The Journal of Functional and Logic Programming

The MIT Press

Volume 1998, Article 4
28 May, 1998

ISSN 1080–5230. MIT Press Journals, Five Cambridge Center, Cambridge, MA 02142-1493, USA; (617)253-2889; *journals-orders@mit.edu, journals-info@mit.edu*. Published one article at a time in LaTeX source form on the Internet. Pagination varies from copy to copy. For more information and other articles see:

- *http://www.cs.tu-berlin.de/journal/jflp/*
- *http://mitpress.mit.edu/JFLP/*
- *gopher.mit.edu*
- *ftp://mitpress.mit.edu/pub/JFLP*

The Journal of Functional and Logic Programming is a peer-reviewed and electronically published scholarly journal that covers a broad scope of topics from functional and logic programming. In particular, it focuses on the integration of the functional and the logic paradigms as well as their common foundations.

A Strict Border for the Decidability of E-Unification for Recursive Functions*

Heinz Faßbender Sebastian Maneth

28 May, 1998

Abstract

During the execution of functional logic programs, E-unification problems have to be solved quite frequently, where the underlying equational theory is induced by recursive functions. But, what about the decidability of those E-unification problems? Up to now, there does not exist a concrete answer to this question for classes of equational theories which are induced by particular recursive functions. In this paper, we try to give an answer to this question by drawing and verifying a strict border between undecidability and decidability of E-unification problems for particular classes of recursive functions. Since this result shows that the E-unification problem is undecidable even for a very restricted class of recursive functions, the nondeterministic implementations of those problems in functional logic programming languages are justified.

1 Introduction

During the execution of functional logic programs, certain E-unification problems must frequently be solved. Consider, e.g., the program of the functional logic programming language BABEL [MR92] in Figure 1. This program defines the predicate *sublist* and the function *append*; *sublist* checks whether its first argument is a sublist of its second argument, and *append* concatenates two lists in the usual way.

*A preliminary version of this paper appeared as [FM96].

$sublist(x, y)$	$:-$	$y = append(z_1, append(x, z_2))$
$append(CONS(x_1, x_2), y_1)$	$:=$	$CONS(x_1, append(x_2, y_1))$
$append(NIL, y_1)$	$:=$	y_1

Figure 1: A functional logic program

In computations of the predicate *sublist*, an equation such as

$$t_y = append(z_1, append(t_x, z_2))$$

must be solved, where t_y and t_x are the current values of the variables y and x, respectively. Clearly, this is just the E_{append}-unification problem for the terms t_y and $append(z_1, append(t_x, z_2))$; and the computing machinery tries to compute an E_{append}-unifier φ, i.e., it tries to answer the question whether t_y and $append(z_1, append(t_x, z_2))$ are E_{append}-unifiable.

From the implementational point of view, for functional logic programming languages, it is very important to know whether the considered E-unification problem is decidable, i.e., whether there exists an algorithm that yields an answer for every possible input. Note that an input of such an algorithm consists of a set E of recursive function definitions and two terms t and s which should be E-unified.

In general, it is well known that the decidability of an E-unification problem depends on the set E of equations. For example, if E is the empty set, then the E-unification problem coincides with the usual unification problem of terms, which is decidable [Rob65, MM82]. If E is the set of Peano's axioms, then the E-unification problem coincides with Hilbert's tenth problem, which was shown to be undecidable [Mat70]. However, there are many specific theories that are decidable, such as associativity A [Mak77], associativity and commutativity AC [Sti75], etc. (see [BS94] for a survey). When dealing with functional logic programs, we are rather interested in *general* results that hold for a whole class of theories. Usually, such classes of E-unification problems are characterized by particular classes of term-rewriting systems (TRS; see, e.g., [DJ90]), referring to their ability to describe (directed) equalities. In this context, there are a number of results on decidability of E-unification for particular classes of TRSs, which will be discussed in Section 4. As we will see, from a programming point of view, some of the classes considered in Section 4 are much more practical than the classes for which we show decidability. However, our intention was not to put very specific conditions on the

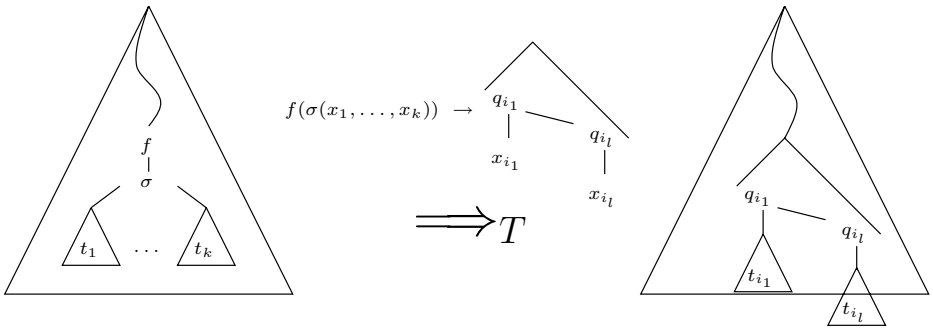

Figure 2: A derivation step of a top-down tree transducer

TRSs (or even on the goals), but rather to show for known classes of TRSs where the border of un-/decidability of the induced E-unification problems lies. We feel that this gives better insight into decidability of E-unification from a "syntactic restriction" point of view.

In particular, we show that the E-unification problem is undecidable for the class $\mathcal{E}_{tdT}$ of sets of recursive functions, which are induced by very restricted term-rewriting systems called *top-down tree transducers*. In other words, we show that there exists a set $E \in \mathcal{E}_{tdT}$ and two terms t and s such that it is not decidable whether t and s are E-unifiable.

Intuitively, a top-down tree transducer [Tha70, Rou70] is a left-linear, confluent, terminating, constructor-based [You89], and strictly not subunifiable [Ech90] TRS, such that every function symbol has rank 1 and for every function symbol f and every constructor σ of rank $k \geq 0$, there exists exactly one rewrite rule of the form

$$f(\sigma(x_1, \ldots, x_k)) \to r$$

where r is a term over constructors and function calls of the form $g(x_i)$, such that g is a function symbol and x_i is a variable from the left-hand side. A derivation step of such a tree transducer is shown in Figure 2.

For example, in Figure 3 we present the rewrite rules of a top-down tree transducer that defines a function f which computes Fibonacci numbers. Since it is only allowed to call the predecessor of the left-hand side's argument in the right-hand side, we use a second function h to compute f on the predecessor of the predecessor of the left-hand side's argument. In other words, we turn course of value recursion into simultaneous recursion. Natural

$$
\begin{array}{lcl}
f(0) & \to & s(0) \\
f(s(x_1)) & \to & +(f(x_1), h(x_1)) \\
h(0) & \to & 0 \\
h(s(x_1)) & \to & f(x_1)
\end{array}
$$

Figure 3: Rewrite rules of a top-down tree transducer for computing Fibonacci numbers

numbers are described by terms over the input symbols s and 0 of rank 1 and 0, respectively. Furthermore, the addition is presented as the binary constructor $+$, which has to be interpreted if the Fibonacci number should really be evaluated.

In particular, we will show that the E-unification problem is even undecidable for the class of equational theories which are induced by top-down tree transducers with *two* different function symbols, and where the rank of every constructor is less than or equal to 1, i.e., *monadic trees*.

Furthermore, we show that the considered class of sets of recursive functions is minimal with respect to undecidability; i.e., if we restrict the class of sets of recursive functions which are considered, then the E-unification problem becomes decidable.

That means, the main point of the paper is drawing and verifying a strict border between decidable and undecidable equational theories that are induced by TRSs that define recursive functions. For this purpose, we consider monadic tree homomorphisms and monadic top-down tree transducers. The difference between these two is the number of function symbols. Since the border between undecidability and decidability lies between these two classes, the number of functions seems to be crucial with respect to the decidability of E-unification for recursive functions.

This paper is organized as follows. Section 2 contains preliminaries. The considered TRSs and E-unification problems are defined in Section 3 and compared with related results in Section 4. The undecidability of the E-unification problem for equational theories that are induced by monadic top-down tree transducers is shown in Section 5. In Section 6, the other side of the border of decidability is shown by verifying the decidability of the E-unification problem for equational theories that are induced by monadic tree homomorphisms. Finally, in Section 7 we give some conclusions and further

(un)decidability results that follow from the results of Sections 5 and 6.

