Complexity of Deciding Syntactic Equivalence up to Renaming for Term Rewriting Systems

Michael Christian Fink Amores
LUDWIG-MAXIMILIANS-UNIVERSITÄT
Munich, Germany
Michael.Fink@campus.lmu.de

David Sabel
LUDWIG-MAXIMILIANS-UNIVERSITÄT
Munich, Germany
david.sabel@lmu.de

Inspired by questions from program transformations, eight notions of isomorphisms between term rewriting systems are defined, analysed and classified. The notions include global isomorphisms where the renaming of variables and / or function symbols is the same for all term rewriting rules of the system, and local ones where a single renaming for every rule is used. The complexity of the underlying decision problems are analysed and either shown to be efficiently solvable or proved to be complete for the graph isomorphism complexity class.

1 Introduction

Motivation and Goals. We consider programs and programming languages and their syntax and semantics. We are particularly interested in program transformation and optimization. Given a programming language (or a representation of it, like a core language) an important question is if two programs are equal. Clearly, there are several notions of equality. In this paper we are mainly interested in the question that two programs are syntactically equivalent upto a renaming of variable names and function names. However, an exact definition of such an equality depends on the concrete application.

For instance, one application is to decide whether the transformation "common subexpression elimination" is applicable, which identifies duplicated (i.e. equal) code and shares it. Another application is to identify equal programs in example data bases that are used for tests, benchmarks and competitions. Having duplicated problems in the problem set (without knowing it) can distort benchmark and competition results. For term rewriting systems (TRSs, for short), a further application is Knuth-Bendix-completion where new rewriting rules are added during the completion procedure, here one has to check whether the new rules are really new ones or whether they are already present in the system (perhaps with variables renamed, but in this application renaming of function symbols is not requested).

We consider TRSs, since they are a well-studied formalism to represent computations and programs (e.g., they are used as target language for showing termination of real programs [3, 5]). Our goal is to take into account different notions of "syntactic equivalence up to renaming" for TRSs, to compare and classify them, and to analyse their complexity as a decision problem. We want to cover such equivalences that allow to rename only the names of formal parameters (i.e. the variables of the TRSs) or also the names of functions or data (i.e. the function symbols of the TRSs). A further distinction is whether the renaming must be done globally for the whole TRS, or locally on a rule by rule basis.

Results. We introduce our notions of equality by considering isomorphisms between TRSs. The different notions stem from whether variables and function symbols are renamed globally, locally for each rule of the TRS, or not at all. We analyze the relation between the eight notions resulting in a hierarchy of equivalence notions (Proposition 2.7). We show that three (local) equivalence notions can be solved in polynomial time (Theorem 3.3) and thus their decision problems are in **P**. The other five equivalence

WPTE 2021 informal proceedings

notions are shown to be **GI**-complete (Theorem 4.1), where **GI** is the complexity class corresponding to the graph isomorphism problem.

Related work. Several isomorphism problems in this paper are shown to be graph-isomorphism-complete (**GI**-complete). For the graph-isomorphism problem as a complexity class **GI**, it is known that $\mathbf{P} \subseteq \mathbf{GI} \subseteq \mathbf{NP}$ holds, but it is unknown for both $\subseteq$-relations whether they are strict. Proving **GI**-completeness of a problem indicates hardness of the problem and that there is no known polynomial time algorithm for it. Details on the complexity class **GI** can, for instance, be found in [10, 7], a recent overview from a theoretical and a practical perspective on the graph isomorphism problem is given in [6]. Several related problems are shown to be **GI**-complete in [2, 11].

A related problem to equality of TRSs is equality of terms and expressions in languages. In [1] term equality including associative-commutative operators is shown to be **GI**-complete, but equality of TRSs is not considered. However, one may use the result to show that equivalence of TRS is in **GI** by encoding the given TRSs into single big terms with associative-commutative operators. However, for **GI**-hardness an encoding in the other direction would be required, which seems to be non-trivial. Thus, we decided to use other decision problems to prove our **GI**-completeness results.

In [9] the complexity of α-equivalence in lambda-calculi with letrec-bindings was analysed and shown to be **GI**-complete. Compared to this work, we consider and classify different notions of syntactic equivalence, we do not consider a letrec-construct (however, the sets of rules in TRSs behave similar to sets of letrec-bindings) and also no higher-order syntax (we only consider first-order terms).

Our proof method for showing that equalities based on so-called local renamings is known as the template-method which was used before in [8] where structural homomorphisms for context-free (and regular) grammars where analysed and shown to be in **P** or **GI**- or **NP**-complete.

Outline. In Section 2 we introduce various types of isomorphisms on TRSs, Section 3 shows polynomial solvability of local isomorphisms, Section 4 compares TRS isomorphism to isomorphism on respective tree encodings and proves **GI**-completeness of global isomorphisms. We conclude in Section 5. Due to space limits, several proofs are omitted. They can be found in an extended version of this paper [4].

2 Term Rewriting Systems and Structural Isomorphisms

In this section we recall the required notion on term rewriting and we define and discuss different notions of equality between term rewriting systems by defining several notions of isomorphisms between them.

