Model Completeness for Rational Trees*

Silvio Ghilardi[1](✉)[0000−0001−6449−6883] and Lia M. Poidomani[1][0009−0003−7769−328X]

[1]Dipartimento di Matematica, Università degli Studi di Milano (Italy)
silvio.ghilardi@unimi.it
http://users.mat.unimi.it/users/ghilardi/
lia.poidomani@hotmail.it

Abstract. We analyze the theory of rational trees with finitely many constructors, infinitely many atoms and an atomicity predicate. We design a new decision procedure, proving in addition that this theory is model-complete. We also show that the enrichment of the language with selectors and simultaneous parametric fixpoints enjoys quantifier elimination.

Keywords: Infinite Trees · Model Completeness · Decision Procedures

1 Introduction

The theory of finite and infinite trees deserves special attention in computer science applications for its capability of representing both algebraic and co-algebraic datatypes, including for instance (acyclic or even cyclic) lists, streams, etc. The theory has been largely investigated by the logic programming community since its early days [3, 8] and various results have been proved about it, including decidability in the full elementary language [12], albeit within very high (actually non elementary) complexity bounds [21]. The automated reasoning community gave some contributions too, focusing especially on the more restricted constraint solving fragment [20].

Deciding satisfiability of the elementary theory of finite and infinite trees cannot be obtained via quantifier elimination, as pointed out in the comprehensive and detailed paper [4], which improves and extends classical results from [12]. In fact, the normalizations performed by the algorithms proposed in these papers prove that every formula is equivalent to a Boolean combination of special kinds of primitive formulae (thus giving, in terms of prenex forms, a reduction to a $\Delta_2^0 = \exists^*\forall^* \cap \forall^*\exists^*$-class), but not to a quantifier-free formula. The impossibility of full quantifier-elimination can indeed be shown by easy counterexamples. However, quantifier elimination is a nice and desirable property, for instance it facilitates the computation of interpolants [10], whose employment in verification applications is widely recognized [15].

From the point of view of 'abstract logical nonsense', quantifier elimination can be in principle always obtained, by the trivial enlargement of the language naming every formula by a fresh atomic predicate. More concretely, it often happens that manageable enrichments of the language are sufficient to achieve quantifier elimination: a classical example in this sense is the theory of real closed fields which becomes quantifier-eliminable after adding to the language the ordering predicate (this is easily seen to be

* The first author is supported by INdAM's GNSAGA group.

definable in the theory, because 'being positive' turns out to be equivalent to 'being a square'). A more complicated, but still manageable example is linear integer arithmetic, where in order to obtain quantifier elimination it is sufficient to enrich the language with the infinitely many definable binary predicates expressing 'equivalence modulo n'.

We take into consideration the variant of the theory of trees whose signature has *finitely many constructors, infinitely many constants symbols and an atomicity predicate*. We feel that this variant of the theory is rather natural to consider, especially in verification applications, where constants can represent infinite data (maybe constrained by richer theories describing their internal structure). The problem we are addressing in this paper is: what is the price to pay (in terms of language enrichments) to achieve quantifier elimination for the above theory? In [14] quantifier elimination is obtained by introducing *ad hoc* syntactic entities (called 'terms with pointers') and applying to them an extension of the classical Mal'cev quantifier elimination algorithm for finite trees [13]. In this paper we stay inside the native first order language and we prove that our theory is *model-complete* by introducing a novel technique based on *definability analysis*; as a by-product, we achieve full quantifier elimination by enriching the language with extra operation symbols for *selectors* and *simultaneous parametric fixpoints*.

The paper is structured as follows: Sections 2, 3 introduce preliminary definitions, our variant of the theory of trees and establish some basic properties; Section 4 gives a new decision procedure for constraint solving problems using *graph representations, bisimulations and congruence closure*. Section 5 introduces our main technical tool, namely *definability in existential formulae*; Section 6 proves first model-completeness and then shows how to enrich the language to achieve full quantifier elimination. Section 7 concludes and discusses further improvements. The paper is meant to be self-contained; we moved to the online available extended version

`http://users.mat.unimi.it/users/ghilardi/allegati/GP_IJCAR24.pdf`

the proofs of few results which are either well-known or rather straightforward. Such extended version contains also a thorough comparison with the approach of some relevant papers from the literature like [4] and [14].

2 Preliminaries

We adopt the usual first-order syntactic notions of signature, term, atom, literal, (ground) formula, and so on; our signatures always include equality. We compactly represent a tuple of distinct variables as $\underline{x}$. The notation $t(\underline{x}), \phi(\underline{x})$ means that the term t, the formula ϕ has free variables included in the tuple $\underline{x}$. Since our *tuples of variables* are assumed to be formed by *distinct* elements, we underline that when we write e.g. $\phi(\underline{x},\underline{y})$, we mean that the tuples $\underline{x},\underline{y}$ are made of distinct variables and are also disjoint from each other. Notations like $\phi(\underline{t}/\underline{x})$ are used to denote substitutions.

A formula is said to be *universal* (resp., *existential*) if it has the form $\forall\underline{x}\,\phi(\underline{x},\underline{y})$ (resp., $\exists\underline{x}\,\phi(\underline{x},\underline{y})$), where ϕ is quantifier-free. Formulae with no free variables are called *sentences*. A *constraint* is a conjunction of literals; a *primitive* formula is obtained from a constraint by prefixing it a finite string of existential quantifiers.

From the semantic side, given a signature Σ, we use the standard notion of a Σ-structure $\mathcal{M}$ and of truth of a formula in a Σ-structure under a free variables assignment. A *Σ-theory* $\mathcal{T}$ is a set of Σ-sentences; a *model* of $\mathcal{T}$ is a Σ-structure $\mathcal{M}$ where all sentences in $\mathcal{T}$ are true. We use the standard notation $\mathcal{T} \models \phi$ to say that ϕ is *$\mathcal{T}$-valid*, i.e. true in all models of $\mathcal{T}$ for every assignment to the variables occurring free in ϕ. We say that two formulae ϕ and ψ are $\mathcal{T}$-equivalent iff $\phi \leftrightarrow \psi$ is $\mathcal{T}$-valid. A theory $\mathcal{T}$ is *complete* iff for every sentence ϕ, either ϕ or $\neg\phi$ is $\mathcal{T}$-valid. Complete theories can be obtained from any Σ-structure $\mathcal{M}$ by taking the set of the Σ-sentences that are true in $\mathcal{M}$ (such a theory is denoted by $Th(\mathcal{M})$).

We say that ϕ is *$\mathcal{T}$-satisfiable* iff there is a model $\mathcal{M}$ of $\mathcal{T}$ and an assignment to the variables occurring free in ϕ making ϕ true in $\mathcal{M}$. The *elementary* (resp. *constraint*) *satisfiability problem* for $\mathcal{T}$ is the following: we are given a formula (resp. a constraint) $\phi(\underline{x})$ and we are asked whether there exist a model $\mathcal{M}$ of $\mathcal{T}$ and an assignment α to the free variables $\underline{x}$ such that $\mathcal{M}, \alpha \models \phi(\underline{x})$ holds, i.e. such that ϕ is true in $\mathcal{M}$ under such assignment.

A theory $\mathcal{T}$ has *quantifier elimination* iff for every formula $\phi(\underline{x})$ in the signature of $\mathcal{T}$ there is a quantifier-free formula $\phi'(\underline{x})$ such that $\mathcal{T} \models \phi(\underline{x}) \leftrightarrow \phi'(\underline{x})$. We shall be interested also into a condition that is weaker than quantifier elimination, namely model-completeness. Such a notion is usually defined semantically (as the fact that an embedding between two models of the theory is elementary), however there is a well-known equivalent syntactic definition that we are going to extensively use (see [2], Thm. 3.5.1). This equivalent definition is supplied by the following:

Definition 1. *A theory $\mathcal{T}$ is said to be* model-complete *iff every existential formula $\exists\underline{x}\,\phi(\underline{x},\underline{y})$ in the signature of $\mathcal{T}$ is $\mathcal{T}$-equivalent to a universal formula $\forall\underline{x}'\,\phi'(\underline{x}',\underline{y})$.*

Notice that, keeping in mind prenex normal forms and the interdefinability of universal and existential quantifiers, it turns out that in a model complete theory *every formula whatsoever* is $\mathcal{T}$-equivalent to *both* a universal and an existential formula; consequently, *model-completeness reduces the elementary satisfiability problem to the constraint satisfiability problem.*

3 Σ-trees

In the whole paper we fix a signature (let us call it Σ) containing: (i) *infinitely many individual constants*, (ii) *finitely many function symbols* $h_1,\ldots,h_N$ of respective arities $ar_1,\ldots,ar_N \geq 1$ and (iii) *a unary predicate* At. Symbols $h_1,\ldots,h_N$ are called *Σ-constructors* or just *constructors*. We use letters $f,g,\ldots$ for constructors or constants of Σ. We now define Σ-labelled trees.

