TECHNISCHE UNIVERSITÄT DRESDEN

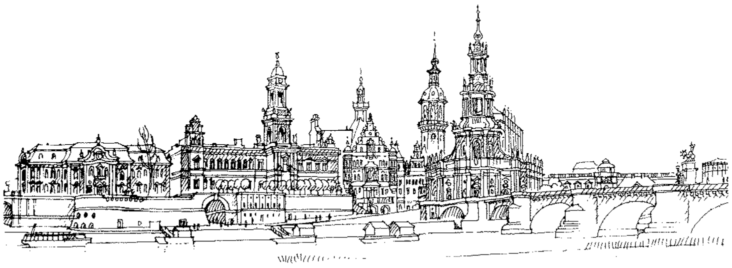
Fakultät Informatik

Technische Berichte
Technical Reports
ISSN 1430-211X

No. TUD / FI 96 / 21 - November 1996

Sebastian Maneth, Technische Universität Dresden
Heinz Faßbender, Deutsche Forschungsanstalt für Luft und Raumfahrt e. V.

A Strict Border for the Decidability of E–Unification for Recursive Functions



Technische Universität Dresden
Fakultät Informatik
D-01062 Dresden
Germany

URL: http://www.inf.tu-dresden.de/

A Strict Border for the Decidability of E–Unification for Recursive Functions

HEINZ FASSBENDER
Deutsche Forschungsanstalt für Luft- und Raumfahrt e.V.
Qualität und Sicherheit
D-51170 Köln, Germany
e-mail: Heinrich.Fassbender@dlr.de

and

SEBASTIAN MANETH
Grundlagen der Programmierung
Institut für Softwaretechnik I, Fakultät Informatik
Technische Universität Dresden
D-01062 Dresden, Germany
e-mail: maneth@inf.tu-dresden.de

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. In this paper, we answer 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, particular E–unification problems have to be solved quite frequently. 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.

$$\begin{array}{lcl} \mathit{sublist}(x,y) & :- & y = \mathit{append}(z_1, \mathit{append}(x,z_2)) \\ \mathit{append}(\mathit{CONS}(x_1,x_2),y_1) & := & \mathit{CONS}(x_1, \mathit{append}(x_2,y_1)) \\ \mathit{append}(\mathit{NIL},y_1) & := & y_1 \end{array}$$

Figure 1: A functional logic program.

In computations of the predicate *sublist*, an equation like

$$t_y = \mathit{append}(z_1, \mathit{append}(t_x, z_2))$$

has to be solved, where t_y and t_x are the current values of the variables y and x, respectively. Clearly, this is nothing else but the E_{append}–unification problem for the terms t_y and $\mathit{append}(z_1, \mathit{append}(t_x, z_2))$; and the computing machinery tries to compute an E_{append}–unifier φ, i.e., it tries to answer the question whether t_y and $\mathit{append}(z_1, \mathit{append}(t_x, z_2))$ 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 which 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 a lot of *specific theories* that are decidable. Such as associativity A [Mak77], associativity and commutativity AC [Sti75], etc. (see [BS93] 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 (for short trs's. 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 trs's, 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 intension was not to put very specific conditions on the trs's (or even on the

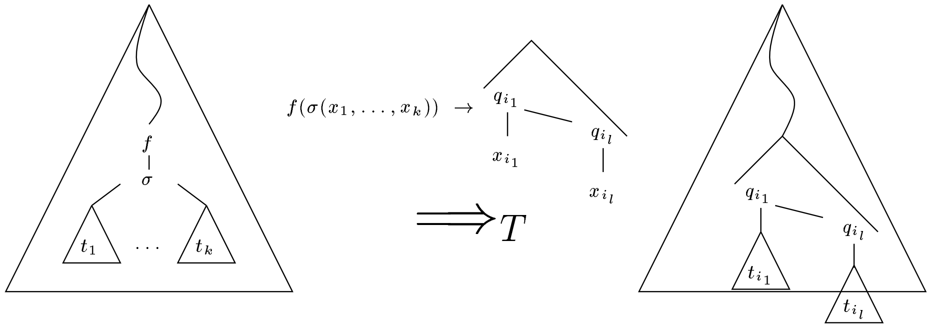


Figure 2: A derivation step of a top–down tree transducer

goals), but to rather show for known classes of trs's where the border of un–/decidability of the induced E–unification problems lies. We feel that this gives a 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 which defines a function f that 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 which computes 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 numbers are described by terms over the input symbols s and 0 of rank 1 and 0, respectively. Furthermore, the addition is presented as 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 <u>two</u> 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

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

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

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 which are induced by trs's 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 in between these two classes, the number of functions seems to be crucial in respect to decidability of E–unification for recursive functions.

