Universität München
Centrum für Informations- und Sprachverarbeitung

On Existential Theories of List Concatenation

Klaus U. Schulz

The paper has been presented at the Conference on Computer Science Logic 94 in Kazimierz, Poland. A longer version will soon appear as CIS report.

On Existential Theories of List Concatenation

Klaus U. Schulz*
CIS, University Munich, Wagmüllerstr. 23
80538 Munich, Germany
e-mail: schulz@cis.uni-muenchen.de
phone: (+49 89) 211 0667

Abstract

We discuss the existential fragments of two theories of concatenation. These theories describe concatenation of possibly nested lists in the algebra of *finite* trees with lists and in the algebra of *rational* trees with lists. Syntax and the choice of models are motivated by the treatment of lists in PROLOG III. In a recent prototype of this language, Colmerauer has integrated a built-in concatenation of lists, and the constraint-solver checks satisfiability of equations and disequations over concatenated lists. But, for efficiency reasons satisfiability is only tested in a rather approximative way[1]. The question arises if satisfiability is decidable. Our main results are the following. For the algebra of finite trees with lists, the existential fragment of the theory is decidable. For the algebra of rational trees with lists, the positive existential fragment of the theory is decidable. Problems in the existential fragment may be traced back to a difficult question about solvability of word equations with length constraints for variables.

1 Introduction

Quine [9] has shown that the theory of concatenation is undecidable. The existential fragment of the theory was shown to be decidable by Büchi and Senger [3], building up on Makanin's decidability result for solvability of word equations [7]. Concatenation, in the sense of Quine, is an operation acting on words over an alphabet of atomic letters, and the classical theory of concatenation is the theory of free monoids. In the meantime, with the development of high level programming languages, concatenation has become relevant as an operation on lists. Lists, as opposed to flat words, may contain complex objects as entries, including nested sublists, for example.

In this paper we want to discuss theories of list concatenation. We shall concentrate on two formal models that are motivated by the treatment of lists in PROLOG III. In a recent prototype of this language, Colmerauer has integrated a built-in concatenation

*Supported by EC Working Group CCL, EP 6028.

[1] We refer to a talk by Alain Colmerauer on the third Workshop on Constraint Logic Programming, Marseille, March 1993.

of lists, and the constraint-solver checks satisfiability of equations and disequations between terms with concatenated lists. For efficiency reasons, however, satisfiability is only tested in a rather approximative way. Colmerauer introduces a non-standard "naive" concatenation on a complicated "extended domain" to explain the precise answer behaviour of the solver declaratively. The question arises if satisfiability of equations and disequations between terms with concatenated lists is decidable.

Approximating the formal model of PROLOG III, we consider the algebra of *finite* trees with lists and the algebra of *rational* trees with lists. In both domains, concatenation is interpreted as a partial operation acting on lists only, free function symbols are interpreted as tree constructors. In view of the results of Quine and Büchi-Senger we only consider the existential fragment of the theories of these two structures. The syntax is more or less identical to the syntax of PROLOG III for constraints over lists. The "list constraint systems" that will be considered are finite sets of equations and disequations between terms with concatenated lists. Arbitrary existential sentences correspond to disjunctions of list constraint systems.

The paper is structured as follows. Section 2 starts with central definitions. In Section 3 we show that solvability of list constraint systems over the algebra of finite trees with lists is decidable. This implies that the existential theory of this structure is decidable. The decision procedure is based on a decomposition technique that was introduced in [2] in the context of disunification in the union of disjoint equational theories. A variant of Makanin's algorithm [7] deciding solvability of word equations is needed.

In Section 4 we consider the algebra of rational trees with lists as solution domain. It is shown that solvability of *equational* list constraint systems is decidable. Thus the positive existential theory of this algebra is decidable. We sketch how the problem of solvability of arbitrary list constraint systems over the algebra of rational trees with lists may be traced back to the following problem: given a word equation with variables $x_1, \ldots x_n$, and given a finite set of constraints of the form $|x_i| = |x_j|$ demanding that the length of the (words to be substituted for the) variables x_i and x_j has to be the same, decide if the word equation has a solution that satisfies these restrictions. Decidability of word equations with these length constraints seems to be a deep problem. G.S. Makanin (personal communication) has shown that a primitive recursive decision procedure would give a primitive recursive algorithm for deciding solvability of equations in free groups. It is known that Makanin's algorithm for free groups [8] is not primitive recursive [5].

2 List Constraint Systems and Solutions

List constraint systems

Following the syntax of PROLOG III we shall use an infinite set of list constructing symbols for representing lists. For each natural number k, let $[\ ]^k$ denote a function symbol of arity k. Let $\Sigma_L := \{[\ ]^k; k \geq 0\}$. Let Σ_F denote a disjoint finite set of free function symbols, containing at least one constant and one non-constant function symbol. The complete signature that we shall use contains binary concatenation "$\circ$", and all symbols from $\Sigma_{L\&F} := \Sigma_L \cup \Sigma_F$. X is a countably infinite set of variables. In the sequel, possibly subscripted symbols $x, y, z, \ldots$ always denote variables.

The set of all (F- and L-) *terms* is recursively defined as follows:

- every variable is an L-term and an F-term,
- if $t_1, \ldots, t_n$ are terms and $f \in \Sigma_F$ is an n-ary function symbol, then $f(t_1, \ldots, t_n)$ is an F-term and $[\ ]^n(t_1, \ldots, t_n)$ is an L-term ($n \geq 0$),
- if l_1 and l_2 are L-terms, then $l_1 \circ l_2$ is an L-term.

Terms $[\ ]^n(t_1, \ldots, t_n)$ will be written in the form $[t_1, \ldots, t_n]$. Since the infix symbol "$\circ$" is interpreted as concatenation, we omit brackets in expressions $l_1 \circ \cdots \circ l_n$. For $n = 0$, an expression $l_1 \circ \cdots \circ l_n$ denotes the empty lists $[\ ]^0$. Of course many "natural" expressions (such as those using "*cons*" and "*conc*", or Prolog-style $[t_1|l]$) are not treated as terms. It is simple to see that for all these expressions there are terms which behave in the same way, in any relevant sense. In order to keep proofs simple we have chosen a minimal syntax which captures all conventional constructions for a combination of terms with lists.

A *list constraint system* is a finite set of equations and disequations Γ of the form

$$\{s_1 \doteq t_1, \ldots, s_n \doteq t_n, s_{n+1} \not\doteq t_{n+1}, \ldots, s_{n+m} \not\doteq t_{n+m}\}$$

where the s_i and t_i are terms.

Example 2.1 Let $\Sigma_F = \{f, g, a, b\}$ where f is binary, g is unary, and a and b are constants. Then $\Gamma_1 = \{[f(z, [x] \circ x), g(y \circ y)] \doteq [f(g(x \circ y), [y] \circ [b, a]), z], y \not\doteq [\ ]\}$ and $\Gamma_2 = \{[x] \doteq x\}$ are list constraint systems.

Two solution domains

We assume that trees (and subtrees) are formalized as usual, i.e., as sets of labelled paths. Paths (= positions) are finite sequences of positive natural numbers. A *tree with lists* is a tree with labels in $\Sigma_{L\&F}$, the arity of the label giving the branching degree at the node. A tree with lists is rational if it has only a finite number of distinct subtrees.