A *tree* T is a set of finite lists of positive natural numbers satisfying the following three conditions; (1) the empty list ε belongs to T; (2) T is prefix-closed, i.e $\sigma * \tau \in T$ implies $\sigma \in T$ (here $*$ is list concatenation); (3) if for some positive numbers n,m we have $n < m$ and $\sigma * m \in T$, then $\sigma * n \in T$.

The elements of a tree T are called *nodes* and the node ε is called the *root* of T; given a node $\sigma \in T$, the set $T_\sigma = \{\tau \mid \sigma * \tau \in T\}$ is a tree itself, called the σ-subtree of T. A *subtree* of T is a σ-subtree of T for some $\sigma \in T$.

A Σ-*labelled tree* (or just a Σ-*tree*) is a pair (T,Λ) such that T is a tree and Λ is a map $\Lambda : T \longrightarrow \Sigma$ that associates with every node σ of T a constructor or a constant in such a way that if $\Lambda(\sigma)$ has arity n, then σ has precisely n-successors. The latter means that for every $i \geq 1$, we have $\sigma * i \in T$ iff $i \leq n$ (as a special case, $\sigma * i$ never belongs to T in case $\Lambda(\sigma)$ is a constant).

Given a node σ in a Σ-tree (T,Λ), the σ-subtree T_σ of T can be endowed with a Σ-tree structure $(T_\sigma,\Lambda_\sigma)$ by putting $\Lambda_\sigma(\tau) = \Lambda(\sigma * \tau)$ for every $\tau \in T_\sigma$. The Σ-*subtrees* of (T,Λ) are the Σ-trees of the form $(T_\sigma,\Lambda_\sigma)$, varying σ among the nodes of T. If $\Lambda(\sigma)$ is a constructor whose arity is n, we call T-*successors* of σ the Σ-trees $(T_{\sigma*1},\Lambda_{\sigma*1}),\ldots,(T_{\sigma*n},\Lambda_{\sigma*n})$; if $\Lambda(\sigma)$ is a constant, σ does not have T-successors and is said to be a *leaf* of T and of (T,Λ).

Remark 1. If (T,Λ) is a Σ-tree, we use the notation $(\sigma,f) \in (T,\Lambda)$ to mean that $\sigma \in T$ and $\Lambda(\sigma) = f$. The above notation can be used in order to *define a* Σ*-tree* (T,Λ) *by specifying by induction on the length of* σ *whether* $(\sigma,f) \in (T,\Lambda)$ holds or not, for every σ and for every (constructor or constant) f.

A Σ-tree (T,Λ) is *finite* iff the set of nodes of T is finite, it is said to be *infinite* otherwise. (T,Λ) is *rational* iff the set of Σ-subtrees of (T,Λ) is finite; notice that a rational tree can be infinite. Now comes the important definition: the sets of Σ-trees, of finite Σ-trees and of rational Σ-trees give rise to Σ-structures in the following way:

- the interpretation of a constant $a \in \Sigma$ is the singleton Σ-tree $\{\varepsilon\}$ labelled with a;
- the interpretation of the predicate At consists of the singleton Σ-trees;
- the interpretation of a constructor $h_i \in \Sigma$ of arity n maps the labelled trees $(T_1,\Lambda_1),\ldots,(T_n,\Lambda_n)$ to the labelled tree (T,Λ) where

$$T = \{\varepsilon\} \cup \bigcup_{j=1}^{n} \{j * \sigma \mid \sigma \in T_j\}$$

and Λ is so defined

$$\Lambda(\varepsilon) = h_i, \quad \Lambda(j * \sigma) = \Lambda_j(\sigma)$$

(thus (T,Λ) is the Σ-tree whose root is labelled h_i and such that the T-successors of the root are $(T_1,\Lambda_1),\ldots,(T_n,\Lambda_n)$).

It is well-known [12, 14] that the Σ-structure of all Σ-trees and of all rational Σ-trees are elementarily equivalent (i.e. the same Σ-sentences are true in them). So it makes no difference to work on rational trees or on all Σ-trees; in this paper we made the choice to work on the structure of *rational* Σ-trees. We call $\mathcal{R}$ this structure. The related complete theory $Th(\mathcal{R})$ will be also called $\mathcal{R}$ (so, for simplicity, *we use the same letter* $\mathcal{R}$ *for the set of rational* Σ*-trees, the related* Σ*-structure and the associated complete theory*).

There are some important formulae valid in $\mathcal{R}$ that we want to list below. In order to have a compact and intuitive notation, we need some abbreviations. If $\underline{x}$ is the tuple $x_1,\ldots,x_n$, let us use $\forall\underline{x}\,\phi$ for $\forall x_1 \cdots \forall x_n \phi$ - a similar convention is used for $\exists\underline{x}\phi$. If $\underline{t} = t_1,\ldots t_n$ is a tuple of terms of the same length as $\underline{x}$, then $\underline{x} = \underline{t}$ stands for $\bigwedge_{i=1}^{n} x_i = t_i$; moreover $\exists!\underline{x}\phi(\underline{x},\underline{y})$ is used for $\exists\underline{x}\,\phi(\underline{x},\underline{y}) \wedge \forall\underline{x}\forall\underline{x}'(\phi(\underline{x},\underline{y}) \wedge \phi(\underline{x}',\underline{y}) \rightarrow \underline{x} = \underline{x}')$.

It is useful to have abbreviations for formulas expressing that "x is rooted by h_i", these are

$$R_{h_i}(x) : \equiv \exists x_1 \cdots \exists x_{ar_i}(x = h_i(x_1, \ldots, x_{ar_i})) \tag{1}$$

for every $i = 1, \ldots, N$ (recall that $h_1, \ldots, h_N$ are our constructors). The claims of the following Proposition are either immediate or well-known [4]:

Proposition 1. *The following sentences are $\mathcal{R}$-valid:*

(i) $\forall \underline{x}\,(h_i(\underline{x}) = h_i(\underline{x}') \rightarrow \underline{x} = \underline{x}')$ *(for $i = 1, \ldots, N$);*
(ii) $\forall x \neg(R_{h_i}(x) \wedge R_{h_j}(x))$ *(for $i \neq j$ and $i, j = 1, \ldots, N$);*
(iii) $\forall x \neg(R_{h_i}(x) \wedge At(x))$ *(for $i = 1, \ldots, N$);*
(iv) $\forall x\,(At(x) \vee \bigvee_{i=1}^{N} R_{h_i}(x))$*;*
(v) $At(a) \wedge a \neq b$ *(for distinct constants $a, b \in \Sigma$);*
(vi) $\forall \underline{z}\,\exists! \underline{x}\,(\underline{x} = \underline{t}(\underline{x}, \underline{z}))$*, where $\underline{t}(\underline{x}, \underline{z})$ are proper flat terms.*

Above, a *flat* term is a constant, a variable, or a constructor applied to variables; a flat term is *proper* iff it is not a variable.

We make a comment on formula (vi) above. To correctly interpret it, recall that $\underline{x}, \underline{z}$ must be distinct tuples of variables including all variables occurring in $\underline{t}$ according to our conventions. Formula (vi) expresses the *existence and uniqueness of simultaneous parametric fixpoints*: the fixpoints are simultaneous because $\underline{x}$ is a tuple of variables (it is not a single variable) and the $\underline{z}$ are the parameters on which the fixpoints depend.

We underline the following important fact, which is a consequence of the above Proposition: the formula $R_{h_i}(x)$ is *existential* according to its definition (1); however, because of Proposition 1(ii)-(iv), it is also logically equivalent to a *universal* formula, namely

$$\neg At(x) \wedge \bigwedge_{j \neq i} \neg R_{h_j}(x) \tag{2}$$

(this is typical of model-complete theories, see Definition 1).

A *flat literal* is a literal of one of the following forms

$$x = t, \;\; x \neq y, \;\; At(x), \;\; \neg At(x)$$

where x, y are variables and t is a flat term; a flat literal is *proper* iff it is of the kind $At(x), x \neq y, x = t$, where x, y are variables and the term t is flat and proper. A proper flat literal of the kind $At(x)$ is said to be an *atomicity* proper flat literal, a proper flat literal of the kind $x \neq y$ is said to be a *disequality* proper flat literal, a proper flat literal of the kind $x = t$ is said to be an *equality* proper flat literal with *head variable* x.

Definition 2. *A constraint $C(\underline{x})$ is in* solved form *iff it is a conjunction of atomicity proper flat literals, disequality proper flat literals and equality proper flat literals whose head variables are* pairwise different*. We also require that if a constraint C in solved form contains an atomicity proper flat literal of the kind $At(v)$, then the variable v is not a head variable of any equality proper flat literal $v = t$ of C, unless t is a constant.*

We have an analogous notion for primitive formulae. A conjunction of variable equalities $E(\underline{x}',\underline{x})$ of the kind $x' = x$ for $x \in \underline{x}$ and $x' \in \underline{x}'$ is an *equivalence diagram* (*e-diagram* for short) iff for every $x' \in \underline{x}'$ there is exactly one equality $x' = x$ with $x \in \underline{x}$ that is a conjunct of E (that is, E is a 'logical representation' of an equivalence relation over $\underline{x} \cup \underline{x}'$, having the $\underline{x}$ as representative elements for the equivalence classes).