This paper is organized as follows. Section 2 contains preliminaries. The considered trs's 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 which 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 which are induced by monadic tree homomorphisms. Finally in Section 7, we give some conclusions and further (un)decidability results which follow immediately from the results of Sections 5 and 6.

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

2 Preliminaries

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

2.1 General Notations

The *set of non-negative integers* is denoted by $I\!N$. The *empty set* is denoted by $\emptyset$. For some $j \in I\!N$, $[j]$ denotes the set $\{1, \ldots, j\}$; therefore $[0] = \emptyset$. For $i, j \in I\!N$ 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 $|w|$.

2.2 Ranked Alphabets, Variables, and Terms

A pair $(\Omega, \underline{rank}_\Omega)$ is called *ranked alphabet* if Ω is a finite set and $\underline{rank}_\Omega : \Omega \to I\!N$ is a total function. For every symbol $f \in \Omega$, the natural number $\underline{rank}_\Omega(f)$ 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 (i) enumerating the finitely many non–empty subsets $\Omega^{(i)}$, or by (ii) 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 $(\Omega, \underline{rank}_\Omega)$ 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 use always x's and in goals, we use always z's.

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 $I\!N^*$ 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 $O(t) = \{\varepsilon\} \cup \bigcup\limits_{i \in [n]} \{iu \mid u \in O(t_i)\}$.

Obviously, this representation becomes ambiguous if the ranked alphabet contains a symbol with rank larger than 9. Then one should be adapting *Dewey's notation* by placing dots between the i's.

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 $\varphi : \mathcal{V} \rightarrow T(\Omega, \mathcal{V})$ is called $(\mathcal{V}, \Omega)$–*substitution*, if $\{v \mid v \in \mathcal{V}$ 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 $l \rightarrow r$ 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 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 $\{(l = r) \mid (l \rightarrow r) \in R\}$. 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 Δ of *constructors*.

Definition 3.1 (top–down tree transducer)

A *top–down tree transducer* is a tuple (F, Δ, R) such that

- F is a ranked alphabet of *function symbols* which are all of rank one,
- Δ is a 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 rather define the derivation relation term rewriting system–like and allow nesting of function symbols.

Definition 3.2 (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 $t, s \in T(F \cup \Delta, \mathcal{V})$: $t \Rightarrow_T s$ iff 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 3.3 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 $\mathcal{U}_{mon-tdT}$.

Definition 3.4 (E–unification problem for mon–tdT)

The *E–unification problem for monadic top–down tree transducers*, denoted by $\mathcal{U}_{mon-tdT}$, 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:

$$nf(\varphi(t), \Rightarrow_T) = nf(\varphi(s), \Rightarrow_T).$$ □

As mentioned before, $\mathcal{U}_{mon-tdT}$ will be shown to be undecidable. We also define the E–unification problem $\mathcal{U}^{\widehat{T}}_{mon-tdT}$ for mon–tdT and a *particular* given monadic top–down tree transducer $\widehat{T}$. Clearly, undecidability of $\mathcal{U}_{mon-tdT}$ is a consequence of the undecidability of $\mathcal{U}^{\widehat{T}}_{mon-tdT}$.

Definition 3.5 (particular E–unification problem for mon–tdT)

Let $\widehat{T} = (F, \Delta, R)$ be a monadic top–down tree transducer. The *E–unification problem for* $\widehat{T}$, denoted by $\mathcal{U}^{\widehat{T}}_{mon-tdT}$, is the following problem:

Given: Two terms t and s over F, Δ, and $\mathcal{V}$.
Question: Are t and s $E_{\widehat{T}}$–unifiable?
In other words:
Does there exist a substitution $\varphi \in Sub(\mathcal{V}, \Delta)$ such that the following condition holds:

$$nf(\varphi(t), \Rightarrow_{\widehat{T}}) = nf(\varphi(s), \Rightarrow_{\widehat{T}}).$$ □

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 3.6 (tree homomorphism)

Let $T = (F, \Delta, R)$ be a top–down tree transducer, then T is called *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 will be denoted by $\mathcal{U}_{mon\text{–}Hom}$. The corresponding class of sets of equations is denoted by $\mathcal{E}_{mon\text{–}Hom}$, In Section 6 it will be 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 trs's. E.g.,