2 Preliminaries

Let us recall some basic definitions and notions that we will use throughout the rest of this paper.

2.1 General Notations

The *set of non-negative integers* is denoted by $\mathbb{N}$. The *empty set* is denoted by $\emptyset$. For some $j \in \mathbb{N}$, $[j]$ denotes the set $\{1, \ldots, j\}$; therefore, $[0] = \emptyset$. For $i, j \in \mathbb{N}$ with $i \leq j$, $[i, j]$ denotes the set $\{i, i+1, \ldots, j\}$. For a finite set A, the *cardinality* of A will be denoted by $card(A)$. The *difference set* of two sets A and B, i.e., the set $\{x \mid x \in A \text{ and } x \notin B\}$, is denoted by $A - B$. For a set A, the *set of words over* A and the *set of nonempty words over* A are denoted by A^* and A^+, respectively. The empty word is denoted by ε. The length of a word $w \in A^*$ will be denoted by $|w|$. For a set A and an integer $n \geq 0$, the *set of words over* A *of length* n is denoted by A^n.

2.2 Ranked Alphabets, Variables, and Terms

A pair $(\Omega, rank_\Omega)$ is called a *ranked alphabet* if Ω is a finite set and $rank_\Omega : \Omega \to \mathbb{N}$ is a total function. For every symbol $f \in \Omega$, the natural number $rank_\Omega(f)$ is called the *rank of* f. For every $i \geq 0$, the set $\Omega^{(i)} \subseteq \Omega$ consists of all symbols in Ω that are of rank i. A ranked alphabet can be defined by either (1) an enumeration of the finitely many nonempty subsets $\Omega^{(i)}$, or (2) a set of symbols which are indexed by their unambiguous rank. In this paper we will mostly use the latter of these methods. In the following, let $(\Omega, rank_\Omega)$ be an arbitrary but fixed ranked alphabet.

For the rest of this paper we choose the *set of variables* to be the set $\mathcal{V} = \{x, z, x_1, z_1, x_2, z_2, \ldots\}$. In rewrite rules, we always use x, and in goals, we use always z.

Let S be an arbitrary set. The *set of terms over* Ω *indexed by* S, denoted by $T(\Omega, S)$, is the smallest set $T \subseteq (\Omega \cup S \cup \{(,), ,\})^*$, such that the following holds: $S \subseteq T$, and if $f \in \Omega^{(n)}$ with $n \geq 0$ and $t_1, \ldots, t_n \in T$, then $f(t_1, \ldots, t_n) \in T$.

For every term $t \in T(\Omega, \mathcal{V})$, the *set of occurrences of t*, denoted by $O(t)$, is a subset of $\mathbb{N}^*$, which is inductively defined as follows:

1. if $t = x$ with $x \in \mathcal{V}$, then $O(t) = \{\varepsilon\}$,
2. if $t = f$ with $f \in \Omega^{(0)}$, then $O(t) = \{\varepsilon\}$, and
3. if $t = f(t_1, \ldots, t_n)$ with $f \in \Omega^{(n)}$ and $n > 0$, and for all $i \in [n] : t_i \in T(\Omega, \mathcal{V})$, then $O(t) = \{\varepsilon\} \cup \bigcup_{i \in [n]} \{iu \mid u \in O(t_i)\}$.

For every term $t \in T(\Omega, \mathcal{V})$ and every occurrence u of t, the *subterm of t at occurrence u* is denoted by t/u. The *label of t at occurrence u* is denoted by $t[u]$. The *set of variables occurring in t* is denoted by $\mathcal{V}(t)$; that means $v \in \mathcal{V}(t)$, if $v \in \mathcal{V}$ and there is a $u \in O(t)$ such that $t/u = v$. The term t with the subterm at occurrence u replaced by the term s is denoted by $t[u \leftarrow s]$.

2.3 Substitutions

A mapping $\varphi : \mathcal{V} \to T(\Omega, \mathcal{V})$ is called a $(\mathcal{V}, \Omega)$-*substitution* if $\{v \mid v \in \mathcal{V} \text{ and } \varphi(v) \neq v\}$ is a finite set. The set of all $(\mathcal{V}, \Omega)$-substitutions is denoted by $Sub(\mathcal{V}, \Omega)$.

Every $(\mathcal{V}, \Omega)$-substitution φ is characterized in terms of those (finitely many) variables that change under φ. Let $v_1, \ldots, v_n \in \mathcal{V}$ be the complete list of variables, such that $\varphi(v_i) \neq v_i$. Then φ is unambiguously defined by $[v_1/\varphi(v_1), \ldots, v_n/\varphi(v_n)]$.

2.4 Term-Rewriting Systems and *E*-Unification

A *term-rewriting system*, denoted by $\mathcal{R}$, is a pair (Ω, R) where Ω is a ranked alphabet and R is a finite set of rules; every rule has the form $l \to r$ with $r, l \in T(\Omega, \mathcal{V})$ and $\mathcal{V}(r) \subseteq \mathcal{V}(l)$.

An equivalence relation $\sim$ over $T(\Omega, \mathcal{V})$ is called an Ω-*congruence over* $T(\Omega, \mathcal{V})$ if for every $f \in \Omega^{(n)}$ with $n > 0$ and for every $t_1, s_1, \ldots t_n, s_n \in T(\Omega, \mathcal{V})$ with $t_1 \sim s_1, \ldots t_n \sim s_n$: $f(t_1, \ldots, t_n) \sim f(s_1, \ldots, s_n)$.

Let $\mathcal{R} = (\Omega, R)$ be an arbitrary term-rewriting system. The set of equations induced by $\mathcal{R}$, denoted by $E_{\mathcal{R}}$, is the set $\{(l = r) \mid (l \to r) \in R\}$. The *E-equality*, denoted by $=_E$, is the finest Ω-congruence containing every pair $(\psi(t), \psi(s))$ where $(t = s) \in E$ and ψ is an arbitrary $(\mathcal{V}, \Omega)$-substitution.

Two terms $s, t \in T(\Omega, \mathcal{V})$ are *E-unifiable* if there exists a $(\mathcal{V}, \Omega)$-substitution φ, such that $\varphi(s) =_E \varphi(t)$.

3 *E*-Unification with Recursive Functions

In this section we describe the E-unification problems $\mathcal{U}_{mon\text{-}tdT}$ and $\mathcal{U}_{mon\text{-}Hom}$ for the class of sets of equations which are induced by monadic top-down tree transducers and monadic tree homomorphisms, respectively. In Sections 5 and 6, it will turn out that $\mathcal{U}_{mon\text{-}tdT}$ is undecidable, whereas $\mathcal{U}_{mon\text{-}Hom}$ is decidable. Thus, the underlying classes of sets of equations are close to the border that can be drawn between undecidability and decidability of the E-unification problem for recursive functions.

We recall the definition of top-down tree transducers [Rou70, Tha70]. Unlike the usual definition, we have omitted the use of separate input and output alphabets, and rather use one alphabet of *constructors*, Δ.

Definition 1 (Top-Down Tree Transducer)
A top-down tree transducer *is a tuple* (F, Δ, R) *such that:*

- F *is a nonempty ranked alphabet of* function symbols *which are all of rank 1,*
- Δ *is a nonempty ranked alphabet of* constructors*, and*
- R *is a finite set of* rewrite rules *which contains, for every* $f \in F$ *and every* $\sigma \in \Delta$ *of rank* $k \geq 0$*, exactly one rule of the form*

$$f(\sigma(x_1, \ldots, x_k)) \to \beta$$

where $\beta \in T(\Delta, \{g(x_i) \mid g \in F \text{ and } i \in [k]\})$.

We denote the class of sets of equations that are induced by top-down tree transducers by $\mathcal{E}_{tdT}$.

A top-down tree transducer is called *monadic* if all constructors are of rank 1 or 0. The corresponding class of sets of equations is denoted by $\mathcal{E}_{mon\text{-}tdT}$. Usually the derivation relation of top-down tree transducers is not defined for terms where function symbols are nested. However, since we want to consider the sets of equations that are induced by such transducers, we define the derivation relation term-rewriting system-like, and allow nesting of function symbols.