The role of the e-diagrams in Definition 3 and in the algorithm of Proposition 2 below is subsequent to a choice of representatives variables for equivalence classes: once a choice is done, e-diagrams neutralize unquantified non-representative variables (these variables cannot be eliminated), by moving them outside the scope of the existential quantifiers (in such position they cannot play any role in future manipulations).

Definition 3. *A primitive formula is in* solved form *iff it can be written (up to logical equivalence) in the form*

$$E(\underline{x}',\underline{x}) \wedge \exists \underline{y} C(\underline{x},\underline{y}) \tag{3}$$

where $C(\underline{x},\underline{y})$ is a constraint in solved form and $E(\underline{x},\underline{x}')$ is an e-diagram.

A primitive formula in solved form (3) has the following important property: any assignment to the free variables $\underline{x}$ satisfying C can be extended to an assignment to $\underline{x}'$ satisfying $E(\underline{x}',\underline{x}) \wedge \exists \underline{y} C(\underline{x},\underline{y})$. The following Proposition shows that the satisfiability of existential formulae can be reduced to satisfiability of primitive formulae in solved form:

Proposition 2. *Every existential formula is $\mathcal{R}$-equivalent to a disjunction of primitive formulae in solved form.*

Proof. We first flatten all terms occurring in our formula using fresh existentially quantified variables and the logical equivalence $\phi(x/t) \leftrightarrow \exists x\,(x = t \wedge \phi)$ (here x is supposed not to occur in t). Then we remove negative atom statements by the $\mathcal{R}$-equivalence $\neg A(t) \leftrightarrow \bigvee_{i=1}^{N} R_{h_i}(x)$. Applying DNF conversion and distributing the existential quantifier over disjunctions, we obtain a disjunction of primitive formulae whose matrices are conjunctions of flat literals.

Now we exhaustively apply to each disjunct of the form $\exists \underline{y}\, \phi(\underline{x},\underline{y})$ the rewrite rules below (disjuncts containing an inconsistency $v \neq v$ are in the end removed). The rules modify the whole disjunct or some conjuncts of its as indicated. When specifying the rules, we use letters $x_i \in \underline{x}$ for the free variables $\underline{x}$ of $\exists \underline{y}\, \phi(\underline{x},\underline{y})$, letter $y \in \underline{y}$ for a quantified variable of $\exists \underline{y}\, \phi(\underline{x},\underline{y})$ and letters $v, v_i, w_j \in \underline{x} \cup \underline{y}$ for any variable occurring in $\exists \underline{y}\, \phi(\underline{x},\underline{y})$; we also fix a total order $\prec$ on the free variables $\underline{x}$.

(0) $v = v \wedge \psi \Longrightarrow \psi$;
(1) $At(v) \wedge v = t \Longrightarrow v \neq v$ (if t is a non-variable, non-constant term);
(2) $v = t \wedge v = t' \Longrightarrow v \neq v$ (if t, t' are non-variable terms rooted with different constructors or constants);
(3) $v = f(v_1,\ldots,v_n) \wedge v = f(w_1,\ldots,w_n) \Longrightarrow v = f(w_1,\ldots,w_n) \wedge \bigwedge_{i=1}^{n} v_i = w_i$;
(4) $\exists \underline{y}\,(x_1 = x_2 \wedge \psi) \Longrightarrow x_1 = x_2 \wedge \exists \underline{y}\,(\psi(x_1/x_2)$ (if $x_1 \prec x_2$; in case $x_2 \prec x_1$, the rule is applied by inverting the roles of x_1 and x_2);
(5) $\exists y\,(y = v \wedge \psi) \Longrightarrow \psi(v/y)$ (where v is a variable different from y).

When applying Rule (4) and moving the equality $x_1 = x_2$ outside the existential quantifiers $\exists \bar{y}$, we simultaneously replace any free variable equality of the kind $x' = x_2$ by $x' = x_1$ (this ensures that free variables equalities outside the existential quantifiers form an e-diagram). The above rules are justified either as logical equivalences or by Proposition 1. Rules (2)-(3) adjust equality proper flat literals whose head variables are the same, whereas Rules (4)-(5) eliminate variable equalities inside the scope of the existential quantifiers. For termination, we consider pairs of natural numbers (k_1, k_2), where k_1 is the number of occurrences of constructors symbols and k_2 is the number of literals inside the existential quantifiers. All rules decrease k_2 (keeping k_1 unchanged), except Rule (3) which decreases k_1.[1] ⊣

Remark 2. From the point of view of complexity, it is clear that DNF conversion produces an exponential blow-up; however Rules (0)-(5) can be exhaustively applied in polynomial (actually quadratic) time.

4 Σ-graphs and bisimulations

In this section, we introduce a procedure to test $\mathcal{R}$-satisfiability of constraints in solved form. The procedure is based on Σ-graphs (we shall essentially use Σ-graphs as alternative representations of Σ-trees):

Definition 4. *A* Σ-graph $\mathcal{G} = (G, \{R^i\}_{i\geq 1}, \lambda)$ *is a set G endowed with an infinite family of unary partial functions R^i ($i \geq 1$) and with a labeling function $\lambda : G \longrightarrow \Sigma$ satisfying the following condition: for every $i \geq 1$, the domain of the partial function R^i is the set of $g \in G$ such that the arity of $\lambda(g)$ is larger than or equal to i.*

Notice that the domains of the R^i exclude the nodes of a Σ-graph labeled by a constant symbol. We use the notation gR^ig' to say that $g \in dom(R^i)$ and $R^i(g) = g'$. We adapt to our context the classical notion of bisimulation:

Definition 5. *A* bisimulation *between Σ-graphs $\mathcal{G} = (G, \{R^i\}_{i\geq 1}, \lambda)$ and $\mathcal{G}' = (G', \{R'_i\}_{i\geq 1}, \lambda')$ is a relation $\rho \subseteq G \times G'$ satisfying the following conditions:*

(i) *for all $g \in G, g' \in G'$, if $\rho(g, g')$ then $\lambda(g) = \lambda(g')$;*
(ii) *for all $g_1 \in G, g'_1, g'_2 \in G'$ and $i \geq 1$, if $\rho(g_1, g'_1)$ and $g'_1 R'_i g'_2$ then there exists $g_2 \in G$ such that $g_1 R^i g_2$ and $\rho(g_2, g'_2)$;*
(iii) *for all $g_1, g_2 \in G, g'_1 \in G'$ and $i \geq 1$, if $\rho(g_1, g'_1)$ and $g_1 R^i g_2$ then there exists $g'_2 \in G$ such that $g'_1 R'_i g'_2$ and $\rho(g_2, g'_2)$.*

A bisimulation relation which is a total function $\rho : G \longrightarrow G'$ is said to be a p-morphism *of $\mathcal{G}$ into $\mathcal{G}'$; if ρ is also surjective, we say that $\mathcal{G}'$ is a* quotient *of $\mathcal{G}$.*

Since our R^i are partial functions and (i) holds, conditions (ii) and (iii) in the definition of a bisimulation are trivially seen to be equivalent to each other (so one of them is redundant).

[1] It is essential that Rule (3) is applied to flat constraints, otherwise it might cause non termination if naively formulated [16].

Merge. If, for some g_1, g_2, g_1', g_2' and $i \geq 1$ we have $\rho(g_1, g_2)$, $g_1 R^i g_1'$ and $g_2 R^i g_2'$, then merge the equivalence classes of g_1', g_2'.

Fail. If, for some g_1, g_2, we have $\rho(g_1, g_2)$ and $\lambda(g_1) \neq \lambda(g_2)$, then fail.