In order to solve list constraint systems we shall consider the two domains $\mathcal{T}_{\text{fin}}^{\Sigma_{L\&F}}$ and $\mathcal{T}_{\text{rat}}^{\Sigma_{L\&F}}$ of all finite (resp. rational) trees with lists. Elements of these domains will often be written in the form $f(t_1, \ldots, t_k)$ or $[\ ]^k(t_1, \ldots, t_k) = [t_1, \ldots, t_k]$, where the t_i denote subtrees. Trees of the form $[t_1, \ldots, t_k]$ will be called lists of length k.

Both domains may be turned into (partial) algebras over the signature $\Sigma_{L\&F} \cup \{\circ\}$: free function symbols $f \in \Sigma_F$ and list symbols $[\ ]^k$ are interpreted as tree constructors, the interpretation of "$\circ$" is the partial function

$$\circ_T : \langle [t_1, \ldots, t_n], [t_{n+1}, \ldots, t_{n+m}] \rangle \mapsto [t_1, \ldots, t_n, t_{n+1}, \ldots, t_{n+m}].$$

Solutions and finite-tree solutions

A tree assignment is a mapping $\alpha : X \longrightarrow \mathcal{T}_{\mathrm{rat}}^{\Sigma_{L\&F}}$. Tree assignments will be used to associate with arbitrary terms t an interpretation $t^\alpha \in \mathcal{T}_{\mathrm{rat}}^{\Sigma_{L\&F}}$. But, since "$\circ_T$" is a partial operation, we have to be careful. We say that $x \in X$ *has type* L *with respect to* t if t has a subterm of the form $x \circ l$ or $l \circ x$. It is not hard to see that t^α is defined if and only if x^α is a list, for every variable x which has type L with respect to t. The variable x *has type* L *with respect to the constraint system* Γ if there is a term t in Γ such that x has type L with respect to t. A partial tree assignment σ is *consistent for* Γ if σ assigns a list x^σ to every variable x that has type L with respect to Γ and an arbitrary tree with lists to the remaining variables y of Γ.

Definition 2.2 Let Γ be a constraint system. A *rational-tree solution* (or simply a solution) of Γ is a partial tree assignment σ which is consistent for Γ such that $s^\sigma = t^\sigma$ ($s^\sigma \neq t^\sigma$) whenever Γ contains an (dis)equation $s \doteq t$ ($s \mathrel{\dot{\neq}} t$). A *finite-tree solution* is a solution σ where $x^\sigma \in \mathcal{T}_{\mathrm{fin}}^{\Sigma_{L\&F}}$, for all variables x occurring in Γ.

Example 2.3 The assignment $x \mapsto [b, a], y \mapsto [b, a], z \mapsto g([b, a, b, a])$ is a finite-tree solution of the constraint system Γ_1 given in Example 2.1. The system Γ_2 does not have a finite-tree solution. But there exists a solution σ which maps x to the rational tree $[[\ldots[\ldots]\ldots]]$.

Flat and nontrivial constraint systems

A term t is called *flat* if t is a variable, if t has the form $f(x_1, \ldots, x_n)$ ($f \in \Sigma_F$), or if t has the form $l_1 \circ \cdots \circ l_n$ ($n \geq 0$) where the arguments l_i are variables or terms of the form $[x]$. A *flat constraint system* is constraint system Γ where both sides of disequations are variables and the left-hand (right-hand) sides of equations are variables (flat terms). Obviously it is possible to compute for an arbitrary list constraint system Γ a flat list constraint system Γ' that is equivalent in the sense that every (finite-tree) solution of Γ can be extended to a (finite-tree) solution of Γ' and every (finite-tree) solution of Γ' is a (finite-tree) solution of Γ. (We just have to introduce additional variables x and new equations of the form $x \doteq t$ in order to get rid of complex subterms.)

A flat list constraint system is *trivial* if it contains an equation $x \doteq t$, where t is a non-variable F-term, and if at the same time x has type L with respect to Γ, or x occurs in an equation $x \doteq l$ where l is a non-variable L-term. Obviously, triviality can be detected algorithmically, and trivial systems are unsolvable. All list constraint systems that will be considered in the following are assumed to be flat and non-trivial.

3 Decidability Result for Finite Tree Solutions

In this section we want to prove the following theorem.

Theorem 3.1 *It is decidable if a list constraint system has a finite-tree solution.*

List constraint systems $\Gamma = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n, s_{n+1} \not\doteq t_{n+1}, \ldots, s_{n+m} \not\doteq t_{n+m}\}$ represent existential sentences γ of the form $\exists \vec{x}((\bigwedge_{i=1}^{n} s_i = t_i) \wedge (\bigwedge_{j=n+1}^{n+m} \neg s_j = t_j))$. Finite-tree solvability of Γ corresponds to validity of γ in $\mathcal{T}_{\text{fin}}^{\Sigma_{L\&F}}$. Obviously arbitrary existential sentences may be represented as disjunctions of list constraint systems.

Corollary 3.2 *The existential theory of the algebra $\mathcal{T}_{\text{fin}}^{\Sigma_{L\&F}}$ is decidable.*

To establish Theorem 3.1 we shall give an algorithm that decomposes a flat nontrivial list constraint system $\Gamma = \Gamma_0$ into a finite set of output pairs. We shall see that Γ_0 is solvable iff both components of an output pair are solvable. Moreover, solvability of both output components will be decidable. Before we describe the steps of the algorithm we shall explain the nature of three types of constraint systems that arise from decomposition. With $T(\Omega, X)$ we denote the set of all terms with variables in X and function symbols in Ω. A *VC-declaration* (VC stands for variable-constant) is a pair (Z_V, Z_C) representing a partition $Z = Z_V \dot\cup Z_C$ of a finite set of variables $Z \subset X$. In the presence of a VC-declaration (Z_V, Z_C), the variables in Z_C are not instantiated in solutions, which means that they are treated as constants.

Free disunification problems with linear constant restriction

A *free disunification problem with linear constant restriction* is a quadrupel $(\Gamma_F, Z_V, Z_C, <)$ where

- (Z_V, Z_C) is a VC-declaration of $Z \subseteq X$,
- Γ_F is a finite set of equations and disequations between terms in $T(\Sigma_F \cup Z_C, Z_V)$ and
- $<$ is a linear ordering on Z.

The first component of each output pair has this complex form. A *solution* of this problem is a $T(\Sigma_F, X)$-substitution σ, not instantiating "constants" in Z_C, which solves all equations and disequations of Γ_F such that $y \in Z_C$ does not occur in x^σ for all $x < y$ $(x \in Z_V)$. A solution σ is called *restrictive* if $x^\sigma \notin X$ for all $x \in Z_V$.