Definition 2 (Reduction Relation for Top-Down Tree Transducers) *Let $T = (F, \Delta, R)$ be a top-down tree transducer. The* reduction relation for *T, denoted by $\Rightarrow_T$, is a binary relation over $T(F \cup \Delta, \mathcal{V})$ such that for every $t, s \in T(F \cup \Delta, \mathcal{V})$: $t \Rightarrow_T s$ if and only if the following holds: there is an occurrence $u \in O(t)$, a substitution $\varphi \in Sub(\mathcal{V}, \Delta)$, and a rule $f(\sigma(x_1, \ldots, x_k)) \to \beta$ such that:*

1. *$f(\sigma(\varphi(x_1), \ldots, \varphi(x_k))) = t/u$ and*
2. *$s = t[u \leftarrow \hat{\varphi}(\beta)]$,*

where $\hat{\varphi}$ is the natural extension of φ from variables to terms in $T(F \cup \Delta, \mathcal{V})$.

Remark 1 *Every top-down tree transducer T is a convergent term-rewriting system [FHVV93], i.e., the reduction relation $\Rightarrow_T$ for T is terminating and confluent. Hence, every term t over F, Δ, and $\mathcal{V}$ has a unique normal form [New42] which is denoted by:*

$$nf(t, \Rightarrow_T).$$

In [Hue80] it is shown for arbitrary convergent term-rewriting systems (and hence, also for top-down tree transducers) T that two terms t and s over F, Δ, and $\mathcal{V}$ are E_T-equal, if and only if their normal forms are equal. More formally,

$$t =_{E_T} s, \text{ if and only if } nf(t, \Rightarrow_T) = nf(s, \Rightarrow_T).$$

We use this simple check from the previous remark to define $\mathcal{U}_{mon\text{-}tdT}$.

Definition 3 (*E*-Unification Problem for *mon-tdT*) *The* E-unification problem for monadic top-down tree transducers*, denoted by $\mathcal{U}_{mon\text{-}tdT}$, is the following problem:*

> Given: *An arbitrary monadic top-down tree transducer $T = (F, \Delta, R)$ and two terms t and s over F, Δ, and $\mathcal{V}$.*
> Question: *Are t and s E_T-unifiable?*

Using the previous remark, this reads as follows:

> *Does there exist a substitution $\varphi \in Sub(\mathcal{V}, \Delta)$ such that the following condition holds:*
>
> $$nf(\varphi(t), \Rightarrow_T) = nf(\varphi(s), \Rightarrow_T).$$

Note that we only consider a restricted kind of E-unification problem, namely, those problems with solutions $\varphi \in Sub(\mathcal{V}, \Delta)$. This kind of E-unifiability should be correctly called (E_T, Δ)-unifiability, as in [Ech90]. Nevertheless, since we are motivated from a functional logic programming point of view, we are only interested in solutions where a variable is bound to a constructor term and not to a function call. Hence, in this paper, it is justified to speak of E-unification problems instead of (E_T, Δ)-unification problems.

As mentioned before, $\mathcal{U}_{mon\text{-}tdT}$ will be shown to be undecidable. We also define the E-unification problem $\mathcal{U}^{T'}_{mon\text{-}tdT}$ for *mon-tdT* and a *particular* given monadic top-down tree transducer T'. Clearly, undecidability of $\mathcal{U}_{mon\text{-}tdT}$ is a consequence of the undecidability of $\mathcal{U}^{T'}_{mon\text{-}tdT}$.

Definition 4 (Particular *E*-Unification Problem for *mon-tdT*)
Let $T' = (F, \Delta, R)$ be a monadic top-down tree transducer. The E-unification problem for T', *denoted by $\mathcal{U}^{T'}_{mon\text{-}tdT}$, is the following problem:*

> Given: *Two terms t and s over F, Δ, and $\mathcal{V}$.*
> Question: *Are t and s $E_{T'}$-unifiable?*

In other words:

> *Does there exist a substitution $\varphi \in Sub(\mathcal{V}, \Delta)$ such that the following condition holds:*
>
> $$nf(\varphi(t), \Rightarrow_{T'}) = nf(\varphi(s), \Rightarrow_{T'}).$$

Let us now define a particular subclass of $\mathcal{E}_{tdT}$, namely, the class $\mathcal{E}_{Hom}$ of sets of equations that are induced by tree homomorphisms.

Definition 5 (Tree Homomorphism)
Let $T = (F, \Delta, R)$ be a top-down tree transducer. Then T is called a tree homomorphism *if $card(F) = 1$.*

We denote the class of sets of equations that are induced by tree homomorphisms by $\mathcal{E}_{Hom}$. As with top-down tree transducers, we call a tree homomorphism *monadic* if all constructors are of rank 1 or 0. The E-unification problem for monadic tree homomorphisms is denoted by $\mathcal{U}_{mon\text{-}Hom}$, and the corresponding class of sets of equations is denoted by $\mathcal{E}_{mon\text{-}Hom}$. In Section 6 it is shown that $\mathcal{U}_{mon\text{-}Hom}$ is decidable.

4 Comparison with Related Results

There are a number of results concerning the decidability of E-unification for particular classes of TRSs, e.g., Peano's axioms without multiplication, which can be expressed in Pressburger arithmetic [Pre27]; monoids [Mak77]; or particular, restricted, confluent string-rewriting systems [OND95].

More closely related to the classes of TRSs that we consider here are the following theories:

1. *Standard theories* [Nie96], which are an extension of *shallow theories* [CHJ94]. A shallow theory is induced by a TRS that is R restricted in the following way: in the right-hand sides of the rules of R, variables may occur only at depth at most 1.

2. Particular, restricted convergent (that is, terminating and confluent) TRSs, as considered in [Hul80, KN87, Mit94].

3. Particular, restricted, confluent, constructor-based, and linear TRSs [LR96].

4. Finite, length-reducing, and confluent Thue systems [NO90].

Let us now elaborate on these results and show that the induced classes of equational theories are incomparable with the class of TRSs for which we prove un-/decidability of E-unification in Section 5 and Section 6, respectively.

4.1 Related Decidability Results

4.1.1 Standard and Shallow Theories

Shallow variables are variables occurring in a term at depth at most 1. Shallow theories are induced by TRSs in which only shallow variables occur. In [CHJ94] it is shown that E-unification is decidable in shallow theories. In [Nie96] an extension of this result is presented, namely, decidability of E-unification for an even larger class is shown. In this class, nonshallow variables may occur in specific cases. However, there is still a restriction that in a rule $s \to t$, common variables of the terms s and t must be shallow.

Let us now show that both of these classes are incomparable with the class $\mathcal{E}_{mon\text{-}Hom}$ of equations induced by monadic tree homomorphisms. Standard

and shallow theories are defined for nonmonadic cases, too. Thus there are shallow theories that are not in $\mathcal{E}_{mon\text{-}Hom}$. On the other hand, monadic tree homomorphisms allow rules of the form $s \to t$, such that s and t have common nonshallow variables.

4.1.2 Results by Mitra

The reference that is most relevant and also most closely related to our results is [Mit94]. There, decidability results for three different classes of convergent TRSs are given: the E-unification problem is decidable if E is induced by:

1. A convergent TRS, in which the right-hand side of each rule is either a variable, a ground term, or a constructor-only term. This is an extension of [Hul80].

2. A convergent TRS, in which for every rule $l \to r$ the following holds: every subterm that occurs in r and the root of which is a function symbol, also occurs in l. This is an extension of [KN87].

3. A linear, convergent TRS, in which the right-hand side of each rule is either a variable, a constructor-only term, or a flat term. A term (which may contain constructors) is *flat* if it has a single defined function and all variables below this function symbol appear directly below it. Also, for each function symbol there may be at most one rule with a flat term as a right-hand side.

Let us now discuss the incomparability of each of these classes with the class *mon-Hom* of monadic tree homomorphisms. Each of the classes in 1–3 are defined for nonmonadic TRSs, also. Thus, in each class there are TRSs that are not in *mon-Hom*. On the other hand, consider a monadic tree homomorphism T that includes the rules $h(\gamma(x_1)) \to \sigma(h(x_1))$ and $h(\gamma'(x_1)) \to \sigma(h(x_1))$. Then it is easy to verify that T is in neither of the classes described under 1, 2, and 3. From this incomparability, the incomparability of the induced classes of equational theories follows.