- Peano's axioms without multiplication which can be expressed in Pressburger arithmetic [Pre27]
- monoids [Mak77]
- particular, restricted, confluent string rewriting systems [OND95]

More closely related to the classes of trs's that we consider here, are the following results:

1. *standard theories* [Nie96] which are an extension of *shallow theories* [CHJ94]. Shallow theories are equational theories induced by particular trs's, in the rules of which variables may only occur at depth at most one.
2. particular, restricted convergent (that is, terminating and confluent) trs's, as considered in [Hul80, KN87, Mit94]
3. finite, length–reducing, and confluent Thue systems [NO90]
4. particular, restricted, confluent, constructor–based, and linear trs's [LR96]

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

4.1 Standard and Shallow theories

Shallow variables are variables occurring in a term at depth at most one. Shallow theories are induced by trs's 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, non–shallow variables may occur in particular 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 non–monadic cases, too. Thus there are shallow theories which 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 non–shallow variables.

4.2 Results by Mitras

The reference which is most relevant and also closest related to our results, is [Mit94]. There, decidability results for three different classes of convergent trs's are given: The E–unification problem is decidable, if E is induced by

(A) 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].

(B) 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, occurs also in l. This is an extension of [KN87].

(C) 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, per function symbol there may be at most one rule with a flat term as 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 (A) to (C) are defined for non–monadic trs's, too. Thus, in each class there are trs's which are not in *mon–Hom*. On the other hand, consider a monadic tree homomorphism which includes the following rule

$$h(\gamma(x_1)) \to \sigma(h(x_1)) \tag{1}$$

Then, clearly it is neither in the class described under (A) (because it is neither a variable, a ground term, nor a constructor–only term) nor in the class described under (B) (because the subterm $h(x_1)$ on the right–hand side does not occur in the left–hand side) . However, the right–hand side of rule (1) is indeed a flat term, and therefore this monadic tree homomorphism could be in the class (C). Now, consider a monadic tree homomorphism consisting of the rule (1) and additionally of the following rule

$$h(\gamma'(x_1)) \to \sigma(h(x_1)) \tag{2}$$

One restriction of (C) is, that per function symbol only one rule with a flat term as right–hand side is allowed. Therefore, the monadic tree homomorphism that includes the rules (1) and (2), is not in the class described under (C). Of course, from the incomparability of the class *mon–Hom* with the classes of trs's described under (A) to (C), the incomparability of the induced classes of equational theories follows.

4.3 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 trs's and therefore let us discuss how the results by Otto et al. are related to our result. (In the following, let us consider Thue systems as monadic trs's.)

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 which 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 trs's) is incomparable with $\mathcal{E}_{mon\text{–}tdT}$.

4.4 Restricted, Confluent, Constructor–Based, and Linear trs's

Limet and Réty show in [LR96] that the E–unification problem for equational theories induced by constructor–based, confluent trs's respecting the following four conditions

1. linearity of rewrite rules,
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,
3. no nested functions in right–hand sides, and
4. linearity of the goal,

is decidable. Since there is no restriction to monadic rules, there are theories induced by trs's respecting 1. to 4. 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, there are equational theories in $\mathcal{E}_{mon\text{–}Hom}$ which are not in the class of equational theories induced by trs's respecting 1. to 4. Furthermore, as we will see in the proof of Lemmata 6.1 and 6.2 the cases of non–linear goals turn out to be the most interesting ones.

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 $\widehat{T}$ such that given two terms s and t, the problem $\mathcal{U}^{\widehat{T}}_{mon\text{-}tdT}$ 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 $\mathcal{U}^{\widehat{T}}_{mon\text{-}tdT}$ are consequences of the result (2).

To prove the result (2) we reduce the modified Post's correspondence problem [Pos46] to it. For this purpose, we first recall the modified Post's correspondence problem in Figure 4, where Σ is an alphabet with $card(\Sigma) \geq 2$. Instead of the usual formal definition where a problem is described by a formal language, we recall it in the same informal way as we have defined $\mathcal{U}_{mon\text{-}tdT}$ in Definition 3.4. The sequence of indices $i_2, \ldots, i_n$ is called a *solution* of the modified Post's correspondence problem.

Given:	Two k–tuples $(u_1, \ldots, u_k), (v_1, \ldots, v_k) \in (\Sigma^+)^k$, where $k \geq 1$, of nonempty words over Σ.
Question:	Do there exist indices $i_1, \ldots, i_n \in [k]$, where $n \geq 1$ and $i_1 = 1$, such that the following condition holds: $u_{i_1} u_{i_2} \ldots u_{i_n} = v_{i_1} v_{i_2} \ldots v_{i_n}$?