Fig. 1. Rules of Pseudo Congruence Closure Algorithm

Bisimulations are closed under unions, so that there is the *biggest bisimulation* among two given Σ-graphs $\mathcal{G}$ and $\mathcal{G}'$. We write $g \sim_b g'$ to mean that there exists a bisimulation between the nodes g and g' of $\mathcal{G}$ and $\mathcal{G}'$ (equivalently, that we have $\rho(g, g')$ for the biggest bisimulation among $\mathcal{G}$ and $\mathcal{G}'$). Also, since the identity relation is a bisimulation, the converse of a bisimulation is a bisimulation and the composition of bisimulations is a bisimulation, the relation $\sim_b$ restricted to the nodes of the same Σ-graph $\mathcal{G}$ turns out to be an equivalence relation. Bisimulation relations between a Σ-graph and itself which are equivalence relations (called *bisimulation equivalences*) produce quotients, in the sense explained by the following Proposition:

Proposition 3. *Suppose that ρ is a bisimulation equivalence on a Σ-graph $\mathcal{G} = (G, \{R^i\}_{i\geq 1}, \lambda)$. Then there are a Σ-graph $\mathcal{G}'$ and a quotient $q : \mathcal{G} \longrightarrow \mathcal{G}'$ such that we have $\rho(g, g')$ iff $q(g) = q(g')$ for all g, g' in $\mathcal{G}$.*

Proof. We let G' be the quotient set of G under the equivalence relation ρ and q be the map associating with g its equivalence class $[g]$. We then put

$$\lambda'([g]) = \lambda(g), \qquad [g] R_i' [g'] \text{ iff } \exists g'' \text{ s.t. } \rho(g', g'') \text{ and } g R^i g'' \text{ (for all } i \geq 0) \ .$$

In this way $\mathcal{G}' = (G', \{R_i'\}_{i\geq 1}, \lambda')$ is a Σ-graph and q is the desired quotient. $\dashv$

Given a *finite* Σ-graph $\mathcal{G} = (G, \{R^i\}_{i\geq 1}, \lambda)$ and a relation $\rho_0 \subseteq G \times G$, we want to know whether there is a bisimulation equivalence ρ such that $\rho_0 \subseteq \rho$. This is a special case of a classical problem well-studied in the literature; since our relations R_i are partial functions, we can give an alternative solution in a congruence closure style: our 'Pseudo Congruence Closure' (PCC) algorithm first computes the smallest equivalence relation ρ extending ρ_0 and then exhaustively applies to it it the two rules of Figure 1. The following Proposition is clear:

Proposition 4. *There is a bisimulation equivalence extending a relation ρ_0 on a finite Σ-graph $\mathcal{G} = (G, \{R^i\}_{i\geq 1}, \lambda)$ iff PCC, applied to ρ_0, terminates without failure. In particular, for $g_1, g_2 \in G$ we have that $g_1 \sim_b g_2$ iff PCC does not fail on input $\langle \mathcal{G}, \rho_0 = \{(g_1, g_2)\}\rangle$.*

Remark 3. PCC can be simulated by an ordinary congruence closure problem (thus, PCC inherit the complexity bound $O(n \log n)$ [9, 17, 18] of congruence closure). To see this, it is sufficient to consider the signature comprising unary function symbols F_i ($i \geq 0$) and unary predicates P_f (varying f among the function and constant symbols of Σ). Notice that such a signature is infinite but only finitely many symbols are used when

handling a finite Σ-graph. Then consider the congruence closure problem given by the following literals: $g_1 = g_2$ (varying $(g_1, g_2) \in \rho_0$), $F_i(g) = g'$ (for gR^ig' in $\mathcal{G}$), $P_f(g)$ (if $\lambda(g) = f$), $\neg P_f(g)$ (if $\lambda(g) \neq f$). Now it is easy to see that, despite the fact that the R^i are partial and the F_i are total functions, PCC and standard congruence closure run in the same way.

An important (infinite) Σ-graph is the *Σ-graph of all rational trees*. This is the Σ-graph $Rat = (\mathcal{R}, \{\mathcal{R}^i\}_i, \lambda_{\mathcal{R}})$, whose underlying set is formed by the set of rational trees $\mathcal{R}$ and where we have

- $(T,\Lambda)R^i(T',\Lambda')$ iff (T',Λ') is the i-th successor of T (i.e. it is the i-th T-successor of the root subtree $T_\varepsilon = T$ of T);
- $\lambda_{\mathcal{R}}(T,\Lambda) = \Lambda(\varepsilon)$ (i.e. the label of (T,Λ) is the label of the root of T).

Inside *Rat*, bisimulation is trivial:

Proposition 5. *In the Σ-graph Rat, we have $(T,\Lambda) \sim_b (T',\Lambda')$ iff $(T,\Lambda) = (T',\Lambda')$*

Proof. Suppose that $(T,\Lambda) \sim_b (T',\Lambda')$. Recall from Remark 1 that if (T,Λ) is a Σ-tree, we write $(\sigma, f) \in (T,\Lambda)$ to mean that $\sigma \in T$ and $\Lambda(\sigma) = f$; moreover recall also that $(T,\Lambda) = (T',\Lambda')$ iff we have $(\sigma, f) \in (T,\Lambda)$ iff $(\sigma, f) \in (T',\Lambda')$ for all (σ, f). This is what we are going to prove by induction on the length of σ.

If $\sigma = \varepsilon$, then from $(T,\Lambda) \sim_b (T',\Lambda')$ it follows that the root-labeling symbols of (T,Λ) and (T',Λ') are the same.

Suppose now that $\sigma = i * \tau$ and that $(i * \tau, f) \in (T,\Lambda)$; then $\Lambda(\varepsilon)$ is a function symbol of arity bigger or equal to i, so that there is the i-th successor (T_i,Λ_i) of (T,Λ) and we have $(T,\Lambda)\mathcal{R}^i(T_i,\Lambda_i)$ and $(\tau, f) \in (T_i,\Lambda_i)$. Since (T,Λ) and (T',Λ') are bisimilar, the symbols labeling their roots are the same, so for the i-th successor (T'_i,Λ'_i) of (T',Λ') we have that $(T',\Lambda')\mathcal{R}^i(T'_i,\Lambda'_i)$ and also that $(T_i,\Lambda_i) \sim_b (T'_i,\Lambda'_i)$ (again by bisimilarity). By induction hypothesis, $(\tau, f) \in (T'_i,\Lambda'_i)$, that is $(i * \tau, f) \in (T',\Lambda')$ as required. $\dashv$

The importance of *Rat* is due to the fact that every finite Σ-graph compares with it:

Proposition 6. *For every finite Σ-graph $\mathcal{G} = (G, \{R^i\}_{i\geq 1}, \lambda)$ there is a p-morphism $p_{\mathcal{G}} : \mathcal{G} \longrightarrow Rat$.*

Proof. Given $g \in G$, let us define a rational tree $p_{\mathcal{G}}(g)$ by specifying under which conditions we have $(\sigma, f) \in p_{\mathcal{G}}(g)$ for a finite list of positive numbers σ and $f \in \Sigma$ (recall Remark 1). We stipulate that $(i_1 \cdots i_k, f) \in p_{\mathcal{G}}(g)$ iff there are $g_1, \ldots, g_k \in G$ such that $g\rho g_1 \cdots \rho g_k$ and $\lambda(g_k) = f$ (for $k = 0$, this means that $\lambda(g) = f$). From this definition, it is easy to see that $p_{\mathcal{G}}(g)$ is indeed a Σ-tree and that $p_{\mathcal{G}}$ is a p-morphism. The fact that $p_{\mathcal{G}}(g)$ is rational follows from the p-morphism condition, because the Σ-subtrees of $p_{\mathcal{G}}(g)$ turn out to be of the form $p_{\mathcal{G}}(g')$ for $g' \in G$ and G is finite. $\dashv$

We now relate Σ-graphs and constraint satisfiability in $\mathcal{R}$. We know from Proposition 2 that in order to check satisfiability of primitive formulae we can restrict to

primitive formulae in solved form. To any constraint in solved form $\phi(x_1,\ldots,x_n)$ we associate a Σ-graph

$$\mathcal{G}_\phi = (G_\phi, \{R^i_\phi\}_i, \lambda_\phi)$$

as follows. The nodes in G_ϕ are $x_1,\ldots,x_n$; we have $x_k R^i x_j$ iff ϕ contains an equality of the kind $x_k = f(\cdots,x_j,\cdots)$ (with x_j as i-th argument) and in such a case we put $\lambda_\phi(x_k) = f$. If ϕ contains an equality of the kind $x_k = a$ for a constant a, we put $\lambda_\phi(x_k) = a$. Notice that an equality like $x_k = f(\cdots,x_j,\cdots)$ or $x_k = a$ is uniquely identified for a variable x_k according to Definition 2. The variables x_k for which there are no equalities of the kind $x_k = f(\cdots)$ or $x_k = a$ in ϕ are called *external* in ϕ. For such x_k we put $\lambda_\phi(x_k) = a_k$, where the a_k are *fresh distinct constants* in Σ (recall that Σ has infinitely many constants).

Theorem 1. *A constraint $\phi(x_1,\ldots,x_n)$ in solved form is $\mathcal{R}$-satisfiable iff ϕ does not contain a disequality $x_k \neq x_j$ such that we have $x_k \sim_b x_j$ in $\mathcal{G}_\phi$.*

Proof. Consider the p-morphism $p_{\mathcal{G}_\phi} : \mathcal{G} \longrightarrow Rat$ of Proposition 6 and let us assign $p_{\mathcal{G}_\phi}(x_k)$ to every variable x_k occurring in ϕ; notice that this $\mathcal{R}$-assignment (called the *canonical $\mathcal{R}$-assignment*) satisfies all equalities and all atomicity statements in ϕ (but it might not satisfy disequalities).