4.1.3 Restricted, Confluent, Constructor-Based, and Linear TRSs

Limet and Réty show in [LR96] that the E-unification problem is decidable for equational theories induced by constructor-based, confluent TRSs with linear goals that respect the following three conditions:

1. rewrite rules are linear,
2. if a subterm r of some right-hand side unifies with some left-hand side l, then the most-general unifier σ does not modify the variables of l, and
3. right-hand sides contain no nested functions.

Since there is no restriction to monadic rules, there are theories induced by TRSs respecting Sections 4.1.1 through 4.1.3 which are not in $\mathcal{E}_{mon\text{-}Hom}$. On the other hand, the restriction of linearity of the goal of their class is quite strong. The problem $\mathcal{U}_{mon\text{-}Hom}$ is not restricted to linear goals. Thus the class of E-unification problems considered in [LR96] and the class of problems in $\mathcal{U}_{mon\text{-}Hom}$ are incomparable. Furthermore, as we will see in the proofs of Lemmas 1 and 2, the cases of nonlinear goals turn out to be the most interesting.

4.2 Related Undecidability Results

4.2.1 Finite, Length-Reducing, and Confluent Thue Systems

Let us now consider an undecidability result, and compare it with our undecidability result of Section 5. In Theorem 3.2 of [Ott86], the undecidability of the uniform E-unification problem for finite, length-reducing, and confluent Thue systems is proved. In [NO90] this result is improved, and for a given Thue system T and given words u and v it is shown that the question whether there exists a word w, such that $uw =_T vw$, is undecidable.

Thue systems are very close to monadic TRSs; therefore let us discuss how the results by Narendran and Otto are related to our result. (In the following, let us consider Thue systems as monadic TRSs.)

The Thue systems considered in [NO90] are not constructor-based, and a rule of the form $\gamma \to c$ is allowed. Thus, equational theories can be induced that are not in $\mathcal{E}_{mon\text{-}tdT}$. On the other hand, rules of a monadic tree homomorphism need not be length-reducing, and thus the class of equational theories induced by finite, length-reducing, and confluent Thue systems (seen as monadic TRSs) is incomparable with $\mathcal{E}_{mon\text{-}tdT}$.

Fixed:	$(u_2, \ldots, u_k), (v_2, \ldots, v_k) \in (\Sigma^+)^{k-1}$
Given:	Two words $u_1, v_1 \in \Sigma^+$.
Question:	Do there exist indices $i_2 \ldots, i_n \in [k]$, where $n \geq 2$, such that the following condition holds: $u_1 u_{i_2} \ldots u_{i_n} = v_1 v_{i_2} \ldots v_{i_n}$?

Figure 4: The problem *MPCP′*

5 Undecidability of $\mathcal{U}_{mon\text{-}tdT}$

In this section, we show the undecidability of $\mathcal{U}_{mon\text{-}tdT}$. In fact, we show two stronger undecidability results; namely:

1. we give a particular monadic top-down tree transducer T' such that given two terms s and t, the problem $\mathcal{U}^{T'}_{mon\text{-}tdT}$ is undecidable, and
2. we show that the problem remains undecidable if s and t are of a particular given form.

Both the undecidability of $\mathcal{U}_{mon\text{-}tdT}$ and the undecidability of $\mathcal{U}^{T'}_{mon\text{-}tdT}$ are consequences of result 2.

To prove result 2, we reduce an arbitrary instance, denoted by *MPCP′*, of the modified Post's correspondence problem (MPCP for short) [Pos46], which is undecidable, to it. *MPCP′* is defined in Figure 4, where Σ is an alphabet with $card(\Sigma) \geq 2$. The sequence of indices $i_2, \ldots, i_n$ is called a *solution* of *MPCP′*.

Let us now reduce the problem *MPCP′* to problem 2. For this purpose we construct the monadic top-down tree transducer T' and a mapping *red* which maps an arbitrary input (u_1, v_1) of *MPCP′* to an input $red((u_1, v_1))$ of problem 2.

The top-down tree transducer T' is constructed in such a way that $\bar{u}_1(f(z))$ and $\bar{v}_1(g(z))$ are $E_{T'}$-unifiable if and only if *MPCP′* has a solution for input (u_1, v_1), where $\bar{u}_1$ and $\bar{v}_1$ are corresponding monadic term representations of u_1 and v_1, respectively.

Theorem 1 *There is a monadic top-down tree transducer $T' = (F, \Delta, R)$, such that the following problem is undecidable:*

> Given: *Two monadic terms $u_1, v_1 \in T(\Delta', \{x\})$ where $\Delta' = \Delta - \Delta^{(0)}$.*

Question: *Are* $u_1[x/f(z)]$ *and* $v_1[x/g(z)]$ $E_{T'}$*-unifiable, where* $f, g \in F$*?*

Proof of Theorem 1 Let us now construct the monadic top-down tree transducer T'. We start from the fixed tuples of words (implied by $MPCP'$, as mentioned above) $(u_2, \ldots, u_k)$ and $(v_2, \ldots, v_k)$. The top-down tree transducer T' is constructed as follows:

$$T' = (F, \Delta, R), \text{where}$$

1. $F = \{f, g\}$.
2. Δ is the union of the three ranked alphabets *Indices*, *Letters*, and $\{e^{(0)}, c_f^{(0)}, c_g^{(0)}\}$. *Indices* is the ranked alphabet

   $$\{i^{(1)} \mid i \in [k]\}.$$

   and *Letters* is the ranked alphabet

   $\{a^{(1)} \mid a \in \Sigma$ and there exist a $\nu \in [k]$ and a $v, u \in \Sigma^*$, such that $u_\nu = vau$ or $v_\nu = vau$ holds.$\}$.

3. The set of rewrite rules, denoted by R, is the set:

$$\begin{array}{lllll}
 & \{f(i(x_1)) & \to \bar{u}_i(f(x_1)) & \mid i \in \mathit{Indices}\} & (1)\\
\cup & \{f(e) & \to e\} & & (2)\\
\cup & \{g(i(x_1)) & \to \bar{v}_i(g(x_1)) & \mid i \in \mathit{Indices}\} & (3)\\
\cup & \{g(e) & \to e\} & & (4)\\
\cup & \{f(a(x_1)) & \to c_f & \mid a \in \mathit{Letters}\} & (5)\\
\cup & \{g(a(x_1)) & \to c_g & \mid a \in \mathit{Letters}\} & (6)\\
\cup & \{f(c_f) & \to c_f\} & & (7)\\
\cup & \{f(c_g) & \to c_f\} & & (8)\\
\cup & \{g(c_f) & \to c_g\} & & (9)\\
\cup & \{g(c_g) & \to c_g\} & & (10)
\end{array}$$

Rules of the forms 5–10 are only of technical nature. Furthermore, the part of a monadic tree that corresponds to u_i or v_i is denoted by $\bar{u}_i$ or $\bar{v}_i$, respectively.

From the construction of T', it obviously follows for every $i_2, \ldots, i_n \in [k]$:

$$nf(f(i_2 \ldots i_n e), \Rightarrow_{T'}) = \bar{u}_{i_2} \ldots \bar{u}_{i_n} e$$

and

$$nf(g(i_2 \ldots i_n e), \Rightarrow_{T'}) = \bar{v}_{i_2} \ldots \bar{v}_{i_n} e.$$

That means the sequence of indices $i_2, \ldots, i_n$ is a solution of *MPCP'* for the input (u_1, v_1), if and only if the substitution $[z/i_2 \ldots i_n e]$ is an $E_{T'}$-unifier of $\bar{u}_1(f(z))$ and $\bar{v}_1(g(z))$. Obviously, the mapping *red* is simply given by $red((u_1, v_1)) = (\bar{u}_1(f(z)), \bar{v}_1(g(z)))$.

Since by construction of T' only $E_{T'}$-unifiers of $f(z)$ and $g(z)$ (and therefore of $\bar{u}_1(f(z))$ and $\bar{v}_1(g(z))$) do exist which have this form, we have reduced *MPCP'* to our problem. Thus, from the undecidability of *MPCP'*, the undecidability of our problem follows.

Proof of Theorem 1 □

From Theorem 1 we can infer the undecidability of $\mathcal{U}^{T'}_{mon\text{-}tdT}$, where T' is the monadic top-down tree transducer defined in the proof of Theorem 1, and of the uniform problem $\mathcal{U}_{mon\text{-}tdT}$.