Figure 4: The modified Post's correspondence problem.

The undecidability of the modified Post's correspondence problem can be shown by reducing the halting problem for Turing machines to it (cf. Theorem 8.8 of [HU79]). Thus, we know that for every Turing machine M and input w we can define an instance $\text{MPCP}_{M,w}$ of the modified Post's correspondence problem such that M accepts on input w iff $\text{MPCP}_{M,w}$ has a solution. Now, let us take a particular Turing machine $\widehat{M}$ such that the language $L(\widehat{M})$ of words accepted by $\widehat{M}$ is non–recursive. For $\text{MPCP}_{M,w}$ this implies two fixed tuples of words $(u_2, \ldots, u_k)$ and $(v_2, \ldots, v_k)$. The resulting undecidable problem is denoted by $\widehat{\text{MPCP}}$ and is shown in Figure 5.

Let us now reduce the problem $\widehat{\text{MPCP}}$ to the problem (2). For this purpose we construct the monadic top–down tree transducer $\widehat{T}$ and a mapping *red* which maps an arbitrary input (u_1, v_1) of $\widehat{\text{MPCP}}$ to an input $red((u_1, v_1))$ of the problem (2).

The top–down tree transducer $\widehat{T}$ is constructed in such a way, that $\bar{u}_1'(f(z))$ and $\bar{v}_1'(g(z))$ are $E_{\widehat{T}}$–unifiable, if and only if $\widehat{\text{MPCP}}$ has a solution for input (u_1, v_1); where $\bar{u}_1'$ and $\bar{v}_1'$

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 5: The problem $\widehat{\text{MPCP}}$.

are corresponding monadic term representations of u_1 and v_1, respectively.

Theorem 5.1 There is a monadic top–down tree transducer $\widehat{T} = (F, \Delta, R)$, 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 $v_1[x/g(z)]$ $E_{\widehat{T}}$–unifiable; where $f, g \in F$.

Proof:

Let us now construct the monadic top–down tree transducer $\widehat{T}$. We start from the fixed tuples of words (implied by $\widehat{M}$, as mentioned above)

$$(u_2, \ldots, u_k) \text{ and } (v_2, \ldots, v_k).$$

In the construction of $\widehat{T}$, the sequence of indices as well as corresponding words of the modified Post's correspondence problem have to be denoted by monadic trees. For this purpose, every letter of a u_i or v_i and every index is defined as a unary constructor and we use the constant constructor e (empty word) as the leaf of a monadic tree. In the description of the set of rewrite rules of $\widehat{T}$, we denote the part t of a monadic tree which corresponds to w_i or v_i, by $\bar{w}_i$ or $\bar{v}_i$, respectively. We also omit parentheses in the representation of monadic trees.

Since, in Definition 3.1, it is required that for every function symbol and for every constructor, there exists exactly one rewrite rule, we also have to construct rewrite rules for constructors which correspond to letters in u_i or v_i. Such rules are only of technical nature. Nevertheless, in their definitions, we have to ensure that the trees $f(z)$ and $g(z)$ do not become $E_{\widehat{T}}$–unifiable by these rules. That means, for every $E_{\widehat{T}}$–unifier φ of $f(z)$ and $g(z)$, we have to ensure that $\varphi(z)$ only consists of constructors which correspond to indices and that it does not consist of constructors which correspond to letters. For this purpose, Δ includes two additional constant constructors c_f and c_g and every monadic tree the root of which is labeled by a constructor which does not correspond to an index, is mapped by

f and g to c_f and c_g, respectively. Then, for every substitution φ, where $\varphi(z)$ consists of a constructor which does not correspond to an index, we have

$$\varphi(f(z)) \neq \varphi(g(z)),$$

because their leaves are labeled by c_f and c_g, respectively. In particular, φ is not an $E_{\widehat{T}}$–unifier of $f(z)$ and $g(z)$. Finally, we have to define f– and g–rules for c_f and c_g which will never be applied.

