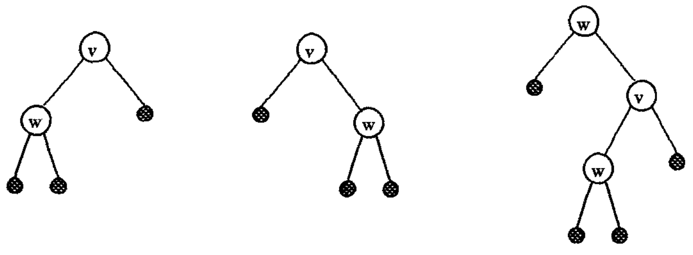
In conclusion we have:

$$\begin{aligned}
\mathcal{V}_1 &= (\{z,v,w\}; \chi(z) = 4, \chi(v) = 2, \chi(w) = 2)\\
\sigma_1 &= [v(w(\cdot,\cdot),\cdot)/x, v(\cdot, w(\cdot,\cdot))/y]\\
S_1 &: z(\cdot,\cdot,\cdot,\cdot) = w(\cdot, v(w(\cdot,\cdot),\cdot))
\end{aligned}$$

Iteration 2. This time case 3 applies and we have

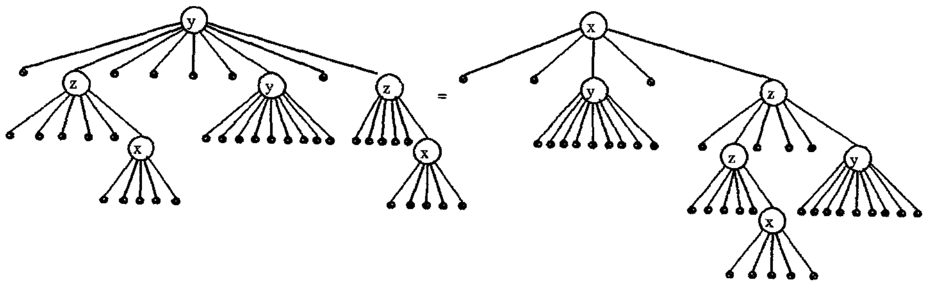
$$\begin{aligned}
\mathcal{V}_2 &= (\{v,w\}; \chi(v) = 2, \chi(w) = 2)\\
\sigma_2 &= \sigma_1 \circ [w(\cdot, v(w(\cdot,\cdot),\cdot))/z]\\
S_2 &: \emptyset
\end{aligned}$$

At this point the algorithm terminates and output the assignment $\sigma = \pi_1 \circ \pi_2$ defined by $\sigma(x) = v(w(\cdot,\cdot),\cdot)$, $\sigma(y) = v(\cdot, w(\cdot,\cdot))$ and $\sigma(z) = [w(\cdot, v(w(\cdot,\cdot),\cdot))]$, or more pictorially,



$\sigma(x)$ $\sigma(y)$ $\sigma(z)$

Example 4. We run the algorithm over a more complicated equation over the graded alphabet $(\{x,y,z\}; \chi(x) = 5, \chi(y) = 9, \chi(z) = 6)$ defined as follows:

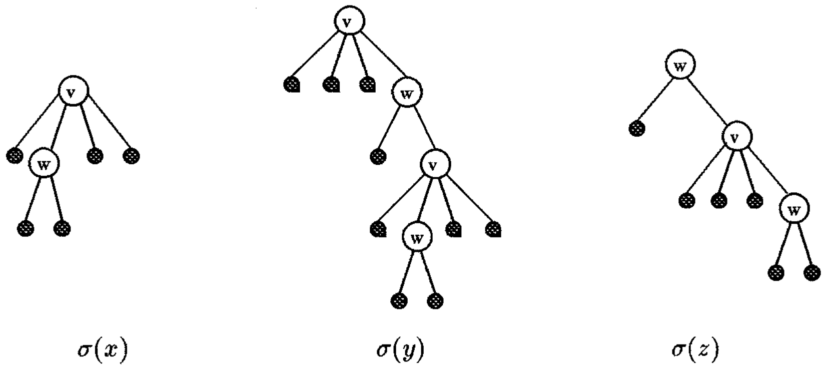


The solution is given by the following variable assignment:

$$\sigma(x) = v(\cdot, w(\cdot,\cdot),\cdot,\cdot)$$

$$\sigma(y) = v(\cdot,\cdot,\cdot,w(\cdot,v(\cdot,w(\cdot,\cdot),\cdot,\cdot)))$$
$$\sigma(z) = w(\cdot,v(\cdot,\cdot,\cdot,w(\cdot,\cdot)))$$

that is



If we are looking for a solution over the set of binary trees, then we can map w to a punctual tree, and v to any tree with three nodes.