Corollary 1 *Let T' be the monadic top-down tree transducer defined in the proof of Theorem 1. Then, $\mathcal{U}^{T'}_{mon\text{-}tdT}$ is undecidable.*

Corollary 2 *$\mathcal{U}_{mon\text{-}tdT}$ is undecidable.*

By proving the undecidability of $\mathcal{U}_{mon\text{-}tdT}$, we have considered one side of the border of decidability of E-unification for recursive functions. In the following section, we consider the other side. Furthermore, in Section 7, we mention some undecidability results that follow immediately from Theorem 1.

6 Decidability of $\mathcal{U}_{mon\text{-}Hom}$

In this section we show that the E-unification problem for the class of sets of equations induced by monadic tree homomorphisms is decidable. Let us first state two lemmas for specific cases of E-unification problems for this class of sets of equations which will be needed for the proof of the decidability of $\mathcal{U}_{mon\text{-}Hom}$.

Since we are dealing with monadic constructors, let us switch to word notation, i.e., let $\gamma_1(\dots\gamma_n(\alpha)\dots)$ be denoted as $\gamma_1\dots\gamma_n\alpha$. Let Δ^T denote all word representations of terms in $T(\Delta)$, that is, the regular language $\{wa \mid w \in \Delta_1^*, a \in \Delta_0\}$, where Δ_1 is an alphabet of the symbols in $\Delta^{(1)}$ and Δ_0 is an alphabet of the symbols in $\Delta^{(0)}$. Furthermore, we say a word $w \in \Delta_1^*$ can be *consumed*, if for every letter γ that occurs in w, there is a rewrite rule $h(\gamma x) \to h(x)$ in R. In particular, if $w = \varepsilon$, then no such rule is necessary in R.

Lemma 1 *Let $T = (\{h\}, \Delta, R)$ be a monadic tree homomorphism, and let $v \in \Delta_1^*$. Then, $\mathcal{U}_{mon\text{-}Hom}$ is decidable for goals of the form $z_1 =_{E_T} vh(z_1)$.*

Proof of Lemma 1 Let $T = (\{h\}, \Delta, R)$ be a monadic tree homomorphism. Consider the procedure $solve_1$, shown in Figure 5. This procedure takes as input a string of symbols in Δ_1 and a set of symbols in Δ_1.

Claim: The call $solve_1(v, \emptyset)$ returns an E_T-unifier for the goal $z_1 =_{E_T} vh(z_1)$ if and only if there exists one; otherwise it returns a string of the form $w\,false$, with $w \in \Delta_1^*$.

Let us first prove termination for $solve_1$, and then show that it indeed returns an E_T-unifier, if there is one.

Cases 1 and 2 of $solve_1$ terminate immediately. Only case 3 calls $solve_1$ recursively. In case 3, the string v' is nonempty; thus $S' = \{\gamma \mid \gamma \text{ occurs in } v'\}$ is a nonempty set. If S' is not a subset of S, then $solve_1$ is called recursively with the set $S \cup S'$ as a second argument. Then, in particular, it holds that $card(S \cup S') > card(S)$, $S, S' \in \mathcal{P}(\Delta_1)$, thus $solve_1$ will be called at most $card(\Delta_1)$ times, which is a finite number, because Δ is finite by Definition 1.

Let us now show why $solve_1$ returns an E_T-unifier if and only if there is one. Consider $\xi = nf(h(vz), \Rightarrow_T)$ for $v \in \Delta_1^*$. Obviously, ξ will either be a terminal word in Δ^T (case 1) or $h(z)$ will occur in ξ. For the latter case, we distinguish between $v'h(z)$ for $|v'| = 0$ (case 2) and $|v'| > 0$ (case 3).

In case 1, $solve_1$ returns the string vw as the solution. If we check this by substituting it for z_1 in the goal, we obtain $vw = vh(vw)$, and since $nf(h(vz), \Rightarrow_T) = w$, it follows that $vw = vw$; thus vw is in fact an E_T-unifier.

In case 2, $solve_1$ returns $vw\alpha$ if there is a rule $h(\alpha) \to w\alpha$, and w can be consumed. By substituting this solution for z_1 in the goal, we obtain $vw\alpha =_{E_T} vh(vw\alpha) =_{E_T} vh(w\alpha) =_{E_T} vh(\alpha) =_{E_T} vw\alpha$. Thus

$solve_1(v : \Delta_1^*, S : \mathcal{P}(\Delta_1)) : \bar{\Delta}^T$ (where $\bar{\Delta} = \Delta \cup \{false^{(0)}\}$)
case $nf(h(vz), \Rightarrow_T)$
(1) $= w$, with $w \in \Delta^T$: **break**(vw);
(2) $= h(z)$: **if** there is a rule $h(\alpha) \to w\alpha$ in R with $w \in \Delta_1^*$, $\alpha \in \Delta_0$
and w can be consumed **then break**$(vw\alpha)$
elseif there is a rule $h(\gamma x) \to w\gamma w'$ in R with $\gamma \in \Delta_1$, $w \in \Delta_1^*$,
$w' \in \Delta^T$ **and** w can be consumed **then break** $(vw\gamma w')$
else break$(false)$;
(3) $= v'h(z)$, with $v' \in \Delta_1^+$: let $S' = \{\gamma \mid \gamma$ occurs in $v'\}$;
if $S' \subseteq S$ **then break**$(false)$ **else break**$(v \cdot solve_1(v', S \cup S'))$;
end (of case)

Figure 5: The decision procedure $solve_1$

$vw\alpha$ is indeed an E_T-unifier. If there is a rule $h(\gamma x) \to w\gamma w'$ and w can be consumed, then $vw\gamma w'$ is returned. This is an E_T-unifier, because $vw\gamma w' =_{E_T} vh(vw\gamma w') =_{E_T} vh(w\gamma w') =_{E_T} vh(\gamma w')$, which is equal to $vw\gamma w'$ by the existence of the rule $h(\gamma x) \to w\gamma w'$. If there is no rule of the form $h(\alpha) \to w\alpha$ or $h(\gamma x) \to w\gamma w'$, then there does not exist an E_T-unifier, because there is no rule to make the rightmost ends of the strings z and $h(z)$ equal; therefore these two strings cannot be E_T-equal for any z.

In case 3, $(v \cdot solve_1(v', S \cup S'))$ is returned (the dot denotes concatenation here).

At this point it is only certain that if there is a solution, it is of the form vy with $y \in \Delta^T$. Let us verify this by substituting vy in the goal: $vy =_{E_T} vh(vy)$ and thus $y =_{E_T} v'h(y)$, which is a goal of the original form. However, if v' only contains symbols that we have already checked in terms of computing the normal form of strings of the form $h(vz)$, then there cannot be any solution, and $solve_1$ returns *false*.

Proof of Lemma 1 □

Let us now give an example of a monadic tree homomorphism and a goal of the form $z_1 = vh(z_1)$ to illustrate the decision procedure $solve_1$.

Example 1 Let $T = (\{h\}, \Delta, R)$ be a monadic tree homomorphism with $\Delta = \{a^{(1)}, b^{(1)}, c^{(1)}, d^{(1)}, e^{(1)}, \alpha^{(0)}\}$, and let R consist of the following rules:

$$\begin{array}{lcll} h(a(x_1)) & \to & d\,h(x_1) & (1)\\ h(b(x_1)) & \to & c\,h(x_1) & (2)\\ h(c(x_1)) & \to & ab\alpha & (3)\\ h(d(x_1)) & \to & h(x_1) & (4)\\ h(e(x_1)) & \to & e\,h(x_1) & (5)\\ h(\alpha) & \to & dd\alpha & (6) \end{array}$$

Let us now consider three different goals and show the calls to $solve_1$, the involved normal form, and case.

1. Goal: $z_1 =_{E_T} a\,h(z_1)$. $solve_1(a, \emptyset)$ computes $nf(h(az), \Rightarrow_T) = d\,h(z)$, and then by case 3 returns $a \cdot solve_1(d, \{d\})$.
   $solve_1(d, \{d\})$ computes $nf(h(dz), \Rightarrow_T) = h(z)$. Rule 6 is of the form $h(\alpha) \to w\alpha$, and $w = dd$ can be consumed by rule 4; thus by the (first) if-part of case 2 this call of $solve_1$ returns $vw\alpha = ddd\alpha$ and the initial call $solve_1(a, \emptyset)$ returns the string $adddd\alpha$.