The top–down tree transducer $\widehat{T}$ is constructed as follows:
$\widehat{T} = (F, \Delta, R)$, 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:

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

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

$$\begin{array}{lllll}
 & \{f(ix_1) & \to \bar{u}_i f(x_1) & \mid i \in \mathit{Indices}\} \\
\cup & \{f(e) & \to e\} & \\
\cup & \{g(ix_1) & \to \bar{v}_i g(x_1) & \mid i \in \mathit{Indices}\} \\
\cup & \{g(e) & \to e\} & \\
\cup & \{f(ax_1) & \to c_f & \mid a \in \mathit{Letters}\} \\
\cup & \{g(ax_1) & \to c_g & \mid a \in \mathit{Letters}\} \\
\cup & \{f(c_f) & \to c_f\} & \\
\cup & \{f(c_g) & \to c_f\} & \\
\cup & \{g(c_f) & \to c_g\} & \\
\cup & \{g(c_g) & \to c_g\} &
\end{array}$$

From the construction of $\widehat{T}$, it obviously follows for every $i_2, \dots, i_n \in [k]$:

$$nf(f(i_2 \dots i_n e), \Rightarrow_{\widehat{T}}) = \bar{u}_{i_2} \dots \bar{u}_{i_n} e$$

and

$$nf(g(i_2 \dots i_n e), \Rightarrow_{\widehat{T}}) = \bar{v}_{i_2} \dots \bar{v}_{i_n} e.$$

That means, the sequence of indices $i_2, \dots, i_n$ is a solution of $\widehat{\text{MPCP}}$ for the input (u_1, v_1), if and only if the substitution $[z/i_2 \dots i_n e]$ is an $E_{\widehat{T}}$–unifier of $\bar{u}_1 f(z)$ and $\bar{v}_1 g(z)$. Obviously, the mapping *red* is simply given by $\mathit{red}(u_1, v_1) = (\bar{u}_1(x), \bar{v}_1(x))$.

$$\begin{array}{rcl} f(5x_1) & \rightarrow & abcf(x_1) \\ g(5x_1) & \rightarrow & cbag(x_1) \end{array}$$

Figure 6: Rewrite rules of $\widehat{T}$ for $w_5 = abc$ and $v_5 = cba$.

Since, as explained above, only $E_{\widehat{T}}$–unifiers of $f(z)$ and $g(z)$ (and therefore of $\bar{u}_1 f(z)$ and $\bar{v}_1 g(z)$, respectively) do exist which have this form, we have reduced $\widehat{\text{MPCP}}$ to our problem. Thus, from the undecidability of $\widehat{\text{MPCP}}$, the undecidability of our problem follows. □

For illustrating the construction in the previous proof, in Figure 6, we list the f– and g–rules for an input of $\widehat{\text{MPCP}}$, where we only consider the fifth components $w_5 = abc$ and $v_5 = cba$ of the two tuples, respectively. Furthermore, we only list the rewrite rules for constructors which correspond to index 5.

From Theorem 5.1 we can of course follow the undecidability of $\mathcal{U}^{\widehat{T}}_{mon-tdT}$, where $\widehat{T}$ is the monadic top–down tree transducer defined in the proof of Theorem 5.1, and of the uniform problem $\mathcal{U}_{mon-tdT}$.

Corollary 5.2 Let $\widehat{T}$ be the monadic top–down tree transducer defined in the proof of Theorem 5.1. Then, $\mathcal{U}^{\widehat{T}}_{mon-tdT}$ is undecidable.

Corollary 5.3 $\mathcal{U}_{mon-tdT}$ is undecidable.

By proving the undecidability of $\mathcal{U}_{mon-tdT}$, 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 which follow immediately from Theorem 5.1.

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

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 $\mathcal{U}_{mon-Hom}$.

Since we are dealing with monadic constructors, let us again switch to word–notation, i.e., let $\gamma_1(\dots\gamma_n(\alpha))$ be denoted as $\gamma_1\cdots\gamma_n\alpha$. 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 which 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 6.1 Let $T = (\{h\}, \Delta, R)$ be a monadic tree homomorphism and let $v \in \Delta_1^*$. Then, $\mathcal{U}_{mon-Hom}$ is decidable for goals of the form $z_1 =_{E_T} vh(z_1)$.

Proof:

If $|v| = 0$, then we get the goal $z_1 =_{E_T} h(z_1)$ which is solvable iff there is a rule $h(\alpha) \to w\alpha$ in R, where $\alpha \in \Delta_0, w \in \Delta_1^*$ and w can be *consumed*.

For $|v| \geq 1$, let $z_1 = vz_1'$, where z_1' is a new variable. Then, the goal becomes $vz_1' =_{E_T} vh(vz_1')$ which reduces to $z_1' =_{E_T} h(vz_1')$. If $nf(h(vz_1'), \Rightarrow_T) = w$ with $w \in \Delta^T$, then $z_1 = vw$ is a solution to the E–unification problem. If $nf(h(vz_1'), \Rightarrow_T)$ is of the form $v'h(z_1')$, then we can again substitute a new variable z_2' in same way as before to get the goal $z_2' =_{E_T} h(v'z_2')$. The decision procedure *loop* which is shown in Figure 7, iterates this procedure with the termination condition as explained below.