Suppose that there is no disequality $x_k \neq x_j$ in ϕ such that we have $x_k \sim_b x_j$ in $\mathcal{G}_\phi$. If $x_k \neq x_j$ occurs in ϕ, we cannot have $p_{\mathcal{G}_\phi}(x_k) = p_{\mathcal{G}_\phi}(x_j)$, otherwise from $x_k \sim_b p_{\mathcal{G}_\phi}(x_k) = p_{\mathcal{G}_\phi}(x_j) \sim_b x_j$ we would get $x_k \sim_b x_j$ (this is because p-morphisms and their converses are bisimulations and bisimulations do compose). So the canonical $\mathcal{R}$-assignnment is indeed a satisfying assignment for ϕ.

Vice versa, suppose that ϕ is satisfiable. By the next lemma, the canonical $\mathcal{R}$-assignment is a satisfying assignment for ϕ. If we have $x_k \sim_b x_j$ for an inequality $x_k \neq x_j$ occuring in ϕ, then we have that $p_{\mathcal{G}_\phi}(x_k) \sim_b p_{\mathcal{G}_\phi}(x_j)$ (because p-morphisms are bisimulations and bisimulations do compose) which contradicts $p_{\mathcal{G}_\phi}(x_k) \neq p_{\mathcal{G}_\phi}(x_j)$ and Proposition 5. $\dashv$

Lemma 1. *If a constraint $\phi(x_1,\ldots,x_n)$ in solved form is $\mathcal{R}$-satisfiable, then it is satisfied by the canonical $\mathcal{R}$-assignment.*

Proof. Let α be a satisfying $\mathcal{R}$-assignment for ϕ and let α_C be the canonical $\mathcal{R}$-assignment. Since the latter can only fail to satisfy the disequalities from ϕ, it is sufficient to prove that for $x_k, x_j \in \{x_1,\ldots,x_n\}$, we have $\alpha(x_i) \neq \alpha(x_j) \Rightarrow \alpha_C(x_i) \neq \alpha_C(x_j)$. In other words, we prove that *if there is (σ, f) such that $(\sigma, f) \in \alpha(x_k)$ and $(\sigma, f) \notin \alpha(x_j)$, then $\alpha_C(x_k) \neq \alpha_C(x_j)$*. We make an induction on the length of such σ.

Independently on the inductive argument, we notice that if either x_k or x_j (or both) is external in ϕ, the claim is trivial because external variables are mapped by α_C into singleton trees labelled by fresh distinct constants by the construction of $\mathcal{G}_\phi$. If x_k, x_j are both non external, in ϕ there are equalities $x_k = l(v_1,\ldots,v_m)$ and $x_j = l(w_1,\ldots,w_m)$ with $v_1,\ldots,v_m,w_1,\ldots,v_m \in \{x_1,\ldots,x_n\}$ (in the case where in ϕ there are equalities $x_k = l(\cdots)$ and $x_j = l'(\cdots)$ with $l \neq l'$, we again trivially have $\alpha_C(x_k) \neq \alpha_C(x_j)$ because α_C satisfies the equalities in ϕ).

Let us now argue by induction on σ and restrict to the case where x_k, x_j are as above. The case $\sigma = \varepsilon$ is trivial because in that case we have $f = l$ and (ε, l) belongs

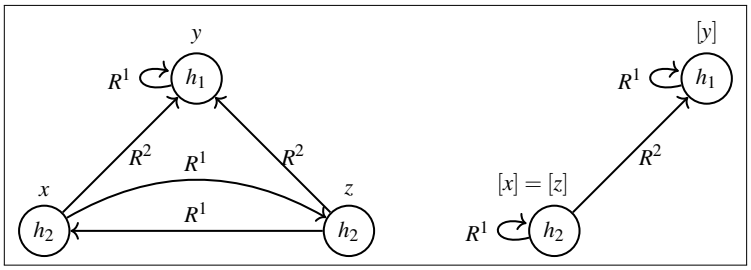

Fig. 2. Σ-graph from Example 1 (left) and its quotient under maximum bisimulation (right).

to both $\alpha(x_k)$ and $\alpha(x_j)$, contrary to the fact that we supposed $(\sigma, f) \in \alpha(x_k)$ and $(\sigma, f) \notin \alpha(x_j)$ (α is a satisfying assignment, so $(\varepsilon, l) \in \alpha(x_j)$). So assume that $\sigma = i * \tau$; then $(i * \tau, f) \in \alpha(x_k)$ implies $(\tau, f) \in \alpha(v_i)$ because α satisfies $x_k = g(v_1, \ldots, v_m)$. For the same reason, we have $(\tau, f) \notin \alpha(w_i)$, because otherwise $(i * \tau, f) \in \alpha(x_j)$ because α satisfies $x_j = g(w_1, \ldots, w_m)$. We can then apply induction to (τ, f) and conclude that $\alpha_C(v_i) \neq \alpha_C(w_i)$. Since α_C nevertheless satisfies the equalities from ϕ, from the facts that $x_k = g(v_1, \ldots, v_m)$ and $x_j = g(w_1, \ldots, w_m)$ are true under α_C and that $\alpha_C(v_i) \neq \alpha_C(w_i)$, we conclude that $\alpha_C(x_k) \neq \alpha_C(x_j)$. ⊣

Remark 4. As a consequence of Theorem 1 and Proposition 4, in order to solve a $\mathcal{R}$-constraint satisfiability problem for a constraint $\phi(\underline{x})$ in solved form, it is sufficient to run PCC on inputs $\langle \mathcal{G}_\phi, \rho_0 = \{(x_1, x_2)\} \rangle$ for every disequality $x_1 \neq x_2$ occurring in ϕ. ***The constraint is satisfiable iff PCC ends in failure for all such disequalities***. This gives a $O(n^2 \log n)$ complexity bound for constraints in solved form. As an alternative, one can directly compute the maximum bisimulation equivalence on the associated Σ-graph using some known efficient procedure [5, 7, 6].

Example 1. [*In all examples, we assume that we have only two constructors: h_1 (with arity 1) and h_2 (with arity 2)*]. Consider the constraint

$$x = h_2(z, y) \wedge y = h_1(y) \wedge z = h_2(x, y) \wedge x \neq y \wedge x \neq z$$

whose Σ-graph is depicted in Figure 2. Nodes x and y are not bisimilar in this Σ-graph, but the nodes x and z are bisimilar. Hence the constraint is unsatisfiable. ⊣

5 Definability

In this section we investigate what it means for a set of variables to be 'definable' (via selectors and simultaneous parametric fixpoints) from the free variables $\underline{x}$ of an existential formula $\exists \underline{y}\, \phi(\underline{y}, \underline{x})$. Definable and undefinable variables will be subject to different symbolic manipulations when converting an existential formula to a universal one (the possibility of such conversion is precisely the content of model completeness, see Definition 1).

Below, when we say that "ϕ is $\phi' \wedge \psi$" we mean that ϕ is a conjunction and that it can be written as specified (modulo associativity and commutativity of $\wedge$).

Definition 6. *Let $\exists \underline{y}\phi(\underline{y},\underline{x})$ be an existential formula with free variables $\underline{x}$ and bounded variables $\underline{y}$. The set of* definable variables *of $\exists \underline{y}\phi(\underline{y},\underline{x})$ is the smallest subset $D \subseteq \underline{x} \cup \underline{y}$ satisfying the following conditions:*

(o) $\underline{x} \subseteq D$;
(i) *if $u \in D$ and ϕ is $\phi' \wedge u = t(\underline{v})$ for a proper flat term t, then $\underline{v} \subseteq D$;*
(ii) *if $\underline{u} \subseteq D$ and ϕ is $\phi' \wedge \underline{v} = \underline{t}(\underline{v},\underline{u})$ for proper flat terms $\underline{t}$, then $\underline{v} \subseteq D$.*

Intuitively, the condition of Definition 6(i) says that the $\underline{v}$ are reachable from u via selectors, whereas the condition of Definition 6(ii) says that the $\underline{v}$ are reachable via some fixpoints having the $\underline{u}$ as parameters. The next couple of lemmas show how to handle definable and non definable variables: the idea is that definable variables can be converted into universally quantified variables and that non definable variables can be removed because of validity/invalidity reasons.