   Let us verify that this is indeed an E_T-unifier by substituting it for z_1 in the goal:

   $$adddd\alpha =_{E_T} a\,h(adddd\alpha) \Rightarrow_T ad\,h(ddd\alpha) \Rightarrow_T^3 ad\,h(\alpha) \Rightarrow_T adddd\alpha.$$

2. Goal: $z_1 =_{E_T} ab\,h(z_1)$. $solve_1(ab, \emptyset)$ computes $nf(h(abz), \Rightarrow_T) = dc\,h(z)$, and then by case 3 returns $ab \cdot solve_1(dc, \{c, d\})$.
   $solve_1(dc, \{c, d\})$ computes $nf(h(dcz), \Rightarrow_T) = ab\alpha$, and by case 1 this returns $vw = dcab\alpha$. Thus the initial call $solve_1(ab, \emptyset)$ returns the string $abdcab\alpha$.

   Let us verify that this is indeed an E_T-unifier by substituting it for z_1 in the goal:

   $$\begin{aligned} abdcab\alpha =_{E_T} ab\,h(abdcab\alpha) \Rightarrow_T abd\,h(bdcab\alpha) \Rightarrow_T abdc\,h(dcab\alpha)\\ \Rightarrow_T abdc\,h(cab\alpha) \Rightarrow_T abdcab\alpha \end{aligned}$$

3. Goal: $z_1 =_{E_T} e\,h(z_1)$. $solve_1(e, \emptyset)$ computes $nf(h(ez), \Rightarrow_T) = e\,h(z)$, and then by case 3 returns $e \cdot solve_1(e, \{e\})$. Again $nf(h(ez), \Rightarrow_T) = e\,h(z)$ is computed. However, now by case 3 it is checked if $S' = \{e\} \subseteq \{e\} = S$, which is true; therefore *false* is returned from this second call to $solve_1$. Thus the initial call of $solve_1(e, \emptyset)$ returns $e\,false$.

To demonstrate the elseif part of case 2, consider the monadic tree homomorphism $T' = (\{h\}, \Delta, R')$, where R' consists of the rules 1–4 and rule 5′, which is defined as follows:

$$h(e(x_1)) \quad \rightarrow \quad deacdc\alpha \quad (5')$$

Now consider the goal $z_1 = d\,h(z_1)$. Then $solve_1(d, \emptyset)$ computes

$$nf(h(dz), \Rightarrow_{T'}) = h(z).$$

By the elseif part of case 2, the string $ddeacdc\alpha$ is returned. As the reader may confirm, $d\,h(ddeacdc\alpha) \Rightarrow^3_{T'} ddeacdc\alpha$.

Lemma 2 *Let $T = (\{h\}, \Delta, R)$ be a monadic tree homomorphism, and let $v \in \Delta_1^*$. Then, $\mathcal{U}_{mon\text{-}Hom}$ is decidable for goals of the form $h(z) =_{E_T} vz$.*

Proof of Lemma 2 For showing the correctness of the lemma, we define the decision procedure $solve_2$ in Figure 6. This procedure takes as input a string of symbols in Δ_1 and a subset of Δ_1^*, which is used for checking whether all possible cases are checked in case 5.

Claim: The call $solve_2(v, \emptyset)$ returns a string ST of symbols that are of the form $w \cdot false$ or $w\alpha$, where $w \in \Delta_1^*$ and $\alpha \in \Delta_0$. The terms $h(z)$ and vz are E_T-unifiable, if and only if ST contains at least one term $t \in \Delta^T$. In this case, the substitution $[z/t]$ is an E_T-unifier of $h(z)$ and vz.

Although cases 4 and 5 in $solve_2$ seem very similar, they have to be separated to ensure the termination of the procedure.

Let us first prove termination of $solve_2$.

In cases 0 and 3, $solve_1$ is called, which terminates by Lemma 1.

Cases 1 and 2 terminate immediately.

For proving the termination in cases 4 and 5, we define the complexity measure $\mathcal{M}(v, S)$ and show that it will be reduced by every recursive call of $solve_2$.

$\mathcal{M}(v, S)$ is defined as

$$|v|^2 \cdot card(\Delta)^2 - card(S).$$

Note that by definition of the second argument of $solve_2$, every element in S is of length $|v|$. Thus,

$$card(S) \leq |v| \cdot card(\Delta). \qquad (*)$$

$solve_2(v : \Delta_1^*, S : \mathcal{P}(\Delta_1^{|v|})) : (\bar{\Delta}^T)^*$ (where $\bar{\Delta} = \Delta \cup \{false^{(0)}\}$)

let $\alpha \in \Delta_0$; $\beta, \beta' \in \Delta_1$; $w \in \Delta_1^*$ **in**
if $v = \varepsilon$ **then**
    (0) **break**$(solve_1(\varepsilon, \emptyset))$;
if $v \neq \varepsilon$ **then**
    (1) **if** there is a rule $h(\alpha) \to v\alpha \in R$ **then break**(α);
    (2) **if** there is a rule $h(\beta x) \to v\beta\alpha \in R$ **then break**$(\beta\alpha)$;
    (3) **for** every rule $h(\beta x) \to v\beta w h(x) \in R$ **do**
        break$(\beta \cdot solve_1(w, \emptyset))$;
    (4) **for** every rule $h(\beta x) \to w h(x) \in R$, where there is a
            $u \in \Delta_1^*$, such that $wu = v$ and $|w| > 1$ **do**
        break$(\beta \cdot solve_2(u\beta, \{u\beta\}))$;
    (5) **for** every rule $h(\beta x) \to \beta' h(x) \in R$, where there is a
            $u \in \Delta_1^*$, such that $\beta' u = v$ **do**
        if $u\beta \notin S$ **then break**$(\beta \cdot solve_2(u\beta, S \cup \{u\beta\}))$;
break$(false)$.

Figure 6: The decision procedure $solve_2$

Case 4: We have to show $\mathcal{M}(u\beta, \{u\beta\}) < \mathcal{M}(v, S)$.
We have $|u| < |v| - 1$; that means $|u\beta| < |v|$.
We get:

$$\mathcal{M}(u\beta, \{u\beta\}) = |u\beta|^2 \cdot card(\Delta)^2 - 1 < \mathcal{M}(v, S)$$

because of $(*)$ and the fact that for every $m < n : m^2 \leq n^2 - n$.

Case 5: We have to show $\mathcal{M}(u\beta, S \cup \{u\beta\}) < \mathcal{M}(v, S)$.
We have $|u| = |v| - 1$; that means $|u\beta| = |v|$.
We get:

$$\mathcal{M}(u\beta, S \cup \{u\beta\}) = \mathcal{M}(v, S) - 1 < \mathcal{M}(v, S)$$

which completes the proof of the termination of $solve_2$.

Let us now show why $solve_2$ produces an E_T-unifier, if and only if one exists.

Case 0 is solved by $solve_1$.

If there exists a rule $h(\alpha) \to v\alpha \in R$ in case 1, then $[z/\alpha]$ is an E_T-unifier, because $h(\alpha) =_{E_T} v\alpha$.

If there exists a rule $h(\beta x) \to v\beta\alpha \in R$ in case 2, then $[z/\beta\alpha]$ is an E_T-unifier, because $h(\beta\alpha) =_{E_T} v\beta\alpha$.

In case 3, v is a prefix of a rule's right-hand side. If there exists a rule $h(\beta x) \to v\beta w h(x) \in R$, then a possible solution starts with β, followed by a solution of the goal $wh(z') =_{E_T} z'$. The new goal has a form as the goal in Lemma 1. Hence, a solution is computed by $solve_1$, if and only if one exists.

In case 4, the prefix of the rule's right-hand side is a prefix of v. A possible solution starts with β followed by a solution of the goal $h(z') =_{E_T} u\beta z'$. The new goal is of the same form as the previous one. But, as shown in the termination proof, it is of reduced complexity. Hence, it can be solved by a recursive call of $solve_2$, if and only if a solution exists.

In case 5, the prefix of the rule's right-hand side is a symbol that is also the leftmost symbol in v. A possible solution starts with β, followed by a solution of the goal $h(z') =_{E_T} u\beta z'$. The new goal is of the same form as the previous one, where $u\beta$ has the same length as v. Hence, it can be solved by a recursive call of $solve_2$, if and only if a solution exists. As shown by the termination proof, the recursive call does not lead to an infinite loop. This is guaranteed by the second argument of $solve_2$, which includes all prefixes that have been checked so far.