Consider Figure 7. An occurrence of **break** will stop the loop immediately and return the value given as argument. Intuitively, this loop calculates $h^i(vz)$ for $i \geq 1$. If there is an i such that $h^i(vz)$ reduces to

1. a constructor term, i.e., a word in Δ^T, then there is a solution to the E–unification problem.
2. a constructor term which is followed by a variable, then the goal has a solution, if and only if the constructor term in the last iteration step is empty.
3. a constructor term which is followed by a function call $h(z_i')$ and the constructor term in the last iteration step is empty, then the goal has a solution, if and only if there exists a rule $h(\alpha) \to w\alpha \in R$ and w is consumed.
4. a constructor term which is followed by a function call $h(z_i')$ and the constructor term in the last iteration step is not empty, then we start the next iteration.

The decision procedure *loop* checks for every iteration of i, if new constructors have occurred that did not occur in previous iterations. The termination condition, i.e., that no new constructors have occurred, is justified, since h is total deterministic. Thus, only

$loop(v : \Delta_1^*) : \{true, false\}$
$S_1 := \{\gamma \in \Delta_1 \mid \gamma$ occurs in $v\}; i := 1; v_1 := v;$
repeat
  case $nf(h(v_i z_i'), \Rightarrow_T)$
  (1) $= w$, with $w \in \Delta^T$: **break**($true$);
  (2) $= v_{i+1} z_i'$, with $v_{i+1} \in \Delta_1^*$: **if** $|v_{i+1}| = 0$ **then break**($true$)
    else break($false$);
  (3) $= h(z_i')$: **if** there is a rule $h(\alpha) \to w\alpha$ in R **and**
    w is consumed by other rules **then break**($true$);
    else break($false$);
  (4) $= v_{i+1} h(z_i')$, with $v_{i+1} \in \Delta_1^+$: Let $z_i' = v_{i+1} z_{i+1}'$,
    where z_{i+1}' is a new variable;
  end (of case)
  $S_{i+1} := S_i \cup \{\gamma \in \Delta_1 \mid \gamma$ occurs in $v_{i+1}\}$;
  $i := i + 1$;
until $S_i = S_{i-1}$;
break($false$).

Figure 7: The decision procedure *loop*.

rules that have already been applied, can be applied in a new iteration step and therefore no new constructors can occur in following iterations. This would lead to an infinite computation. Hence, in this case *loop* terminates with the result *false*, because of the non–E_T–unifiability of the initial terms. The algorithm will always terminate, since $S_i \subseteq \Delta_1$ for all i, and Δ_1 is a finite set. And therefore there is an N such that $S_N = S_{N+\nu}$ for all $\nu > 0$. If the loop stops, because $S_i = S_{i-1}$, then there is no E_T–unifier for the given goal as explained above.

From the previous discussion it follows that the loop terminates and decides, if there is a solution to $z_1 =_{E_T} vh(z_1)$ with $v \in \Delta_1^+$. This concludes the proof of Lemma 6.1. □

Lemma 6.2 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:

For showing the correctness of the lemma, we define the decision procedure *solve* in Figure 8 which is, in opposite to the procedure *loop* in Figure 7, recursively defined, because it is a better formalization for this procedure.

The procedure *solve* takes as input a word $v \in \Delta_1^*$ and a subset of Δ_1^* which is used for checking whether all possible cases are checked in Case (5). Since the procedure is recursively called, it may yield a sequence of *true* and *false*. The two terms are E_T–unifiable, if *solve* yields for v and $\{v\}$ as input at least one *true*. Otherwise, the two terms are not E_T–unifiable.

$solve(v : \Delta_1^*, S : \mathcal{P}(\Delta_1^*)) : \{true, false\}^*$

let $\alpha \in \Delta_0$; $\beta, \beta' \in \Delta_1$; $w \in \Delta_1^*$ **in**

if $v = \varepsilon$ **then** (0) **if** there is a rule $h(\alpha) \to w\alpha \in R$ and w is consumed by other rules **then** **break**(*true*);

if $v \neq \varepsilon$ **then**

(1) **if** there is a rule $h(\alpha) \to v\alpha \in R$ **then** **break**(*true*);

(2) **if** there is a rule $h(\beta x) \to v\beta\alpha \in R$ **then** **break**(*true*);

(3) **for** every rule $h(\beta x) \to v\beta w h(x) \in R$ **do**

  substitute z by $\beta z'$;
  if $w = \varepsilon$ **then** $solve(\varepsilon, \emptyset)$;
  if $w \neq \varepsilon$ **then** **break**(*loop* (w));

(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**

  substitute z by $\beta z'$;
  $solve(u\beta, \{u\beta\})$;

(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**

  substitute z by $\beta z'$;
  if $u\beta \notin S$ **then** $solve(u\beta, S \cup \{u\beta\})$;

break(*false*).