5 Correctness of the Algorithm

In this section we will show that the algorithm always terminates and gives the right answer. That is, the algorithm outputs an assignment σ, if and only if the input system is satisfiable, and in this case σ is the most general solution to the system.

Theorem 18. *The algorithm always terminates. Moreover the number of iterations performed is linear in the size of the system.*

Proof. Define the weight of equation (τ_1, τ_2) to be

$$w(\tau_1,\tau_2) = |\mathrm{fr}^+(\tau_1)| + |\mathrm{fr}^+(\tau_2)| - 1.$$

Define also the weight of a system by

$$W(S) = \begin{cases} 0 & \text{if } S = \{False\} \\ \sum_{e\in S} w(e) & \text{otherwise} \end{cases}$$

We prove that $W(S_i)$ decreases at each iteration. Since $W(S_i)$ is always a non negative integer, this proves that the algorithm stops after at most $W(S)$ iterations. The proof of $W(S_{i+1}) < W(S_i)$ is by cases on the branch taken by the algorithm. In case 1 and 3, $W(S_{i+1}) = W(S_i) - w(\tau_1,\tau_2) < W(S_i)$ (notice that the outer frontier of the empty tree has cardinality 1). In case 2 and 4a, $W(S_{i+1}) = 0 < W(S_i)$. Finally, in case 4b $W(S_{i+1}) = W(S_i) - w(\tau_1,\tau_2) + \sum_{k=1}^{m} w(E_k) < W(S_i)$ because $\sum_{k=1}^{m} w(E_k) = w(\tau_1,\tau_2) - m + 1$ and $m \geq 2$.

The correctness of the algorithm is based on the following lemma.

Lemma 19. *For all i, $\sigma: \mathcal{V} \to \mathcal{A}^{\#}$ is a solution to S iff $\sigma = \sigma_i \circ \tilde{\rho}_i$ for some solution $\rho_i: \mathcal{V}_i \to \mathcal{A}^{\#}$ to S_i.*

Proof. (Sketch) The proof is by induction on i. If $i = 0$, then $S_0 = S$ and the lemma is obviously true. The inductive step is proved by cases on the branch taken by the algorithm using lemmas 12, 13, 14, 16 and 17.

Using the above lemma the proof of correctness of the algorithm is immediate.

Theorem 20. *The algorithm outputs an assignment σ if and only if the system is satisfiable, and in such a case σ is a most general solution to the system.*

Proof. By Theorem 18 the algorithm always terminates. Therefore, for some n either $S_n = \emptyset$ or $S_n = \{False\}$. If $S_n = \emptyset$, the algorithm outputs σ_n. By Lemma 19, σ_n is a solution to S, so S is satisfiable. Moreover, for any solution $\sigma': \mathcal{V} \to \mathcal{A}^{\#}$ to S, there exists an assignment $\rho: \mathcal{V}_n \to \mathcal{A}^{\#}$ such that $\sigma' = \sigma \circ \rho$. This proves that σ is a most general solution to S. Conversely, if $S_n = \{False\}$, the algorithm terminates with failure. By Lemma 19 if σ is solution to S, then $\sigma = \sigma_n \circ \rho$ for some solution ρ to S_n, but this is impossible because S_n is unsatisfiable. Therefore also S must be unsatisfiable.

Theorem 11 on the existence of most general solutions follows immediately from Theorem 20.

6 Equations with Constants

We now consider equations containing constants. Let S be a system of equations over variables $\mathcal{X}$ and constants $\mathcal{A}$ and let $\mathcal{B}$ be a graded alphabet such that $\mathcal{A} \subseteq \mathcal{B}$. We introduce a new variable x_a with arity $\chi(x_a) = \alpha(a)$ for each constant symbol $a \in A$ and define the constant-free system S' over the variables $\mathcal{X} \cup \mathcal{X}'$ by replacing each constant a in S by the corresponding variable x_a.