Since $solve_2$ compares v with the right-hand sides of the rewrite rules in R, and only those rewrite rules the right-hand sides of which start with something that is comparable with a prefix of v are considered, a solution is computed, if and only if one exists.

Proof of Lemma 2 □

Let us now give an example of a monadic tree homomorphism and a goal of the form $vz =_{E_T} h(z)$ to illustrate the decision procedure $solve_2$.

Example 2 Let $T = (\{h\}, \Delta, R)$ be a monadic tree homomorphism with $\Delta = \{a^{(1)}, b^{(1)}, c^{(1)}, d^{(1)}, e^{(1)}, \alpha^{(0)}\}$, and let R consist of the following rules:

$$\begin{array}{lcll} h(a(x_1)) & \to & c\,h(x_1) & (1) \\ h(b(x_1)) & \to & ab\,h(x_1) & (2) \\ h(c(x_1)) & \to & bc\alpha & (3) \\ h(d(x_1)) & \to & e\,h(x_1) & (4) \\ h(e(x_1)) & \to & d\,h(x_1) & (5) \\ h(\alpha) & \to & ba\alpha & (6) \end{array}$$

Let us now consider four different goals and show the calls to $solve_2$, the involved normal form, and case.

1. Goal: $baz =_{E_T} h(z)$. $solve_2(ba, \emptyset)$ computes as solution α in case 1. Let us verify that this is indeed an E_T-unifier by substituting it for z in the goal: $ba\alpha =_{E_T} h(\alpha) \Rightarrow_T ba\alpha$.

2. Goal: $bz =_{E_T} h(z)$. $solve_2(b, \emptyset)$ computes as solution $c\alpha$ in case 2. Let us verify that this is indeed an E_T-unifier by substituting it for z in the goal: $bc\alpha =_{E_T} h(c\alpha) \Rightarrow_T bc\alpha$.

3. Goal: $abcz =_{E_T} h(z)$. $solve_2(abc, \emptyset)$ computes in case 4 to $b \cdot solve_2(cb, \{cb\})$, which computes in case 5 to $ba \cdot solve_2(ba, \{cb, ba\})$, which computes as solution $ba\alpha$ in case 1. Let us verify that this is indeed an E_T-unifier by substituting it for z in the goal: $abcba\alpha =_{E_T} h(ba\alpha) \Rightarrow_T ab\,h(a\alpha) \Rightarrow_T abc\,h(\alpha) \Rightarrow_T abcba\alpha$.

4. Goal: $dez =_{E_T} h(z)$. $solve_2(de, \emptyset)$ computes $e \cdot solve_2(ee, \{ee\})$ by case 5, which computes $ed \cdot solve_2(ed, \{ee, ed\})$ by case 5, which computes $edd \cdot solve_2(dd, \{ee, ed,\ dd\})$ by case 5, which computes $edde \cdot solve_2(de, \{ee, ed, dd, de\})$ by case 5, which computes by case 5 $edde \cdot false$, because ee is still in S.

Note that the procedure $solve_2$ may compute more than one solution, because every possible construction of a solution is considered.

We now quote a result for monadic top-down tree transducers that we will need for certain cases in the proof of the decidability of $\mathcal{U}_{mon\text{-}Hom}$. Top-down tree transducers for monadic constructor alphabets coincide with generalized sequential machines (GSMs) [Gin62] (cf. Section 4 in Chapter 1 of [Sal73] for a suitable definition of GSMs). We can therefore make use of the following well-known lemma.

Lemma 3 (Theorem 3.1 of [GR63]) *Regular languages are closed under GSM mappings.*

Now, we are ready to prove the following theorem.

Theorem 2 $\mathcal{U}_{mon\text{-}Hom}$ *is decidable.*

Proof of Theorem 2 Let $T = (\{h\}, \Delta, R)$ be a monadic tree homomorphism. Then, the E-unification problem can occur in the following four forms (modulo commutativity), where we have only to consider terms in normal form:

1. $w =_{E_T} w'$
2. $vh^i(z_1) =_{E_T} w'$
3. $vh^i(z_1) =_{E_T} v'h^j(z_1)$ or
4. $vh^i(z_1) =_{E_T} v'h^j(z_2)$

for $i, j \geq 0$, $w, w' \in \Delta^T$, and $v, v' \in \Delta_1^*$.

Case 1: Trivially decidable.

Case 2: Applying Lemma 3 and the fact that the language Δ^T is regular, we get that the language $L_1 = \{nf(vh^i(u), \Rightarrow_T) \mid u \in \Delta^T\}$ is regular. Therefore, this case is decidable, because $L_1 \cap \{w'\}$ is regular and the emptiness problem for regular languages is decidable (see, e.g., [HU79]).

Case 3: We can reduce the goal $vh^i(z_1) =_{E_T} v'h^j(z_1)$ in the following way: without loss of generality, $|v| \leq |v'|$. Then either $v' = v\hat{v}$ for some $\hat{v} \in \Delta_1^*$, or there is no E_T-unifier. Thus the goal reduces to $h^i(z_1) =_{E_T} \hat{v}h^j(z_1)$.

This goal can again be reduced in the following ways:

1. If $i < j$, then we can reduce the goal to the form $z_1 =_{E_T} \hat{v}h^{j-i}(z_1)$. Since tree homomorphisms are closed under composition [Eng75], there is a tree homomorphism h' such that $nf(h'(w), \Rightarrow_T) = nf(h^{j-i}(w), \Rightarrow_T)$ for all $w \in \Delta^T$, and therefore this case reduces to $z_1 =_{E_T} \hat{v}h'(z_1)$, which is decidable by Lemma 1.

2. If $i = j$, then there is a solution if and only if $|\hat{v}| = 0$.

3. If $i > j$, then we can reduce the goal to the form $h^{i-j}(z_1) =_{E_T} \hat{v}z_1$, which is (again by the closure of tree homomorphisms) reducible to $h'(z_1) =_{E_T} \hat{v}z_1$ and therefore decidable by Lemma 2.

Case 4: Let

$$L_1 = \{nf(vh^i(u), \Rightarrow_T) \mid u \in \Delta^T\}$$

and

$$L_2 = \{nf(v'h^j(u), \Rightarrow_T) \mid u \in \Delta^T\}.$$

Since, as in case 2, L_1 and L_2 are regular, $L_1 \cap L_2 = \emptyset$ is decidable, and thus the E_T-unification problem is decidable for this case.

Proof of Theorem 2 □

7 Conclusions and Immediate Results

In this paper we have shown the undecidability of the E-unification problem $\mathcal{U}_{mon\text{-}tdT}$ and the decidability of the E-unification problem $\mathcal{U}_{mon\text{-}Hom}$. The two underlying classes of equational theories for these E-unification problems are induced by very similar types of restricted term-rewriting systems called monadic top-down tree transducers and monadic tree homomorphisms, respectively. Thus, we have drawn a quite strict border between the undecidability and the decidability of E-unification for recursive functions.

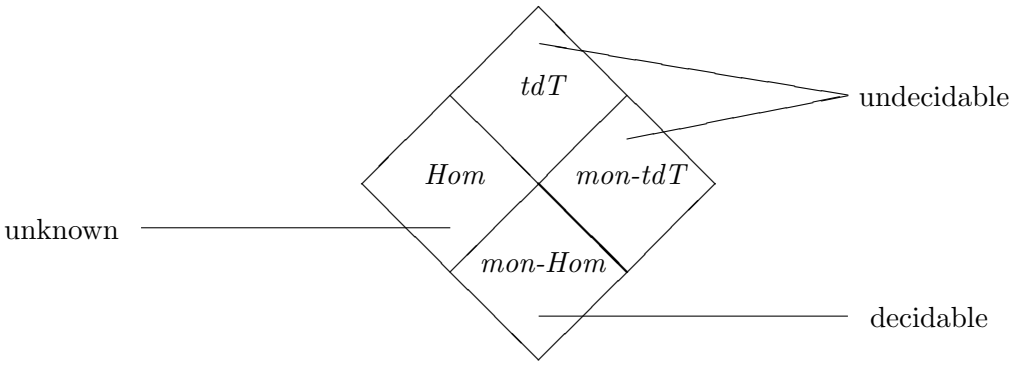

Figure 7: Results of this paper

The following undecidability results follow immediately from the verified results.