Figure 8: The decision procedure *solve*.

In particular, *solve* compares v with the right–hand sides of the rewrite rules in R. Rewrite rules the right–hand sides of which start with something different from a prefix from v are not considered, because they are not able to produce the term vz.

For the other rules, the following cases may occur:

- If $v = \varepsilon$, then the E_T-unifiability is decided as in the proof of Lemma 6.1.
- If a stop rule exists of the forms in (1) or (2), then the terms are simply E_T–unifiable (a stop rule is a rule with right–hand side in Δ^T).
- If $v\beta$ is a prefix of the right–hand side (Case (3)), then we substitute z by $\beta z'$ and we get a problem as in Case (0) or as in Lemma 6.1. In the former case, we simply call *solve* which evaluates as in Case (0) and stops. In the latter case, we decide as in the proof of Lemma 6.1.
- In Case (4), w the length of which is greater than 1, is a prefix of v. Then we substitute z by $\beta z'$ and we get a problem as in the previous goal. But the new $u\beta$ is shorter than the old v. Hence, the application of this case reduces the "complexity"

of the problem. It may be applied only for at most $|v|$ times. Then, we would have Case (0), i.e., the procedure also stops in this case.

- Case (5) is the most difficult case. In this case, we get a goal which is very close to the initial one. The length of the word in front of $h(z)$ is still the same, i.e., this rule may be applied infinitely often. For stopping this infinite loop, we use the second argument of *solve*. If we have called *solve* for the same situation before, then we will not get a solution by the same argument as in the proof for Lemma 6.1. Otherwise, we call *solve* with the resulting goal. Since Δ is finite, the number of possible prefixes of length $|v|$ is finite. Hence, *solve* also stops in this case.

That means, *solve* always stops with the right result, i.e., we have constructed a decision procedure for the considered goal. This concludes the proof of Lemma 6.2. □

Remark that, the formal verification of the termination of *solve* as well as for *loop* is out of the scope of this paper. For instance, for *solve*, it can be formally proved by introducing a measure which depends on the set S in the second argument of *solve* and the length of the first argument v. Since in recursive calls of *solve* either the length of v is reduced or S is extended and we have the termination conditions $v = \varepsilon$ or $u\beta \notin S$, respectively, and the capacity of S is finite, we get the termination.

We now quote a result for monadic top–down tree transducers which 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 (GSM's) [Gin62]. We can therefore make use of the following well known lemma.

Lemma 6.3 (Theorem 3.1 of [GR63]) Regular languages are closed under GSM mappings.

Now, we are ready to prove the following theorem.

Theorem 6.4 $\mathcal{U}_{mon\text{–}Hom}$ is decidable.

Proof:

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. $vh^i(z_1) =_{E_T} v'h^j(z_1)$
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 6.3 and the fact that the language Δ^T is regular, we get that the language $L_1 = \{vh^i(z_1) \mid z_1 \in \Delta^T\}$ 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: w.l.o.g. $|v| < |v'|$. 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 $h'(w) = h^{j-i}(w)$ 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 6.1.
2. If $i = j$, then there is a solution iff $|\hat{v}| = 0$.
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 6.2.

Case 4: Let $L_1 = \{vh^i(z_1) \mid z_1 \in \Delta^T\}$ and $L_2 = \{v'h^j(z_2) \mid z_2 \in \Delta^T\}$. 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.

This concludes the proof of Theorem 6.4. $\square$

7 Conclusions and Immediate Results

In this paper we have shown the undecidability of the E–unification problem $\mathcal{U}_{mon-tdT}$ and the decidability of the E–unification problem $\mathcal{U}_{mon-Hom}$. 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 for E–unification for recursive functions.

The following undecidability results follow immediately from the verified results.

Undecidability for top–down tree transducers

The class $\mathcal{E}_{mon-tdT}$ is a subclass of $\mathcal{E}_{tdT}$, but since $\mathcal{U}_{mon-tdT}$ 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.