The notion of a disunification problem with linear constant restriction and the notion of a restrictive solution have been introduced in [2] in the context of disunification modulo equational theories. There it has been shown (proof of Corollary 4.8):

Lemma 3.3 *It is decidable whether a free disunification problem with linear constant restriction has a restrictive solution.*

Flat pure list constraint systems with linear constant restriction

A *flat pure list constraint system with linear constant restriction* is a quadrupel $(\Gamma_L, Z_V, Z_C, <)$ where

- (Z_V, Z_C) is a VC-declaration of $Z \subseteq X$,
- Γ_L is a finite set of of disequations of the form $x \neq y$ $(x, y \in Z_V)$ and of equations of the form $x = l_1 \circ \ldots \circ l_n$ $(n \geq 0)$ where $x \in Z_V$ and the l_i have the form $z \in Z_V$ or the form $[y]$ with $y \in Z = Z_V \cup Z_C$,
- $<$ is a linear ordering on Z.

Let M be a set. With $\mathcal{L}^M_{\mathrm{nested,fin}}$ we denote the set of all finite, possibly nested lists where elements that are not itself lists are in M. This domain contains only *finite* trees.

A *solution* of $(\Gamma_L, Z_V, Z_C, <)$ is a mapping σ which assigns to every variable $x \in Z_V$ an element $x^\sigma \in \mathcal{L}^X_{\mathrm{nested,fin}}$ such that the canonical extension of σ on pure L-terms[2] solves all equations and disequations of Γ_L and the constant $c \in Z_C$ does not occur in x^σ for all $x < c$ $(x \in Z_V)$. The solution σ is called *compatible with* $<$ if x_1^σ is never a proper subtree of x_2^σ for $x_2 < x_1$ $(x_1, x_2 \in Z_V)$.

In the third step of the algorithm, systems of this type are created. *Nested* lists as solution values may be necessary since variables may occur among the *elements* of lists in equations. This is the important distinction to the following type of system.

Shallow pure list constraint systems with linear constant restriction

Let $(\Gamma_L, Z_V, Z_C, <)$ be a flat pure list constraint system with linear constant restriction. The *shallow version* $(\dot{\Gamma}_L, Z_V, Z_C \cup \dot{Z}_V, \dot{<})$ *of* $(\Gamma_L, Z_V, Z_C, <)$ is obtained by

(1) introducing the new set of constants $\dot{Z}_V := \{\dot{x}; x \in Z_V\}$,

(2) replacing every term $[x]$ in Γ_L with an *embedded* occurrence of a variable $x \in Z_V$ by an expression $[\dot{x}]$,

(3) using the linear ordering $\dot{<}$ which is the extension of $<$ on $Z_V \cup Z_C \cup \dot{Z}_V$ where each constant $\dot{x}$ is the immediate successor of x with respect to $\dot{<}$ $(x \in Z_V)$.

The second components of the output pairs will have this form. The domain $\mathcal{L}^{X \cup \dot{Z}_V}_{\mathrm{flat}}$ contains all lists of the form $[l_1, \ldots, l_n]$ $(n \geq 0)$ with elements $l_i \in X \cup \dot{Z}_V$.

A *solution* of $(\dot{\Gamma}_L, Z_V, Z_C \cup \dot{Z}_V, \dot{<})$ is a mapping σ which assigns to every $x \in Z_V$ a value $x^\sigma \in \mathcal{L}^{X \cup \dot{Z}_V}_{\mathrm{flat}}$ such that the canonical extension[3] of σ on terms in $\dot{\Gamma}_L$ solves all equations and disequations of $\dot{\Gamma}_L$ and the constant $c \in Z_C \cup \dot{Z}_V$ does not occur in x^σ for all $x \dot{<} c$ $(x \in Z_V)$.

Lemma 3.4 *It is decidable whether the shallow version of a flat pure list constraint system with linear constant restriction has a solution.*

[2] Where $c^\sigma := c$ for $c \in Z_C$, $[l_1, \ldots, l_n]^\sigma = [l_1^\sigma, \ldots, l_n^\sigma]$ and $(l_1 \circ l_2)^\sigma$ is the concatenation of l_1^σ and l_2^σ.

[3] Defined as above, with $\dot{x}^\sigma = \dot{x}$ for $\dot{x} \in \dot{Z}_V$. Note that the canonical extension of σ assigns to both sides of each equation of $\dot{\Gamma}_L$ again values in $\mathcal{L}^{X \cup \dot{Z}_V}_{\mathrm{flat}}$ since there are no variables in element positions.

Proof. (Sketch). Suppose that $\dot{\Gamma}_L$ has m disequations. It is first shown that solvability of $(\dot{\Gamma}_L, Z_V, Z_C \cup \dot{Z}_V, \dot{<})$ may be tested in a domain $\mathcal{L}_{\text{flat}}^{X_0 \cup Z_C \cup \dot{Z}_V}$ where $X_0 \subset X$ has $2m+1$ elements and $X_0 \cap Z_C = \emptyset$. Now we have a finite solution alphabet, and the method of Büchi and Senger ([3]) may be used to compute an equivalent finite set of systems with equations only. This latter systems are like word unification problems with linear constant restriction, where solvability is known to be decidable (see [1]). (More details of all steps can be found in [2] where the almost identical case of associative disunification with linear constant restriction has been treated.) □

3.1 First decomposition algorithm (Algorithm 1)

The *input* of Algorithm 1 is a flat and nontrivial list constraint system Γ_0.

Step 1: variable identification. *Consider all partitions of the set of all variables occurring in Γ_0 such that distinct variables x, y are in the same class of the partition if the system contains the equation $x \doteq y$, and distinct variables x, y are in distinct classes of the partition if the system contains the disequation $x \neq y$. Each of these partitions yields one of the new systems Γ_1 as follows. The variables in each class of the partition are "identified" with each other by choosing an element of the class as representative, and replacing in the system all occurrences of variables of the class by this representative. Afterwards, trivial equations $x \doteq x$ are erased. In addition, we add a disequation $x \neq y$ for every pair x, y of distinct representatives to the system if this disequation is not already present. Systems that are trivial now are excluded.*

In each system Γ_1, the right-hand side of every equation is either an F-term or an L-term (but not a variable). We may speak about F-equations and L-equations accordingly.

Step 2: choose ordering, type variables. *For a given system Γ_1, consider all possible strict linear orderings $<$ on the variables of the system. Guess a type assignment which maps every variable x to an element type(x) of $\{F, L\}$, satisfying the following restrictions: if x has type L with respect to Γ_1, or if Γ_1 contains an equation $x \doteq t$ where t is a non-variable L-term (resp. F-term), then type$(x) = L$ (resp. type$(x) = F$). Each pair $(<, \text{type})$ yields one of the new systems obtained from the given one.*

For a system Γ_2 obtained by Step 2, let $X_{3,F}$ ($X_{3,L}$) denote the set of variables of type F (L) occurring in Γ_2. Let $X_2 = X_{3,F} \cup X_{3,L}$. Now left-hand sides of F (L) equations are in $X_{3,F}$ ($X_{3,L}$).