Lemma 2. *Suppose that the existential formula π is of the kind*

$$\exists \underline{w} \exists \underline{u}\, \phi(\underline{w},\underline{u},\underline{x})$$

and that the variables $\underline{u}$ are definable in it. The π is $\mathcal{R}$-equivalent to a formula of the kind

$$\forall \underline{u}'(\psi(\underline{u}',\underline{x}) \wedge (\theta(\underline{u}',\underline{x}) \rightarrow \exists \underline{w}\, \phi(\underline{w},\underline{u},\underline{x}))), \tag{4}$$

where $\underline{u}' \supseteq \underline{u}$ (i.e. $\underline{u}'$ possibly extends $\underline{u}$ by some fresh variables) and ψ, θ are quantifier-free. In particular, π is $\mathcal{R}$-equivalent to a universal formula *in the case where $\underline{w} = \emptyset$ (i.e. in the case where all quantified variables of π are definable).*

Proof. We view Definition 6 as a recursive definition and use the equivalences

$$\left[\exists \underline{v}'\,(u = t(\underline{v},\underline{v}') \wedge \phi')\right] \leftrightarrow \left[R_{h_i}(u) \wedge \forall \underline{w} \forall \underline{v}'(u = t(\underline{w},\underline{v}') \rightarrow \underline{v} = \underline{w} \wedge \phi')\right] \tag{5}$$

(where t is a proper flat term whose root symbol is the constructor h_i) and

$$\left[\exists \underline{v}\,(\underline{v} = t(\underline{v},\underline{u}) \wedge \phi')\right] \leftrightarrow \left[\forall \underline{v}\,(\underline{v} = t(\underline{v},\underline{u}) \rightarrow \phi')\right] \tag{6}$$

Syntactic details are straightforward (they can be found in the online available extended version of the paper). ⊣

Example 2. Consider the formula

$$\exists w_1 \exists w_2 \exists w' \exists w_2' \exists u\,(x_1 = h_2(w_1,w_2) \wedge x_2 = h_2(w_1',w_2') \wedge u = h_1(w_1) \wedge u \neq w_1) \tag{7}$$

whose existentially quantified variables are all definable. In particular, the variables w_1, w_2, w_1', w_2' falls within case (i) of Definition 6, hence their conversion into universal variables follows the schema (5). On the contrary, u is converted into a universal variable using schema (6), because u falls within the case (ii) of Definition 6: in fact, u is recursively definable from w_1 via the equality $u = h_1(w_1)$ (the latter is a fixpoint equations - indeed a trivial fixpoints equation where the variables on the left hand side of the equality symbol do not occur on the right hand side term). ⊣

Lemma 3. *Suppose that $C(\underline{x},\underline{y})$ is a constraint in solved form and that in each literal from C there is at least one occurrence of a variable from $\underline{y}$; suppose also that in the existential formula $\exists\underline{y}C(\underline{x},\underline{y})$ the variables $\underline{y}$ are not definable. Then we have*

$$\mathcal{R} \models \forall\underline{x}(\mathtt{Distinct}(\underline{x}) \rightarrow \big[\exists\underline{y}C(\underline{x},\underline{y}) \leftrightarrow \exists\underline{x}\exists\underline{y}C(\underline{x},\underline{y})\big]) \tag{8}$$

(here $\mathtt{Distinct}(\underline{x})$ *means $\bigwedge(x' \neq x'')$, varying the conjuncts on pairwise different $x',x'' \in \underline{x}$).*

Proof. First a comment: what the lemma says is that, in contexts where all parameters $\underline{x}$ are interpreted as distinct trees, $\exists\underline{y}C(\underline{x},\underline{y})$ is equivalent to the *sentence* $\exists\underline{x}\exists\underline{y}C(\underline{x},\underline{y})$, which in turn is always equivalent to either $\top$ or $\bot$ by Theorem 1.

The implication from left to right of $\leftrightarrow$ is trivial, so let us assume that in the rational tree structure $\mathcal{R}$ the sentence

$$\exists\underline{x}\exists\underline{y}C(\underline{x},\underline{y}) \tag{9}$$

is true and let us assign to $\underline{x} = x_1,\ldots,x_n$ arbitrary but distinct rational trees $\alpha(x_1),\ldots,\alpha(x_n)$. We want to show that this assignment satisfies the formula $\exists\underline{y}C(\underline{x},\underline{y})$. Since our trees are rational, the trees $\alpha(x_1),\ldots,\alpha(x_n)$ have only finitely many subtrees; let their number be $n+n'$ and let us list them as

$$\alpha(x_1),\ldots,\alpha(x_n),\alpha(x'_1),\ldots,\alpha(x'_{n'})\ . \tag{10}$$

We now write down a constraint $C'(\underline{x},\underline{x}')$ (where $\underline{x}' = x'_1,\ldots,x'_{n'}$) whose *unique* solution is precisely the $n+n'$-tuple of trees (10).[2] We shall show that the constraint

$$\exists\underline{x}\exists\underline{x}'\exists\underline{y}(C(\underline{x},\underline{y}) \wedge C'(\underline{x},\underline{x}')) \tag{11}$$

is satisfiable; this proves our claim, because the satisfiability of (11) implies in particular that $\alpha(x_1),\ldots,\alpha(x_n)$ satisfy $\exists\underline{y}C(\underline{x},\underline{y})$. We now build the Σ-graphs

$$\mathcal{G}_{(9)} = (G_{(9)},\{R^i_{(9)}\}_i,\lambda_{(9)}) \ \text{ and } \ \mathcal{G}_{(11)} = (G_{(11)},\{R^i_{(11)}\}_i,\lambda_{(11)})$$

associated with the formulae (9) and (11), respectively. In both cases, *when choosing the constants labeling external variables, we take fresh constants* not appearing in (9) and (11): we can do that because our signature Σ contains infinitely many constants and the trees $\alpha(x_1),\ldots,\alpha(x_n)$ are rational.

We make a couple of observations. Since the $\underline{y}$ occur in each literal from C and the $\underline{y}$ are not definable, the equality flat literals from C must be of the kind $v = t(\underline{x},\underline{y})$ where $v \in \underline{y}$, so *the $\underline{x}$ are not head variables in C*. Moreover, *every head variable $v \in \underline{y}$ has a path*[3] *to an external variable $w \in \underline{y}$* in $\mathcal{G}_{(9)}$ (and consequently also in the bigger Σ-graph $\mathcal{G}_{(11)}$): if it were not be so, considering the equality proper flat literals from C reachable

[2] For instance, if $\alpha(x_1)$ is rooted by h_2 and has immediate successors the pair formed by $\alpha(x'_4),\alpha(x_1)$, then C' contains as a conjunct the equality $x_1 = h_2(x'_4,x_1)$, etc. Here we are using Proposition 1(vi) (with an empty parameters tuple $\underline{z}$).

[3] By a path in a Σ-graph $\mathcal{G} = (G,\{R^i\}_{i\geq 1},\lambda)$ from a node $g \in G$ to a node $g' \in G$ we mean a chain of the kind $g = g_0 R^{i_1} g_1 \cdots g_{n-1} R^{i_k} g_k = g'$.

from v, we could obtain a set of proper flat equalities witnessing the definability of a subset of variables that includes v (see Definition (6)(ii)). By the equality proper flat literals *reachable from* v, we mean the smallest set of literals from C containing the equality proper flat literal headed by v and such that if it contains $v' = t(\underline{u},\underline{x})$, then for every $u \in \underline{u}$ it contains also the equality proper flat literal headed by u (if it exists).

Let us now consider *a bisimulation relation in* $\mathcal{G}_{(11)}$: we claim that this must be of the kind $\rho \cup id_{G_{(11)}}$, where ρ is a set of pairs (v_1, v_2) formed by nodes which are *both head nodes belonging to* $\underline{y}$. In fact, external nodes from $\underline{y}$ are not bisimilar to each other (they are labeled by different constants) nor can be bisimilar to the $\underline{x},\underline{x}'$ (the constants labeling them are disjoint from the constants labeling the $\underline{x},\underline{x}'$) nor to head nodes from $\underline{y}$ (the latter are not labeled by constants). Similarly, the nodes $\underline{x},\underline{x}'$ are not bisimilar to each other (the sub-Σ-graph of $\mathcal{G}_{(11)}$ involving the $\underline{x},\underline{x}'$ is a sub-Σ-graph of *Rat* - where the only bisimulation is identity, see Proposition 5 - recall that the $\underline{x},\underline{x}'$ represent in $\mathcal{G}_{(11)}$ the pairwise *different* trees (10)) nor to head nodes from $\underline{y}$ (because the latter have a path to some external node taken from the $\underline{y}$, as explained above, and the $\underline{x},\underline{x}'$ only have paths inside themselves).