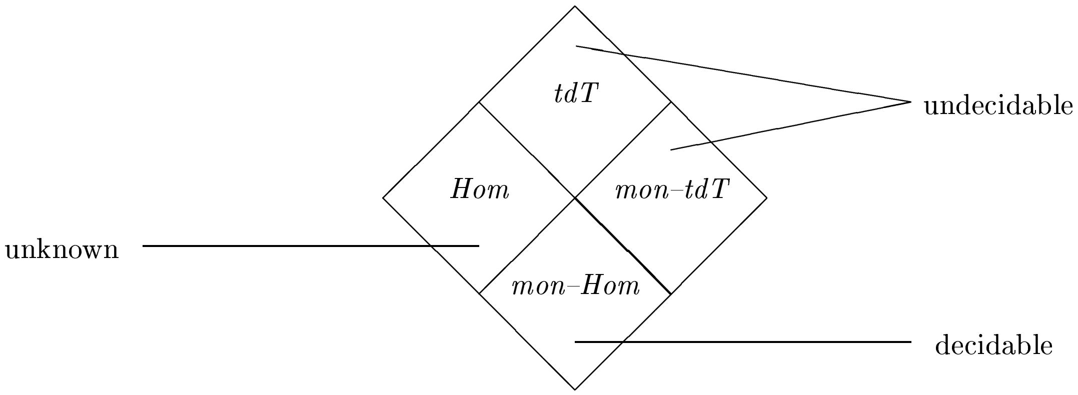


Figure 9: Results of this paper.

Our results are depicted in Figure 9. There, we have depicted the different classes of tree transducers which 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, Mitras' flat term restriction for linear, convergent trs's (as discussed in the introduction) plus the additional possibility to call functions that are defined by restricted tree homomorphisms.

Acknowledgments

We are thankful to Friedrich Otto for pointing out to us the possibility to improve our previous decidability result by using the modified version of Post's correspondence problem.

References

[BS93] 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*. Oxford Univ. Press, 1993.

[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, Vol. B*, chapter 6, pages 243–320. Elsevier, 1990.

[Ech90] R. Echahed. On completeness of narrowing strategies. *Theoret. Comput. Sci.*, 72:133–146, 1990.

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

[EV91] J. Engelfriet and H. Vogler. Modular tree transducers. *Theoret. Comput. Sci.*, 78:267–304, 1991.

[FHVV93] Z. Fülöp, F. Herrmann, S. Vágvölgyi, and H. Vogler. Tree transducers with external functions. *Theoret. Comput. Sci.*, 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 *LNCS*, pages 194–208. Springer-Verlag, September 1996.

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

[GR63] S. Ginsburg and G. F. Rose. Operations which preserve definability in languages. *J. Assoc. Comput. Mach.*, 10(2):175–195, 1963.

[Han96] M. Hanus. Proceedings of the Poster Session of ALP'96 — Fifth International Conference on Algebraic and Logic Programming. Technical Report 96-6, RWTH Aachen, 1996.

[HU79] J. W. Hopcroft and J. D. Ullmann. *Introduction to automata theory, languages, and computation.* Addison-Wesley, 1979.

[Hue80] G. Huet. Confluent reductions: abstract properties and applications to term rewriting systems. *J. Assoc. Comput. Mach.*, 27:797–821, 1980.

[Hul80] J.M. Hullot. Canonical forms and unification. In *Proceedings of the 5th conference on automated deduction*, volume 87 of *LNCS*, pages 318–334. Springer–Verlag, 1980.

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

[LR96] S. Limet and P. Rety. On decidability of unification modulo rewrite systems. Technical report, LIFO – Universite d'Orleans, 1996. Research Report LIFO ??? (to appear), an extended abstract of this paper appeared in [Han96].

[Mak77] G. S. Makanin. The problem of solvability of equations in a free semigroup. *Math. 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 Logic Programming*, 12:191–223, 1992.

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

[Nie96] R. Nieuwenhuis. Basic paramodulation and decidable theories. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, 1996. (to appear).

[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 *LNCS*, pages 276–291. Springer–Verlag, July 1990.

[OND95] F. Otto, P. Narendran, and D. J. Dougherty. Some independance results for equational unification. In J. Hsiang, editor, *Proceedings of the 6th International Conference on Rewriting Techniques and Applications*, volume 914 of *LNCS*, pages 367–381, 1995.

[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. *Bull. 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. Comput. Mach.*, 20:23–41, 1965.

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

[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. *J. Comput. Syst. Sci.*, 4:339–367, 1970.

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