Step 3: split systems. *A given system Γ_2 is divided into two systems $\Gamma_2 = \Gamma_{3,F} \cup \Gamma_{3,L}$. The "free" subsystem $\Gamma_{3,F}$ contains all F-equations of Γ_2, the "L"-subsystem $\Gamma_{3,L}$ contains all L-equations of Γ_2. Disequations with at least one variable of type F are added to the free subsystem, the other disequations are added to $\Gamma_{3,L}$. Now $(\Gamma_{3,F}, X_{3,F}, X_{3,L}, <)$ is a free disunification problem with linear constant restriction and $(\Gamma_{3,L}, X_{3,L}, X_{3,F}, <)$ is a flat pure list constraint system with linear constant restriction.*

Step 4: dot embedded variables. *In this step we compute the shallow version* $(\dot{\Gamma}_{3,L}, X_{3,L}, X_{3,F} \cup \dot{X}_{3,L}, \dot{<})$ *of the flat pure list constraint system with linear constant restriction,* $(\Gamma_{3,L}, X_{3,L}, X_{3,F}, <)$*, obtained in the previous step.*

Terms of $\dot{\Gamma}_{3,L}$ have the form $l_1 \circ \cdots \circ l_m$ $(m \geq 0)$ where the subterms l_i are variables $x \in X_{3,L}$ or lists $[t]$ where $t \in X_{3,F} \cup \dot{X}_{3,L}$ is a constant.

Note that Steps 1 and 2 are non-deterministic. The *output of the algorithm* consists of all pairs

$$((\Gamma_{3,F}, X_{3,F}, X_{3,L}, <), (\dot{\Gamma}_{3,L}, X_{3,L}, X_{3,F} \cup \dot{X}_{3,L}, \dot{<}))$$

which are obtained from Γ_0 by means of the Steps 1 – 4. □

Theorem 3.1 is a direct consequence of the following proposition, using Lemmata 3.3 and 3.4.

Proposition 3.5 *The input system* Γ_0 *has a finite-tree solution if and only if there exists an output pair*

$$((\Gamma_{3,F}, X_{3,F}, X_{3,L}, <), (\dot{\Gamma}_{3,L}, X_{3,L}, X_{3,F} \cup \dot{X}_{3,L}, \dot{<}))$$

such that $(\Gamma_{3,F}, X_{3,F}, X_{3,L}, <)$ *has a restrictive solution and* $(\dot{\Gamma}_{3,L}, X_{3,L}, X_{3,F} \cup \dot{X}_{3,L}, \dot{<})$ *has a solution.*

3.2 Correctness of Algorithm 1

In order to prove Proposition 3.5 we shall prove four subpropositions.

Proposition 3.6 *If the input system* Γ_0 *is solvable, then there exists a pair*

$$((\Gamma_{3,F}, X_{3,F}, X_{3,L}, <), (\Gamma_{3,L}, X_{3,L}, X_{3,F}, <))$$

reached after Step 3 such that $(\Gamma_{3,F}, X_{3,F}, X_{3,L}, <)$ *has a restrictive solution and* $(\Gamma_{3,L}, X_{3,L}, X_{3,F}, <)$ *has a solution that is compatible with* $<$.

Proof. Suppose that σ is a solution of Γ_0. We have to determine choices in the non-deterministic Steps 1 and 2 which lead–after Step 3—to a pair of systems as described in the proposition. In Step 1 of the algorithm two variables x, y are identified iff $x^\sigma = y^\sigma$. With this choice σ is a solution of Γ_1. In Step 2 of the algorithm the linear order $<$ which we choose is an arbitrary extension of the partial order $\prec$ defined by

$$x \prec y :\Leftrightarrow x^\sigma \text{ is a proper subtree of } y^\sigma.$$

A variable x receives type F iff the topmost label of x^σ is in Σ_F. These choices are consistent with the restrictions in Steps 1 and 2 and define a pair of systems $((\Gamma_{3,F}, X_{3,F}, X_{3,L}, <), (\Gamma_{3,L}, X_{3,L}, X_{3,F}, <))$ which is reached after Step 3. We have to show that these systems have solutions as described in the proposition.

Let $\beta : \mathcal{T}_{\text{fin}}^{\Sigma_{L\&F}} \longrightarrow X_2 \cup Y$ be a bijection such that $\beta(x^\sigma) = x$ for all $x \in X_2$. Here $Y \subset X$ is a set of variables that is disjoint to X_2. Since $x_1^\sigma \neq x_2^\sigma$ for all $x_1, x_2 \in X_2$ with $x_1 \neq x_2$ such a bijection exists. Now β defines two projections $\pi_F : \mathcal{T}_{\text{fin}}^{\Sigma_{L\&F}} \longrightarrow T(\Sigma_F \cup X_{3,L}, Y)$ and $\pi_L : \mathcal{T}_{\text{fin}}^{\Sigma_{L\&F}} \longrightarrow \mathcal{L}_{\text{nested,fin}}^{X_{3,F} \cup Y}$ as follows:

$$\begin{aligned}
\pi_F(f(t_1, \ldots, t_n)) &= f(\pi_F(t_1), \ldots, \pi_F(t_1)) \qquad (f \in \Sigma_F) \\
\pi_L(f(t_1, \ldots, t_n)) &= \beta(f(t_1, \ldots, t_n)) \qquad (f \in \Sigma_F) \\
\pi_F([l_1 \ldots l_n]) &= \beta([l_1 \ldots l_n]) \quad (n \geq 0) \\
\pi_L([l_1 \ldots l_n]) &= [\pi_L(l_1), \ldots, \pi_L(l_n)] \quad (n \geq 0).
\end{aligned}$$

Note that our decision concerning variable typing guarantees that the projections have ranges as stated above. We define the assignments $\sigma_F : x \mapsto \pi_F(x^\sigma)$ $(x \in X_{3,F})$ and $\sigma_L : x \mapsto \pi_L(x^\sigma)$ $(x \in X_{3,L})$ and claim that σ_F solves the free system and σ_L solves the L-system.

Let $x \doteq t$ be an equation of $\Gamma_{3,F}$. We know that $x^\sigma = t^\sigma$. Since t has only function symbols from Σ_F and since $\pi_F(y^\sigma) = \beta(y^\sigma) = y$ for all $y \in X_{3,L}$ the last equality in

$$x^{\sigma_F} = \pi_F(x^\sigma) = \pi_F(t^\sigma) = t^{\sigma_F}$$

holds. Thus σ_F solves all equations of $\Gamma_{3,F}$. Let $x \doteq l_1 \circ \cdots \circ l_n$ $(n \geq 0)$ be an equation of $\Gamma_{3,L}$. The l_i are variables in $X_{3,L}$ or lists $[t]$ where $t \in X_2$. We know that $x^\sigma = (l_1 \circ \cdots \circ l_n)^\sigma$. Since $\pi_L(y^\sigma) = \beta(y^\sigma) = y$ for all $y \in X_{3,F}$ we get $x^{\sigma_L} = \pi_L(x^\sigma) = \pi_L((l_1 \circ \cdots \circ l_n)^\sigma) = \pi_L(l_1^\sigma) \circ_T \cdots \circ_T \pi_L(l_n^\sigma) = (l_1 \circ \cdots \circ l_n)^{\sigma_L}$. Thus σ_L solves all equations of $\Gamma_{3,L}$.

Let $x_1 \neq x_2$ be a disequation of $\Gamma_{3,F}$. At least one variable has type F. We have $x_1^\sigma \neq x_2^\sigma$. The inequality $\pi_F(x_1^\sigma) \neq \pi_F(x_2^\sigma)$ follows immediately if both variables have distinct type since in this case exactly one side is a variable. But the same inequality holds also if both variables have type F since β is a bijection, and since x_i^σ does not contain variables $(i = 1, 2)$. It follows that σ_F solves all disequations of $\Gamma_{3,F}$. Similarly it follows that σ_L solves all disequations of $\Gamma_{3,L}$.