Suppose now that (11) is not satisfiable; this means that there is a disequality $u_1 \neq u_2$ from C (recall that C' does not contain disequalities) such that PCC does not fail when initialized to $\rho = \{(u_1,u_2)\}$ in $G_{(11)}$. Since this entails that u_1 and u_2 are bisimilar in $G_{(11)}$, we have that u_1, u_2 are both head nodes belonging to $\underline{y}$. By induction, we show that PCC initialized to $\rho = \{(u_1,u_2)\}$ in $G_{(9)}$ makes precisely the same steps and halts precisely after the same number of steps as PCC initialized to $\rho = \{(u_1,u_2)\}$ in $G_{(11)}$ (this would mean that (9) is not satisfiable either, a contradiction). Suppose in fact that PCC in $G_{(11)}$ added to ρ a pair of head nodes v_1, v_2 belonging to $\underline{y}$ (only such nodes can be bisimilar and not identical to each other) and that we have $v_1 R^i_{G_{(11)}} v'_1$, $v_2 R^i_{G_{(11)}} v'_2$. Then v'_1 and v'_2 are bisimilar and so, if they are not identical to each other, they are also head nodes belonging to $\underline{y}$. In such a case, we have $v_1 R^i_{G_{(9)}} v'_1$, $v_2 R^i_{G_{(9)}} v'_2$ and so PCC in $G_{(9)}$ merges their equivalence classes precisely as PCC does in $G_{(11)}$. Moreover, when PCC halts in $G_{(11)}$, so must do PCC in $G_{(9)}$ because nodes in $\underline{y}$ can see via the R^i the same nodes in both graphs: thus if PCC halted in $G_{(11)}$ and in $G_{(9)}$ for $(v_1, v_2) \in \rho$ we have $v_1 R^i_{G_{(9)}} v'_1$, $v_2 R^i_{G_{(9)}} v'_2$, then we have also $v_1 R^i_{G_{(11)}} v'_1$, $v_2 R^i_{G_{(11)}} v'_2$ in $G_{(11)}$ and (v'_1, v'_2) has been already added by PCC to ρ (actually in both Σ-graphs) because PCC halted in $G_{(11)}$. $\dashv$

6 Model Completeness

We improve Proposition 2. Let us say that a primitive formula is in *reduced solved form* iff it can be written in the form $E(\underline{x}',\underline{x}) \wedge \exists \underline{y} C(\underline{x},\underline{y})$ (see (3)) and, in addition to the requirements of Definition 3, we also ask that the variables $\underline{y}$ are definable in $\exists \underline{y} C(\underline{x},\underline{y})$.

Theorem 2. *Every existential formula is $\mathcal{R}$-equivalent to a disjunction of primitive formulae in reduced solved form and also to a universal formula. As a consequence, $\mathcal{R}$ is model-complete.*

Proof. Consider an existential formula $\pi(\underline{x})$ of the form $\exists \underline{w}\, \phi(\underline{v},\underline{w})$. Given an equivalence relation E over $\underline{v} \cup \underline{w}$, let us call 'full display of E' the conjunction of all the

equalities $z_1 = z_2$ (for $(z_1, z_2) \in E$) and of all the disequalities $z_1 \neq z_2$ (for $(z_1, z_2) \notin E$). We conjoin the matrix $\phi(\underline{v}, \underline{w})$ of our existential formula $\exists \underline{w} \phi(\underline{v}, \underline{w})$ with the disjunction over the full displays of all the possible equivalence relations over $\underline{v} \cup \underline{w}$; then we take DNF, distribute $\exists \underline{w}$ over disjunctions and leave each disjunct be handled by the algorithm of Proposition 2. As a result, our π is equivalent to a disjunction of primitive solved forms $E(\underline{x}', \underline{x}) \wedge \exists \underline{y} C(\underline{x}, \underline{y})$, whose matrix $C(\underline{x}, \underline{y})$ is of the kind $\mathtt{Distinct}(\underline{x}, \underline{y}) \wedge \phi(\underline{x}, \underline{y})$. We shall treat these disjuncts separately.

Fix then such a disjunct. Let us split $\underline{y}$ as $\underline{y}', \underline{y}''$, where the $\underline{y}'$ are definable and the $\underline{y}''$ are not definable in $\exists \underline{y} C(\underline{x}, \underline{y})$. Thus we can rewrite $\exists \underline{y} C(\underline{x}, \underline{y})$ as

$$\exists \underline{y}' (C'(\underline{x}, \underline{y}') \wedge \exists \underline{y}'' C''(\underline{x}, \underline{y}', \underline{y}'')) \tag{12}$$

where in C'' all literals contain at least one occurrence of some of the $\underline{y}''$, the $\underline{y}'$ are definable in $\exists \underline{y}' C'(\underline{x}, \underline{y}')$, the $\underline{y}''$ are not definable in $\exists \underline{y}'' C''(\underline{x}, \underline{y}', \underline{y}'')$ and the constraint C' contains the literals $\mathtt{Distinct}(\underline{x}, \underline{y}')$ (this is because $\mathtt{Distinct}(\underline{x}, \underline{y}', \underline{y}'')$ entails $\mathtt{Distinct}(\underline{x}, \underline{y}')$). Now we can apply Lemma 3, according to which the subformula $\exists \underline{y}'' C''(\underline{x}, \underline{y}', \underline{y}'')$ can be replaced by its existential closure $\exists \underline{x} \exists \underline{y}' \exists \underline{y}'' C''(\underline{x}, \underline{y}', \underline{y}'')$ and the latter simplifies either to $\top$ or to $\bot$.

In conclusion, our initial existential formula $\pi(\underline{x})$ is equivalent to a formula of the required shape, namely to a disjunction of primitive formulae in solved form

$$\bigvee_k \left[E_k(\underline{x}'_k, \underline{x}_k) \wedge \exists \underline{y}_k C_k(\underline{y}_k, \underline{x}_k) \right], \tag{13}$$

where the $\underline{y}_k$ are definable in $\exists \underline{y}_k C_k(\underline{y}_k, \underline{x}_k)$ (these are the disjuncts surviving after the above simplifications). Such a formula is also equivalent to a universal formula by Lemma 2. ⊣

Theorem 2 (together with Theorem 1) ***gives an algorithm for deciding $\mathcal{R}$-satisfiability of any first-order sentence***, because this theorem supplies an effective procedure to rewrite an existential formula to a universal one (see the remark after Definition 1).

Example 3. Consider the formula

$$\exists w_1 \exists w_2 \forall z \forall u \, (x_1 = h_2(w_1, w_2) \wedge x_2 \neq h_1(z) \wedge \neg At(x_2) \wedge \neg (h_1(u) = u \wedge u = w_2)) \ . \tag{14}$$

To bring it to the form (13) (and then check its satisfiability), we first rewrite it as $\exists w_1 \exists w_2 \neg \exists z \exists u \neg \psi$ (here ψ is the matrix of (14)), then convert $\exists z \exists u \neg \psi$ to a universal formula θ; in this way $\exists w_1 \exists w_2 \neg \theta$ will be again existential. To the latter, we can apply the procedure of Theorem 2 and obtain a disjunction of primitive formulae in reduced solved form. After simplifications, in our case we get just one disjunct, namely

$$\exists w_1 \exists w_2 \exists w' \exists w'_2 \exists u \, (x_1 = h_2(w_1, w_2) \wedge x_2 = h_2(w'_1, w'_2) \wedge u = h_1(w_1) \wedge u \neq w_1) \tag{15}$$

(this is the same formula (7) of Example 2 which is, by the way, satisfiable according to Theorem 1). ⊣

The proof of Theorem 2 shows how to build a definitional extension of $\mathcal{R}$ enjoying quantifier elimination. To this aim, it is sufficient to add to the language new operation symbols for selectors and simultaneous parametric fixpoints.

For selectors, we need to take care of the fact that selectors are not totally defined. We adopt the classical solution of [11], saying that a badly applied selector just returns its argument. In other words, for every constructor h_i of arity ar_i, we add to the signature of $\mathcal{R}$ new unary functions symbols $\mathtt{Sel}_i^j$ (for $j = 1,\dots,ar_i$) to be interpreted in rational trees so that the following formula is true for every assignment to x, y:

$$\mathtt{Sel}_i^j(x) = y \;\leftrightarrow\; (\exists z_1 \cdots \exists z_{ar_i}\,(x = h_i(z_1,\dots,z_{ar_i}) \wedge y = z_j)) \vee (\neg R_{h_i}(x) \wedge y = x)) \quad (16)$$

For simultaneous parametric fixpoints, we need infinitely many extra functions. Consider two tuples of variables $\underline{x}^- = x_1^-,\dots,x_n^-$ and $\underline{y}^* = y_1^*,\dots,y_m^*$ and proper flat literals $\underline{x}^- = \underline{t}(\underline{x}^-,\underline{y}^*)$: below, we write $\underline{t}(-,*)$ for $\underline{t}(\underline{x}^-,\underline{y}^*)$ and $\underline{t}(\underline{x},\underline{y})$ for a substitution $\underline{t}(\underline{x}/\underline{x}^-,\underline{y}/\underline{y}^*)$. We introduce n new functions symbols $\mathtt{Fix}^i_{\underline{t}(-,*)}$ $(i = 1,\dots,n)$, all having arity m, interpreted in rational trees so that the following formula is true, for every assignment to the variable x and to the m-tuple of variables $\underline{y}$:

$$\mathtt{Fix}^i_{\underline{t}(-,*)}(\underline{y}) = x \;\leftrightarrow\; \exists \underline{x}\,(\underline{x} = \underline{t}(\underline{x},\underline{y}) \wedge x = \underline{x}_i) \quad (17)$$

where $\underline{x}_i$ denotes the i-th component of the tuple $\underline{x}$.