Undecidability for Top-Down Tree Transducers: The class $\mathcal{E}_{mon\text{-}tdT}$ is a subclass of $\mathcal{E}_{tdT}$, but since $\mathcal{U}_{mon\text{-}tdT}$ was shown to be undecidable, the undecidability result applies to $\mathcal{E}_{tdT}$ as well.

Undecidability for Primitive-Recursive Tree Functions: Since the class of primitive-recursive tree functions is exactly the class of functions which are induced by modular tree transducers [EV91] and every top-down tree transducer is a particular modular tree transducer [EV91], the E-unification problem for primitive-recursive tree functions is undecidable.

Further Research Topics: Our results are depicted in Figure 7. There, we have depicted the different classes of tree transducers that induce equational theories for which the E-unification problem is undecidable or decidable, respectively. In future work it would be interesting to consider the class $\mathcal{E}_{Hom}$ of sets of equations that are induced by tree homomorphisms, i.e., to drop the restriction of monadic terms. Also, it might be interesting to consider the class $\mathcal{E}_{tdT}$ and to restrict the input terms in certain ways, such that the corresponding E-unification problem becomes decidable. It seems like there are only a few restrictions needed to achieve this. Also, combinations between the results of [Mit94] and our results could be considered, such as Mitra's flat-term restriction for linear, convergent TRSs (as discussed in the introduction) plus the additional possibility to call functions that are defined by restricted tree homomorphisms.

In [Sie78], the notion of *unification type* was introduced. It gives infor-

mation about the size and existence of a minimal complete set of E-unifiers for two terms. It is interesting to note that for E-unification problems that are induced by particular classes of TRSs, it is in general not clear of which unification type they are. However, maybe for small classes of TRSs such as monadic top-down tree transductions it could be possible to give more information about the unification type. This seems to be an interesting, open problem that could be the subject of further research. The only thing that seems to be clear for the E-unification problems which we considered here with respect to the unification type is the following. Equational theories induced by monadic tree homomorphisms are in general *not* of unification type 1 (unitary). Consider, e.g., the monadic tree homomorphism $T = (\{h\}, \Delta, R)$ with $\Delta = \{a^{(0)}, b^{(0)}, c^{(0)}\}$ and $R = \{h(a) \to a, h(b) \to b, h(c) \to a\}$ and the goal $h(z) =_{E_T} z$. Then a and b are E-unifiers, i.e., there is no *single* most-general unifier, and thus the unification type of E_T is not equal to 1 (in fact, it is equal to 2 for this example).

Acknowledgments

We are thankful to Friedrich Otto for pointing out to us the possibility to improve our previous undecidability result by using the modified version of Post's correspondence problem. We also thank the referees for many helpful comments and suggestions.

References

[BS94] F. Baader and J. H. Siekmann. Unification theory. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 2, pages 41–125, Oxford, 1994. Oxford University Press.

[CHJ94] H. Comon, M. Haberstrau, and J.-P. Jouannaud. Syntacticness, cycle-syntacticness, and shallow theories. *Information and Computation*, 111:154–191, 1994.

[DJ90] N. Dershowitz and J. P. Jouannaud. Rewrite systems. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 6, pages 243–320, Amsterdam, 1990. Elsevier.

[Ech90] R. Echahed. On completeness of narrowing strategies. *Theoretical Computer Science*, 72:133–146, 1990.

[Eng75] J. Engelfriet. Bottom-up and top-down tree transformations—a comparison. *Mathematical Systems Theory*, 9(3):198–231, 1975.

[EV91] J. Engelfriet and H. Vogler. Modular tree transducers. *Theoretical Computer Science*, 78:267–304, 1991.

[FHVV93] Z. Fülöp, F. Herrmann, S. Vágvölgyi, and H. Vogler. Tree transducers with external functions. *Theoretical Computer Science*, 108:185–236, 1993.

[FM96] H. Faßbender and S. Maneth. A strict border for the decidability of E-unification for recursive functions. In M. Hanus and M. Rodríguez-Artalejo, editors, *5th International Conference on Algebraic and Logic Programming, ALP'96*, volume 1139 of *Lecture Notes in Computer Science*, pages 194–208, Berlin, September 1996. Springer-Verlag.

[Gin62] S. Ginsburg. Examples of abstract machines. *IEEE Transactions on Electronic Computers*, 11(2):132–135, 1962.

[GR63] S. Ginsburg and G. F. Rose. Operations which preserve definability in languages. *Journal of the ACM*, 10(2):175–195, 1963.

[HU79] J. W. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Reading, MA, 1979. Addison-Wesley.

[Hue80] G. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27:797–821, 1980.

[Hul80] J. M. Hullot. Canonical forms and unification. In *Proceedings of the 5th Conference on Automated Deduction*, volume 87 of *Lecture Notes in Computer Science*, pages 318–334, Berlin, 1980. Springer-Verlag.

[KN87] D. Kapur and P. Narendran. Matching, unification and complexity. *ACM SIGSAM Bulletin*, 21(4):6–9, 1987.

[LR96] S. Limet and P. Réty. E-unification by means of tree tuple synchronized grammars. *Discrete Mathematics and Theoretical Computer Science*, 1:69–98, 1996.

[Mak77] G. S. Makanin. The problem of solvability of equations in a free semigroup. *Mathematical USSR Sbornik*, 32(2):129–198, 1977.

[Mat70] Y. Matijasevič. Diophantine representation of recursively enumerable predicates. In *Actes du Congrès International des Mathématiciens*, volume 1, pages 235–238, Nice (France), 1970.

[Mit94] S. Mitra. Semantic unification for convergent systems. Technical Report CS-R-94-1855, University of Illinois at Urbana-Champaign, 1994.

[MM82] A. Martelli and U. Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages Systems*, 4:258–282, 1982.

[MR92] J. J. Moreno-Navarro and M. Rodriguez-Artalejo. Logic-programming with functions and predicates: The language BABEL. *Journal of Logic Programming*, 12:191–223, 1992.

[New42] M. H. A. Newman. On theories with a combinatorial defnition of "equivalence." *Annals of Mathematics*, 43(2):223–243, 1942.

[Nie96] R. Nieuwenhuis. Basic paramodulation and decidable theories (extended abstract). In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science*, pages 473–482, New Brunswick, New Jersey, 1996. IEEE Computer Society Press.

[NO90] P. Narendran and F. Otto. Some results on equational unification. In M. E. Stickel, editor, *Proceedings of the 10th International Conference on Automated Deduction, Kaiserslautern, Germany*, volume 449 of *Lecture Notes in Computer Science*, pages 276–291, Berlin, July 1990. Springer-Verlag.

[OND95] F. Otto, P. Narendran, and D. J. Dougherty. Some independence results for equational unification. In J. Hsiang, editor, *Proceedings of the 6th International Conference on Rewriting Techniques and*

Applications, volume 914 of *Lecture Notes in Computer Science*, pages 367–381, Berlin, 1995. Springer-Verlag.

[Ott86] F. Otto. On two problems related to cancellativity. *Semigroup Forum*, 33:331–356, 1986.

[Pos46] E. Post. A variant of a recursively unsolvable problem. *Bulletin of the AMS*, 52:264–268, 1946.

[Pre27] M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen. In *Comptes Rendus Premier Congrès des Mathématicienes des Pays Slaves, Warsaw*, pages 92–101, 1927.

[Rob65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. Assoc. Computer Mach.*, 20:23–41, 1965.

[Rou70] W. C. Rounds. Mappings and grammars on trees. *Mathematical Systems Theory*, 4:257–287, 1970.

[Sal73] A. Salomaa. *Formal Languages*. New York, 1973. Academic Press.

[Sie78] J. H. Siekmann. Unification and matching problems. Memo, 1978.

[Sti75] M. E. Stickel. A complete unification algorithm for associative-commutative functions. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence (IJCAI), Tbilisi*, pages 71–76, 1975.

[Tha70] J. W. Thatcher. Generalized^2 sequential machine maps. *Journal of Computer Systems Science*, 4:339–367, 1970.

[You89] J. H. You. Enumerating outer narrowing derivations for constructor-based term rewriting systems. *Journal of Symbolic Computation*, 7:319–341, 1989.