Let us now consider the linear constant restriction of the free subsystem. If $x_2 \in X_{3,L}$ occurs in $x_1^{\sigma_F} = \pi_F(x_1^\sigma)$ $(x_1 \in X_{3,F})$, then this occurrence is necessarily the result of projecting an occurrence of x_2^σ in x_1^σ since x_1^σ does not contain variables. Thus x_2^σ is a proper subtree of x_1^σ and $x_2 < x_1$. This shows that σ_F satisfies the constant restriction of the free subsystem. Similarly it follows that σ_L satisfies the constant restriction of the L-subsystem. Since the π_F-projection of a tree with topmost function symbol in Σ_F cannot be a variable it is clear that σ_F is a restrictive solution. It remains to be shown that σ_L is compatible with $<$. Suppose that $x_1^{\sigma_L}$ is a proper subterm of $x_2^{\sigma_L}$ $(x_1, x_2 \in X_{3,L})$. Thus $\pi_L(x_1^\sigma)$ is a proper subterm of $\pi_L(x_2^\sigma)$. The inverse β^{-1} of β may be considered as a substitution. It follows that $x_1^\sigma = (\pi_L(x_1^\sigma))^{\beta^{-1}}$ is a proper subterm of $x_2^\sigma = (\pi_L(x_2^\sigma))^{\beta^{-1}}$ and therefore $x_1 < x_2$. Thus σ_L is in fact compatible with $<$. □

Proposition 3.7 *If a system $(\Gamma_{3,L}, X_{3,L}, X_{3,F}, <)$ obtained as second component after Step 3 has a solution σ that is compatible with $<$, then the dotted system reached after Step 4, $(\dot{\Gamma}_{3,L}, X_{3,L}, X_{3,F} \cup \dot{X}_{3,L}, \dot{<})$, has a solution.*

The proof is similar as the previous one. Compatibility of σ with $<$ is needed to be able to satisfy the linear constant restrictions of the dotted system that are associated with dotted variables. Summarizing, the preceding two propositions show that the decomposition algorithm is complete. Let us now consider soundness.

Proposition 3.8 *If a dotted system $(\dot\Gamma_{3,L}, X_{3,L}, X_{3,F} \cup \dot X_{3,L}, \dot<)$ obtained after Step 4 has a solution, then the original system $(\Gamma_{3,L}, X_{3,L}, X_{3,F}, <)$ has a solution.*

Proof. Let $\dot\sigma$ be a solution of $(\dot\Gamma_{3,L}, X_{3,L}, X_{3,F} \cup \dot X_{3,L}, \dot<)$. We may assume that

$$\dot\sigma : X_{3,L} \to \mathcal{L}_{\mathrm{flat}}^{Y \cup X_{3,F} \cup \dot X_{3,L}}$$

where $Y \subseteq X$ is disjoint to X_2. We shall now use the linear order $\dot<$ in order to define a partial assignment $\sigma : X_{3,L} \cup \dot X_{3,L} \to \mathcal{L}_{\mathrm{nested,fin}}^{Y \cup X_{3,F}}$ such that the restriction on $X_{3,L}$—extended canonically on pure L-terms—solves $(\Gamma_{3,L}, X_{3,L}, X_{3,F}, <)$. Let $x \in X_{3,L} \cup \dot X_{3,L}$ and assume that z^σ has been defined for all $z \in X_{3,L} \cup \dot X_{3,L}$ such that $z \dot< x$. We shall also assume (*) that $z_1^\sigma \neq z_2^\sigma$ for all $z_1, z_2 \in X_{3,L}$ with $x > z_1 \neq z_2 < x$.

If $x = \dot z$ is a dotted variable, then x is the immediate successor of z. We define $x^\sigma := z^\sigma$. If $x \in X_{3,L}$, then the dotted elements of the flat list $x^{\dot\sigma}$ are smaller than x with respect to $\dot<$. We define $x^\sigma := x^{\dot\sigma\sigma}$. Since the flat lists $x^{\dot\sigma}$ and $z^{\dot\sigma}$ are distinct it follows easily, by (*), that $x^\sigma \neq z^\sigma$ for all $z < x$, $z \in X_{3,L}$.

We shall now prove that σ is a solution of $(\Gamma_{3,L}, X_{3,L}, X_{3,F}, <)$. Let $x \doteq l_1 \circ \cdots \circ l_n$ be an equation of $\Gamma_{3,L}$ with counterpart $x \doteq l_1' \circ \cdots \circ l_n'$ in $\dot\Gamma_{3,L}$. We have $x^{\dot\sigma} = (l_1' \circ \cdots \circ l_n')^{\dot\sigma}$. Therefore $x^\sigma = x^{\dot\sigma\sigma} = (l_1' \circ \cdots \circ l_n')^{\dot\sigma\sigma}$. Each l_i' is in $X_{3,L}$ or it has the form $[t]$ where $t \in \dot X_{3,L} \cup X_{3,F}$. It follows easily that $(l_1' \circ \cdots \circ l_n')^{\dot\sigma\sigma} = (l_1 \circ \cdots \circ l_n)^\sigma$, thus σ solves $x \doteq (l_1 \circ \cdots \circ l_n)$. $\Gamma_{3,L}$ contains only disequations where both variables have type L. We have already seen that σ solves these disequations. Let us consider the linear constant restriction which is imposed by $<$. Let $z \in X_{3,F}$, $z > x \in X_{3,L}$. We know that z does not occur in any term of the form $r^{\dot\sigma}$ for $r \dot\leq x$, $r \in X_{3,L}$. From this it follows easily that z does not occur in x^σ. □

Proposition 3.9 *If there exists a pair $((\Gamma_{3,F}, X_{3,F}, X_{3,L}, <), (\Gamma_{3,L}, X_{3,L}, X_{3,F}, <))$ reached after Step 3 such that $(\Gamma_{3,F}, X_{3,F}, X_{3,L}, <)$ has a restrictive solution and $(\Gamma_{3,L}, X_{3,L}, X_{3,F}, <)$ has a solution, then Γ_0 has a solution.*

Proof. Let σ_F be a restrictive solution of the free disunification problem with linear constant restriction $(\Gamma_{3,F}, X_{3,F}, X_{3,L}, <)$, let σ_L be a solution of $(\Gamma_{3,L}, X_{3,L}, X_{3,F}, <)$. We may assume that

$$\begin{aligned} \sigma_F :\ X_{3,F} &\ \to\ T^(\Sigma_F \cup X_{3,L}, Y_F) \\ \sigma_L :\ X_{3,L} &\ \to\ \mathcal{L}_{\mathrm{nested,fin}}^{X_{3,F} \cup Y_L} \end{aligned}$$

where the sets $Y_F = \{y_{1,F}, \ldots, y_{m,F} \subseteq X$ and $Y_L = \{y_{1,L}, \ldots, y_{n,L}\} \subseteq X$ are finite, disjoint and do not contain an element of $X_{3,F} \cup X_{3,L} \cup \dot X_{3,L}$. Since Σ_F contains at least one constant and one non-constant function symbol we may choose n distinct ground terms $t_1, \ldots, t_n$ over this signature which are different from all terms x^{σ_F} for $x \in X_{3,F}$. Similarly we may choose m distinct nested lists $l_1, \ldots, l_m$ where all labels have the form $[]^k$ $(k \geq 0)$, each list l_i being distinct from all lists x^{σ_L} for $x \in X_{3,L}$. Let

$$\begin{array}{rcll} \tau_F : y_{i,F} & \mapsto & l_i & (1 \leq i \leq m), \\ \tau_L : y_{i,L} & \mapsto & t_i & (1 \leq i \leq n). \end{array}$$