Let us denote with $\mathcal{R}^+$ the Σ-structure of rational trees, enriched with the interpretation of the above extra function symbols; we also denote by $\mathcal{R}^+$ the theory whose axioms are the sentences which are true in this expanded structure. From Theorem 2 and (16), (17), we have the following result (see the online available extended version of the paper for the straightforward syntactic details):

Theorem 3. *$\mathcal{R}^+$ enjoys quantifier elimination.*

Example 4. The formula $R_{h_i}(x)$ (that can be written both as a universal and as an existential formula in $\mathcal{R}$) can be rewritten in $\mathcal{R}^+$ as

$$x = h_i(\mathtt{Sel}_i^1(x),\dots,\mathtt{Sel}_i^{ar_i}(x)) \quad (18)$$

without using quantifiers. ⊣

Example 5. Eliminating quantifiers from (15), we obtain

$$x_1 = h_2(\mathtt{Sel}^1_{h_2}(x_1),\mathtt{Sel}^2_{h_2}(x_1)) \wedge x_2 = h_2(\mathtt{Sel}^1_{h_2}(x_2),\mathtt{Sel}^2_{h_2}(x_2)) \wedge \mathtt{Sel}^1_{h_2}(x_1) \neq \mathtt{Fix}^1_{h_1(-)}; \quad (19)$$

this formula says that x_1, x_2 are both rooted with h_2 and that applying the first selector to x_1 we get something different from the fixed point of h_1;[4] the latter can be represented as the infinite term

$$h_1(h_1(h_1 \cdots$$

(it is the infinite unary tree labelled by h_1). ⊣

[4] Formally, $\mathtt{Fix}^1_{h_1(-)}$ is the constant expressing the fixpoint of $z = h_1(z)$: this is the simultaneous fixpoint of a tuple of functions with empty parameters formed by the single function $h_1(z)$ (the parameters $*$ are missing and the superscript 1 means the projection to the first unique component of such tuple of functions).

7 Conclusions, Related and Further Work

In this paper we introduced the variant of the theory of finite and infinite trees, whose signature has finitely many constructors, infinitely many constants and an atomicity predicate. We proved that this theory is model complete, showed that every formula in this theory is equivalent to a disjunction of primitive formulae in reduced solved form and proved that one can achieve full quantifier elimination by adding selectors and simultaneous parametric fixpoints to the language.

One of the main insights given by the above results is the fact that every formula of our theory essentially expresses a Boolean combination of *algebraic dependency relations involving constructors, selectors and fixpoints* (see Example 5 to understand what we mean). We believe that this is the contribution lying behind statements like those of Theorems 2, 3. The 'explicit solved forms' of [4] and the 'terms with pointers' of [14] ultimately carry on this information too, but maybe in a less transparent way (a thorough comparison requires technical details, we cannot report them here for space reasons, but see the online available extended version of the paper).

One may wonder whether our analysis extends to other variants of the theory of trees. Clearly, if for instance we have infinitely many constructors in the signature, we lose the possibility of expressing formulae like $R_{h_i}(x)$ both as existential and universal formulae (see (1) and (2)), so model-completeness is likely to fail. Other partial results might probably be recovered, this has to be investigated by future work.

We wonder whether our techniques can be effective also for extensions of the theory of trees, we leave this too for future work. For instance, adding a finiteness predicate (like in [4],[22]) is worth investigating.

Concerning our algorithms, we underline that the constraint solving algorithm of Section 4 (based on bisimulations and congruence closure) is rather efficient and its complexity is comparable with analogous constraint solving algorithms from the literature [19]. The situation is different for our decision procedure once extended to all elementary formulae. We believe that the algorithm of Theorem 2, relying on model-completeness, is direct and intuitive, however important improvements still need to be designed.

A first improvement should avoid any guessing of an equivalence relation between variables in the proof of Theorem 2: this improvement however requires some care and involves a strengthening of the technical Lemma 3, so we preferred to leave it for future work. The strengthening should be able to compute a disjunction of e-diagrams over $\underline{x}$ equivalent to $\exists \underline{y} C(\underline{x}, \underline{y})$, for every existential formula $\exists \underline{y} C(\underline{x}, \underline{y})$ satisfying the peculiar hypotheses of Lemma 3.

The other source of complexity of the algorithm deciding satisfiability of all first-order formulae is the need of DNF conversions every time a quantifier alternation is removed. Although this problem is somewhat unavoidable (see the lower bound mentioned in the introduction), we believe that many redundancies arising during computations can be removed with suitable heuristics and simplification routines. Another possibility is that of exploiting the rich algebraic structure of $\mathcal{R}^+$ in order to directly design a quantifier elimination algorithm in $\mathcal{R}^+$: here the rich algebraic structure should consent the adoption of 'testing points' methods, similar to those employed in quantifier elimination for numerical domains [1].

References

1. A. Bockmayr and V. Weispfenning. *Handbook of Automated Reasoning*, volume II, chapter Solving Numerical Constraints, pages 751–844. Elsevier and MIT Press, 2001.
2. C.-C. Chang and J. H. Keisler. *Model Theory*. North-Holland Publishing Co., Amsterdam-London, third edition, 1990.
3. A. Colmerauer. Equations and inequations on finite and infinite trees. In *Fifth Generation Computer Systems*, 1984.
4. K. Djelloul, T. Dao, and T. W. Frühwirth. Theory of finite or infinite trees revisited. *Theory and Practice of Logic Programming*, 8(4):431–489, 2008.
5. A. Dovier, C. Piazza, and A. Policriti. A fast bisimulation algorithm. In *Computer Aided Verification: 13th International Conference, CAV 2001 Paris, France, July 18–22, 2001 Proceedings 13*, pages 79–90. Springer, 2001.
6. R. Gentilini, C. Piazza, and A. Policriti. Simulation as coarsest partition problem. In *Tools and Algorithms for the Construction and Analysis of Systems: 8th International Conference, TACAS 2002 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002 Grenoble, France, April 8–12, 2002 Proceedings 8*, pages 415–430. Springer, 2002.
7. R. Gentilini, C. Piazza, and A. Policriti. From bisimulation to simulation: Coarsest partition problems. *Journal of Automated Reasoning*, 31:73–103, 2003.
8. J. Jaffar. Efficient unification over infinite terms. *New Generation Computing*, 2:207–219, 1984.
9. D. Kapur. Shostak's congruence closure as completion. In H. Comon, editor, *Rewriting Techniques and Applications, 8th International Conference, RTA-97, Sitges, Spain, June 2-5, 1997, Proceedings*, volume 1232 of *Lecture Notes in Computer Science*, pages 23–37. Springer, 1997.
10. D. Kapur, R. Majumdar, and C. G. Zarba. Interpolation for data structures. In M. Young and P. T. Devanbu, editors, *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2006, Portland, Oregon, USA, November 5-11, 2006*, pages 105–116. ACM, 2006.
11. K. Kunen. Negation in logic programming. *J. of Logic Programming*, (4):289–308, 1987.
12. M. J. Maher. Complete axiomatizations of the algebras of finite, rational and infinite trees. In *Proceedings Third Annual Symposium on Logic in Computer Science*, pages 348–349. IEEE Computer Society, 1988.
13. A. Mal'cev. On the elementary theory of locally free algebras. *Soviet Math. Doklady*, pages 768–871, 1961.
14. G. Marongiu and S. Tulipani. Quantifier elimination for infinite terms. *Arch. Math. Log.*, 31(1):1–17, 1991.
15. K. L. McMillan. Lazy abstraction with interpolants. In *Proc. of CAV*, pages 123–136, 2006.
16. M. Meister and T. Frühwirth. Complexity of the CHR rational tree equation solver. In *Proc of the third Workshop on Constraint Handling Rules*, 2006.
17. G. Nelson and D. C. Oppen. Fast decision procedures based on congruence closure. *J. ACM*, 27(2):356–364, 1980.
18. R. Nieuwenhuis and A. Oliveras. Fast congruence closure and extensions. *Inf. Comput.*, 205(4):557–580, 2007.
19. A. Podelski and P. V. Roy. The beauty and the beast algorithm: Quasi-linear incremental tests of entailment and disentailment over trees. In *International Logic Programming Symposium (ILPS)*, pages 359–374. MIT Press, 1994.
20. A. Reynolds and J. C. Blanchette. A decision procedure for (co)datatypes in SMT solvers. In S. Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on*

Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016, pages 4205–4209. IJCAI/AAAI Press, 2016.

21. S. Vorobyov. An improved lower bound for the elementary theories of trees. In *Automated Deduction Cade-13: 13th International Conference on Automated Deduction New Brunswick, NJ, USA, July 30–August 3, 1996 Proceedings 13*, pages 275–287. Springer, 1996.
22. F. Zaiser and L. Ong. Abstract: The extended theory of trees and algebraic (co)datatypes. In A. Nadel and A. Niemetz, editors, *Proceedings of the 19th International Workshop on Satisfiability Modulo Theories co-located with 33rd International Conference on Computer Aided Verification (CAV 2021), Online (initially located in Los Angeles, USA), July 18-19, 2021*, volume 2908 of *CEUR Workshop Proceedings*, page 65. CEUR-WS.org, 2021.