There is an obvious bijection between the set of solutions $\rho: \mathcal{A} \cup \mathcal{X} \to \mathcal{B}^{\#}$ to S and the set of solutions $\rho': \mathcal{X}' \cup X \to \mathcal{B}^{\#}$ to S' such that $\rho'(x_a) = a$ for all $a \in A$. Therefore the study of the system S can be reduced to the study of the constant-free system S' with the additional constraint $\rho'(x_a) = a$.

By Theorem 11 the system S' has a most general solution $\sigma: \mathcal{X}' \cup \mathcal{X} \to (\mathcal{X}' \cup \mathcal{X})^{\#}$ and $\rho': \mathcal{X}' \cup \mathcal{X} \to \mathcal{B}^{\#}$ solves S' iff $\rho' = \sigma \circ \pi$ for some $\pi : \mathcal{X}' \cup \mathcal{X} \to \mathcal{B}^{\#}$. We also want $\rho'(x_a) = \pi(\sigma(x_a)) = a$, but this is possible iff $\sigma(x_a)$ is a punctual tree and $\sigma(x_a) \neq \sigma(x_b)$ for all $a \neq b$. Hence we have the following theorem.

Theorem 21. *The system S is solvable iff S' has a most general solution σ such that for all $a \in A$, $\sigma(x_a)$ is punctual and for all $a \neq b$, $\sigma(x_a) \neq \sigma(x_b)$.*

7 Conclusion

We defined an algorithm that on input a set of equations between graded trees, determines a most general solution to the equations if a solution exits, and reports failure otherwise. In particular this solves our initial problem of finding, for χ-ary tree equations, solutions over k-ary trees, as any k-ary solution ρ can be expressed as the composition of the most general solution σ over $(X;\chi)$ with a morphism μ from $(X;\chi)$ to $(A;k)$. Notice that an equation can have no solution over k-ary trees for some k, even if it has a most general solution over $(X;\chi)$. This is because there could not exists any morphism from $(X;\chi)$ to $(A;k)$. However, if a most general solution over $(X;\chi)$ exists, then the system has solutions over binary trees because it is always possible to find a morphism from $(X;\chi)$ to $(A;2)$ when $\chi(x) \geq 2$ for all $x \in X$.

The algorithm partially solves the problem proposed in [4], that is, the problem of finding a solution of equations between ordered trees. Anyway, the problem of solvability of ordered tree equations remains open, since we do not have an efficient procedure to assign the arity function to the variables in the equations. Actually solvability of equations of ordered trees can be reduced to a particular case of the second order unification (that, in general, is undecidable). However we don't know if second order unification becomes decidable for this particular subclass of equations. In fact, the proof of the undecidability of second order unification (cf. [1]) cannot be extended directly to our case.

Notice also that the algorithm in Section 4 allows to find solutions in $(A;\alpha)^{\#}$ also for non constant arity functions α, provided that $\alpha(a) > 1$ for all a. This problem is in some sense complementary to the word equation problem considered in [3]. In [3] it is shown how to solve equations between trees all of whose nodes have arity one (i.e., they are words). Here we solved the problem for trees whose node have arity greater than one.

An interesting question that we leave open, is whether these two results can be combined to give an algorithm to solve equations over arbitrary trees.

References

1. Goldfarb, W.: The Undecidability of the Second-Order Unification Problem. *Theoretical Computer Science* **13** (1981) 225–230.
2. Jaffar, J.: Minimal and complete word unification. *J. ACM* **37** (1990) 47–85.
3. Makanin,G. S.: The problem of solvability of equations in a free semigroup. *Math, USSR Sbornik* **32** (1977) (in AMS 1979) 129–198.
4. Mantaci, S., Restivo, A.: Equations on trees. em Proc. of 21st MFCS(1996), vol.1113 of *LNCS* 443–456.
5. Markov, A. A.: *The Theory of Algorithm.* Trudy Math. Inst. Steklov, 42, (1954).
6. Martelli, A., Montanari, U.: An efficient unification algorithm, TOPLAS 4:2 (1982), 258–282.
7. M. Nivat: Binary tree codes. Tree automata and languages,*Elsevier Science Publishers B.V. (North-Holland)*,(1992). 1–19