We shall define a $\mathcal{T}_{\text{fin}}^{\Sigma_{L\&F}}$-assignment σ on X_2 by induction on the linear ordering $<$. Assume that z^σ has been defined for all $z \in X_2$ preceding $x \in X_2$ with respect to $<$. We shall assume (1) that this assignment is type-conform, which means that z^σ has topmost symbol in Σ_F (of the form $[\ ]^k$) for variables z of type F (type L), (2) that $z_1^\sigma \neq z_2^\sigma$ for all $z_1, z_2 < x$, and (3) that the terms z^σ are not in $\{t_1, \ldots, t_n, l_1, \ldots, l_m\}$ for $z < x$.

Assume that x has type $i \in \{F, L\}$, let $i \neq j \in \{F, L\}$. Since σ_i respects the linear constant restriction of system i, the variables occurring in x^{σ_i} are variables $z \in X_{3,j}$ with $z < x$, or variables from Y_i. Thus we may define $x^\sigma := x^{\sigma_i \tau_i \sigma}$. By induction hypothesis, $z^\sigma \in \mathcal{T}_{\text{fin}}^{\Sigma_{L\&F}}$ for all $x > z \in X_{3,j}$, thus $x^\sigma \in \mathcal{T}_{\text{fin}}^{\Sigma_{L\&F}}$. Since σ_F is restrictive and since σ_L ranges over lists, this assignment is type-conform and assumption (1) holds again. Assume that $x^\sigma = z^\sigma$ for some $z \in X_2$, $z < x$. Then z has type i since σ is type-conform. By assumption (1), the maximal j-subterms of $z^{\sigma_i \tau_i \sigma} = z^\sigma = x^\sigma = x^{\sigma_i \tau_i \sigma}$ are exactly the σ-images of the variables of type j occurring in z^{σ_i} and the τ_i-images of variables $y_{l,i}$. The former variables are smaller than x and the restriction of σ on these variables is injective, by hypothesis. By assumption (3), we obtain z^{σ_i} *and* x^{σ_i} back from $z^\sigma = x^\sigma$ just by a projection which replaces these alien subterms by their unique τ_i- or σ-origines. Thus $x^{\sigma_i} = z^{\sigma_i}$. This is a contradiction since σ_i solves the disequation $x \neq z$. Therefore assumption (2) holds again. If x^{σ_i} contains any variable, then x^σ will have occurrences of free function symbols *and* of a list symbol $[\ ]^k$. Therefore $x^\sigma \notin \{t_1, \ldots, t_n, l_1, \ldots, l_m\}$. If x^{σ_i} is ground, $x^\sigma = x^{\sigma_i} \notin \{t_1, \ldots, t_n, l_1, \ldots, l_m\}$ by choice of the these elements. Therefore assumption (3) holds again.

We may now show that σ solves the system Γ_2 which is reached after Step 2. Since σ is consistent for Γ_1 (see (1) and the restrictions in Step 2) it is then clear that σ can be extended to a solution of Γ_0. By our previous considerations it remains to be shown that σ solves the equations $x \doteq t$ of Γ_2. Assume that $x \doteq t$ is in $\Gamma_{3,i}$, where $i \in \{F, L\}$. Then $x^{\sigma_i} = t^{\sigma_i}$. It follows that $x^\sigma = x^{\sigma_i \tau_i \sigma} = t^{\sigma_i \tau_i \sigma} = t^\sigma$. For the last equality recall that σ_i and τ_i leave all $y \in X_{3,j}$ fixed while $y^{\sigma_i \tau_i \sigma} = y^\sigma$ for $y \in X_{3,i}$. □

4 Results for Rational-Tree Solutions

Here we want to prove the following theorem.

Theorem 4.1 *It is decidable if an equational list constraint system Γ has a rational-tree solution.*

Corollary 4.2 *The positive existential theory of the algebra $\mathcal{T}_{\text{rat}}^{\Sigma_{L\&F}}$ is decidable.*

Before we give a second algorithm for proving these results it is instructive to reconsider Algorithm 1 and its soundness proof: we found that given solutions of the two components of an output pair can be combined to yield a solution of the input system. This solution is found by a *finite* recursive process along the chosen linear ordering. The linear constant restrictions imposed on the components of the output pairs have the effect of a partial occur check, excluding cyclic dependencies between values of F- and L-variables. If we now ask for rational-tree solutions, cyclic dependencies are acceptable and may be necessary. Accordingly, constant restrictions are not used in Algorithm 2.

4.1 Second decomposition algorithm (Algorithm 2)

The *input* is a flat and nontrivial constraint system Γ_0 without disequations. Algorithm 2 is obtained as a simplification of Algorithm 1:

- We ignore all subparts in the description of the steps of Algorithm 1 that refer to disequations.
- In Step 2 (type variables) we do *not* choose a linear order on the variables. Accordingly, the systems obtained after Step 3 have the form $(\Gamma_{3,F}, X_{3,F}, X_{3,L})$ and $(\Gamma_{3,L}, X_{3,L}, X_{3,F})$, and from $(\Gamma_{3,L}, X_{3,L}, X_{3,F})$ we obtain its shallow version $(\dot{\Gamma}_{3,L}, X_{3,L}, X_{3,F} \cup \dot{X}_{3,L})$[4] in Step 4.

The *output* consists of all pairs $((\Gamma_{3,F}, X_{3,F}, X_{3,L}), (\dot{\Gamma}_{3,L}, X_{3,L}, X_{3,F} \cup \dot{X}_{3,L}))$ that are obtained from Γ_0 by means of the new Steps 1 – 4. $\square$

The simplification of the decomposition steps comes in parallel with a modification of the solution domains for the systems that are reached. The free system $(\Gamma_{3,F}, X_{3,F}, X_{3,L})$ is solved in the algebra $\mathcal{T}_{\mathrm{rat}}^{\Sigma_F \cup X_{3,L} \cup Y}$ of rational trees with labels in $\Sigma_F \cup X_{3,L} \cup Y$, treating $X_{3,L}$ as a set of constants. Here $Y \subset X$ is an infinite set of variables that is disjoint to X_2. Solvability of equational systems over $\mathcal{T}_{\mathrm{rat}}^{\Sigma_F \cup X_{3,L} \cup Y}$ is decidable (see [4, 6]). Since Σ_F contains a constant and a non-constant function symbol, solvability and restrictive solvability are equivalent.

Corollary 4.3 *It is decidable if a system* $(\Gamma_{3,F}, X_{3,F}, X_{3,L})$ *has a restrictive solution.*

System $(\Gamma_{3,L}, X_{3,L}, X_{3,F})$ is solved—treating $X_{3,F}$ as a set of constants—in the domain $\mathcal{L}_{\mathrm{nested,rat}}^{X_{3,F} \cup Y}$ of nested lists representing rational trees with labels in $\Sigma_L \cup X_{3,F} \cup Y$. System $(\dot{\Gamma}_{3,L}, X_{3,L}, X_{3,F} \cup \dot{X}_{3,L})$ is solved in $\mathcal{L}_{\mathrm{flat}}^{X \cup \dot{X}_{3,L}}$, as earlier.

Theorem 4.1 is a direct consequence of the following proposition, using Corollary 4.3 and the fact that solvability of word equations is decidable ([7]).

[4]Defined as earlier, ignoring linear orders.

Proposition 4.4 *The input* Γ_0 *of Algorithm 2 has a rational-tree solution if and only if there exists an output pair* $((\Gamma_{3,F}, X_{3,F}, X_{3,L}), (\dot{\Gamma}_{3,L}, X_{3,L}, X_{3,F} \cup \dot{X}_{3,L}))$ *such that* $(\Gamma_{3,F}, X_{3,F}, X_{3,L})$ *has a restrictive solution and* $(\dot{\Gamma}_{3,L}, X_{3,L}, X_{3,F} \cup \dot{X}_{3,L})$ *has a solution.*

4.2 Correctness of Algorithm 2

Completeness of Algorithm 2 is proved in similar manner as for Algorithm 1. We omit this part. For proving soundness let us introduce the following notation: we write $t_1 =_i t_2$ if the rational trees t_1 and t_2 have the same labels for all positions of length (depth) $k \leq i$. Clearly $t_1 = t_2$ iff $t_1 =_i t_2$ for all $i \geq 0$.

Proposition 4.5 *If a dotted system* $(\dot{\Gamma}_{3,L}, X_{3,L}, X_{3,F} \cup \dot{X}_{3,L})$ *obtained after Step 4 has a solution, then the original system* $(\Gamma_{3,L}, X_{3,L}, X_{3,F})$ *has a solution.*

Proof. Let $\dot{\sigma}$ be a solution of $(\dot{\Gamma}_{3,L}, X_{3,L}, X_{3,F} \cup \dot{X}_{3,L})$. We may assume that

$$\dot{\sigma} : X_{3,L} \to \mathcal{L}_{\text{flat}}^{Y \cup X_{3,F} \cup \dot{X}_{3,L}}$$

where $Y \subseteq X$ and X_2 are disjoint. Let τ be the assignment which maps every dotted variable $\dot{x} \in \dot{X}_{3,L}$ to $\dot{x}^{\tau} := x^{\dot{\sigma}}$. Let $\sigma_i := \dot{\sigma} \circ \tau^i$ $(i \geq 1)$. Obviously $x^{\sigma_i} =_k x^{\sigma_j}$ for all $i, j \geq k$ and $x \in X_{3,L}$. There exists a unique tree $t_x \in \mathcal{L}_{\text{nested,rat}}^{X_{3,F} \cup Y}$ such that $x^{\sigma_i} =_k t_x$ for all $i \geq k \geq 1$. We define $x^{\sigma} := t_x$ $(x \in X_{3,L})$. Take an equation $x \doteq l_1 \circ \cdots \circ l_n$ of $\Gamma_{3,L}$ with counterpart $x \doteq l'_1 \circ \cdots \circ l'_n$ in $\dot{\Gamma}_{3,L}$. Let $i > 1$. For $l_j = l'_j = y \in X_{3,L}$ we have $l'^{\sigma_i}_j =_i l^{\sigma}_j$. For $l_j = l'_j = [y]$ with $y \in X_{3,F}$ we have $l'^{\sigma_i}_j = l'_j = l^{\sigma}_j$. For $l'_j = [\dot{y}]$ with $y \in X_{3,L}$ we have $l'^{\sigma_i}_j = [\dot{y}^{\sigma_i}] = [\dot{y}^{\dot{\sigma} \circ \tau^i}] = [\dot{y}^{\tau^i}] = [y^{\dot{\sigma} \tau^{i-1}}] = [y^{\sigma_{i-1}}] =_i [y]^{\sigma} = l^{\sigma}_j$. Thus $x^{\sigma} =_i x^{\sigma_i} = (l'_1 \circ \cdots \circ l'_n)^{\sigma_i} =_i (l_1 \circ \cdots \circ l_n)^{\sigma}$ for $i > 1$ and σ solves $x \doteq l_1 \circ \cdots \circ l_n$. $\square$

Proposition 4.6 *If there exists a pair* $((\Gamma_{3,F}, X_{3,F}, X_{3,L}), (\Gamma_{3,L}, X_{3,L}, X_{3,F}))$ *reached after Step 3 such that* $(\Gamma_{3,F}, X_{3,F}, X_{3,L})$ *has a restrictive solution and* $(\Gamma_{3,L}, X_{3,L}, X_{3,F})$ *has a solution, then* Γ_0 *has a solution.*

Proof. Let σ_F be a restrictive solution of $(\Gamma_{3,F}, X_{3,F}, X_{3,L})$ and let σ_L be a solution of $(\Gamma_{3,L}, X_{3,L}, X_{3,F})$. We may assume that

$$\begin{aligned} \sigma_F : \; X_{3,F} &\to \mathcal{T}_{\text{rat}}^{\Sigma_F \cup X_{3,L} \cup Y} \\ \sigma_L : \; X_{3,L} &\to \mathcal{L}_{\text{nested,rat}}^{X_{3,F} \cup Y} \end{aligned}$$

where $Y = \{y_1, \ldots, y_n\} \subset X$ is finite and $Y \cap X_2 = \emptyset$. Let $\sigma_{F\&L} := \sigma_F \cup \sigma_L$. Choose n distinct ground trees $t_1, \ldots, t_n \in \mathcal{T}_{\text{rat}}^{\Sigma_{L\&F}}$. Let $\tau : y_i \mapsto t_i$ $(1 \leq i \leq n)$. We identify both $\sigma_{F\&L}$ and τ with their homomorphic extension on $\mathcal{T}_{\text{rat}}^{\Sigma_{L\&F} \cup X_2 \cup Y}$. Let $\sigma_1 := \sigma_{F\&L} \cup \tau$, and let $\sigma_i := \sigma^i$ $(i \geq 1)$. Since σ_F is restrictive and each list x^{σ_L} $(x \in X_{3,L})$ has topmost label of the form $[\;]^k$ we know that $x^{\sigma_i} =_k x^{\sigma_j}$ for all $i, j \geq k$ $(x \in X_2)$. There exists a unique tree $t_x \in \mathcal{T}_{\text{rat}}^{\Sigma_{L\&F}}$ such that $x^{\sigma_i} =_k t_x$ for all $1 \leq k \leq i$. We define $x^{\sigma} := t_x$. The restrictions in Step 2 of the algorithm guarantee that σ is consistent for Γ_1.

Let $i > 1$. Consider an F-equation $x \doteq f(y_1, \ldots, y_n)$ of the system reached after Step 1, Γ_1. If $y_j \in X_{3,F}$, then $y_j^{\sigma_F \tau \sigma_{i-1}} = y_j^{\sigma_i} =_{i-1} y_j^{\sigma_{i-1}}$. If $y_j \in X_{1,L}$, then $y_j^{\sigma_F \tau \sigma_{i-1}} = y_j^{\sigma_{i-1}}$. Thus

$$\begin{aligned} x^\sigma &=_i \; x^{\sigma_i} = x^{\sigma_F \tau \sigma_{i-1}} = f(y_1, \ldots, y_n)^{\sigma_F \tau \sigma_{i-1}} \\ &=_i \; f(y_1, \ldots, y_n)^{\sigma_{i-1}} =_i f(y_1, \ldots, y_n)^\sigma. \end{aligned}$$

Therefore σ solves $x \doteq f(y_1, \ldots, y_n)$. Consider an L-equation $x \doteq l_1 \circ \cdots \circ l_n$ $(n \geq 0)$ of Γ_1. If $l_j = y \in X_{3,L}$, then $l_j^{\sigma_L \tau \sigma_{i-1}} = l_j^{\sigma_i} =_{i-1} l_j^{\sigma_{i-1}}$. Similarly $l_j^{\sigma_L \tau \sigma_{i-1}} =_{i-1} l_j^{\sigma_{i-1}}$ for $l_j = [y]$ with $y \in X_{3,L}$. If $l_j = [y]$ where $y \in X_{3,F}$, then $l_j^{\sigma_L \tau \sigma_{i-1}} = l_j^{\sigma_{i-1}}$. Thus

$$\begin{aligned} x^\sigma &=_i \; x^{\sigma_i} = x^{\sigma_L \tau \sigma_{i-1}} = (l_1 \circ \cdots \circ l_n)^{\sigma_L \tau \sigma_{i-1}} \\ &=_{i-1} \; (l_1 \circ \cdots \circ l_n)^{\sigma_{i-1}} =_{i-1} (l_1 \circ \cdots \circ l_n)^\sigma \end{aligned}$$

and σ solves $x \doteq l_1 \circ \cdots \circ l_n$. This shows that σ solves all equations of Γ_1. Thus σ is a solution of Γ_1. It is now trivial to extend σ to a solution of Γ_0. $\square$

4.3 Problems with Disequations

Unfortunately, the treatment of disequations causes problems when we ask for rational-tree solutions. Here is an illustrating example. The input system Γ_0 with equations $x_1 \doteq g(y_1), x_2 \doteq g(y_2), y_1 \doteq [x_1], y_2 \doteq [x_2]$ and disequation $x_1 \neq x_2$ cannot be solved in $\mathcal{T}_{\text{rat}}^{\Sigma_{L\&F}}$ since every solution of the equational part will identify x_1 and x_2. If we decompose Γ_0, treating disequations as in Algorithm 1, one particular output pair with free system $\Gamma_{3,F} = \{x_1 \doteq g(y_1), x_2 \doteq g(y_2), x_1 \neq x_2\} \cup \{x_i \neq y_j; i, j = 1, 2\}$ (constants y_1, y_2) and with L-component $\Gamma_{3,L} = \{y_1 \doteq [x_1], y_2 \doteq [x_2], y_1 \neq y_2\}$ (constants x_1, x_2) is generated. Both systems are solved. Thus decomposition is no longer sound. The reason is that validity of disequations is not preserved when we recombine solutions of the output systems in order to obtain a solution of Γ_0.

Our attempts to prove decidability of the full existential theory of $\mathcal{T}_{\text{rat}}^{\Sigma_{L\&F}}$ have led to a partial result only. The question can be reduced to the following problem for word equations: given a word equation with variables $x_1, \ldots x_n$, and given a finite set of constraints of the form $|x_i| = |x_j|$ demanding that the length of the (words to be substituted for the) variables x_i and x_j has to be the same, decide if the word equation has a solution that satisfies these restrictions. The first reduction step is based on the following observation.

Theorem 4.7 [5] *If a typed flat constraint system Γ has a solution, then the system $\Gamma_{\chi(\Gamma)}$ has a solution that is obtained from Γ by replacing every disequation $x \neq y$ of Γ by a bounded disequation $x \neq_{\chi(\Gamma)} y$. Here $\chi(\Gamma) = n_{\text{emb}}^2 + n_{\text{dis}} + 1$ where n_{emb} is the number of embedded variables of Γ and n_{dis} is the number of disequations of Γ.*

[5] A list constraint system Γ is typed if every variable occurring in Γ has type F or type L. A tree assignment σ solves a bounded disequation $x \neq_k y$ if the trees x^σ and y^σ have a distinct label in depth $j \leq k$. An occurrence of a variable x in a term of t of Γ of the form $[x]$ or $f(\ldots, x, \ldots)$ $(f \in \Sigma_F)$ is called an embedded occurrence of x in Γ.

In a second step, bounded disequations can be eliminated for the price of introducing length constraints of the form $|x| = |y|$ and $|x| > |y|$ that restrict the length of (the values of) L-variables. For each system Δ with equations and bounded disequations we obtain a finite number of systems $\Delta_1, \ldots, \Delta_r$ with length constraints, preserving solvability in both directions. Eventually a variant of Algorithm 2 may be used to decompose the systems Δ_i in a similar way as before, taking length constraints into account. While the free output components take the same form as in the case of Algorithm 2, the L-output systems may be considered as word equations with length constraints as described above. On the level of word equations it suffices to have length constraints of the form $|x| = |y|$.

The remarks given in the introduction indicate that the problem to decide solvability of word equations with length constraints might be extremely difficult.

References

[1] F. Baader and K.U. Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. In *Proceedings of CADE-11*, Springer LNCS 607, 1992.

[2] F. Baader and K.U. Schulz, "Combination Techniques and Decision Problems for Disunification," (extended version) DFKI Research-Report-93-05, German Research Center for AI, Saarbrücken 1993. Short version in *Proceedings RTA '93, Montreal, June 1993*, LNCS. Springer, 1993, pp.301-315.

[3] J.R. Büchi, S. Senger, "Coding in the Existential Theory of Concatenation," Arch. math. Logik **26** (1986/7), pp.101-106.

[4] A. Colmerauer, "Equations and Inequations on Finite and Infinite Trees," *Proceedings of the FGCS'84*, pp.85-99.

[5] A. Kościelski, L. Pacholski, "Complexity of Makanin's Algorithms," Research Report, Universitiy of Wroclaw (1991); preliminary version: "Complexity of Unification in Free Groups and Free Semi-Groups," Proceedings 31st Annual IEEE Symposium on Foundations of Computer Science, Los Alamos (1990), pp.824-829.

[6] M.J. Maher, "Complete axiomatizations of the algebras of finite, rational and infinite trees", In *Proc. LICS 3*, IEEE Computer Society (1988), pp.348-357.

[7] G.S. Makanin, "The Problem of Solvability of Equations in a Free Semigroup," *Mat. USSR Sbornik* **32**, 1977.

[8] G.S. Makanin, "Equations in a Free Group", Izv. Akad. Nauk SSSR Ser. Mat. 46 (1982), 1199-1273; English Translation in Math. USSR Izv. 21 (1983).

[9] W.V. Quine, "Concatenation as a Basis for Arithmetic," J. Symbolic Logic 11 (1946), pp.105-114.