We define terms as follows, where we – unlike other definitions – also allow the set of variables to be finite. This will be helpful when treating the concrete terms that (syntactically) appear as left and right hand sides in term rewriting rules. Let $\mathscr{F}$ be a finite set of *function symbols* where each $f \in \mathscr{F}$ has a fixed arity $\mathrm{ar}(f) \in \mathbb{N}_0$. Let $\mathscr{V}$ be a disjoint, finite or countably infinite set of *variables*. *Terms* $T(\mathscr{F},\mathscr{V},\mathrm{ar})$ (over function symbols $\mathscr{F}$ and variables $\mathscr{V}$) are inductively defined by: $x \in \mathscr{V}$ is a term, any constant (i.e. $f \in \mathscr{F}$, with $\mathrm{ar}(f) = 0$) is a term, and if $t_1,\ldots,t_{\mathrm{ar}(f)} \in T(\mathscr{F},\mathscr{V},\mathrm{ar})$ then $f(t_1,\ldots,t_{\mathrm{ar}(f)}) \in T(\mathscr{F},\mathscr{V},\mathrm{ar})$. For convenience, we sometimes omit ar and simply write $T(\mathscr{F},\mathscr{V})$.

We use letters $c,d,\ldots$ for constants, letters $f,g,h,\ldots$ for arbitrary functions symbols, and letters x,y,z for variables. For a term t, we use $\mathrm{Var}(t)$ to denote the variables occurring in t and $\mathrm{Func}(t)$ to denote the function symbols occurring in t. For convenience, we assume that the set of variables in a term rewriting system is finite. This will ease notation, when we define mappings between them. A *term rewriting system (TRS)* over terms $T(\mathscr{F},\mathscr{V})$ (where $\mathscr{V}$ is finite) is a finite set $\mathscr{R}$ of rules $\mathscr{R} = \{\ell_1 \to r_1,\ldots,\ell_n \to r_n\}$ where $\ell_i, r_i \in T(\mathscr{F},\mathscr{V})$ are terms, ℓ_i is not a variable, and $\mathrm{Var}(r_i) \subseteq \mathrm{Var}(\ell_i)$. We write $R = (\mathscr{F},\mathscr{V},\mathscr{R})$ for a TRS to make the function symbols and the set of variables explicit.

For functions $f : A \to B$ we also use its *extension to sets* $f(X) = \{f(x) : x \in X\}$ where $X \subseteq A$. For $X \subseteq A$, we denote the *restriction* of f to set X with $X|_f$. With $\{a_1 \mapsto b_1, \ldots, a_n \mapsto b_n\}$ we denote the map $f : \{a_1, \ldots, a_n\} \to \{b_1, \ldots, b_n\}$ mapping a_i onto b_i. Moreover, if A and B are disjoint sets, $f : A \to C$ and $g : B \to D$ maps defined on these sets, let $f \sqcup g : A \cup B \to C \cup D$ be the unique map so that $(f \sqcup g)|_A = f$ and $(f \sqcup g)|_B = g$. This can be generalised to arbitrary many, pairwise disjoint, sets and is the preferred notation to denote finite mappings.

Example 2.1. Let $\mathscr{F} = \{f, h, g, c\}$, $\mathscr{V} = \{x, y\}$ and $\mathrm{ar} = \{c \mapsto 0, g \mapsto 1, h \mapsto 1, f \mapsto 2\}$. Exemplary rewriting rules over $T(\mathscr{F}, \mathscr{V}, \mathrm{ar})$ are $h(y) \to f(c, y)$, $f(g(x), y) \to g(x)$, $h(g(x)) \to g(x)$ or $h(y) \to c$. Note that we have a total of three conditions: (1) Left and right side of a rule have to be valid terms, (2) the left side is not allowed to be a single variable, (3) the right side does not introduce new variable. Then for example, $f(x) \to c$ violates (1), $x \to h(x)$ violates (2) and $g(x) \to h(y)$ violates (3).

To model local remapping of rules, i.e. renaming symbols on a per rule basis, we introduce so-called term homomorphisms, which allow us to transform underlying term sets.

Definition 2.2 (Term Homomorphism)**.** A term homomorphism between two term sets $T(\mathscr{F}_1, \mathscr{V}_1, \mathrm{ar}_1)$ and $T(\mathscr{F}_2, \mathscr{V}_2, \mathrm{ar}_2)$ is a bijective map $\phi : (\mathscr{F}_1 \cup \mathscr{V}_1) \to (\mathscr{F}_2 \cup \mathscr{V}_2)$, such that $\phi(\mathscr{V}_1) = \mathscr{V}_2$ (and automatically $\phi(\mathscr{F}_1) = \mathscr{F}_2$) and $\mathrm{ar}_1(f) = \mathrm{ar}_2(\phi(f))$ for every $f \in \mathscr{F}_1$, canonically extended to all terms $T(\mathscr{F}_1, \mathscr{V}_1)$ via $\phi(f(t_1, \ldots, t_{\mathrm{ar}(f)})) = \phi(f)(\phi(t_1), \ldots, \phi(t_{\mathrm{ar}(f)}))$. Homomorphism ϕ is *$\mathscr{V}$-invariant* if $\mathscr{V}_1 = \mathscr{V}_2$ and $\phi|_{\mathscr{V}_1} = \mathrm{id}_{\mathscr{V}_1}$, and analogously *$\mathscr{F}$-invariant* if $\mathscr{F}_1 = \mathscr{F}_2$ and $\phi|_{\mathscr{F}_1} = \mathrm{id}_{\mathscr{F}_1}$.

Now we can specify certain normal forms of TRSs by prohibiting "equivalent" rewriting rules, based on manipulation by term homomorphisms.

Definition 2.3 (Equivalence of Rewriting Rules, Normal Forms)**.** Let $R = (\mathscr{F}, \mathscr{V}, \mathscr{R})$ be a TRS. R is in *($\mathscr{V}$-/$\mathscr{F}$-)normal form*, if we cannot find distinct rewriting rules $\ell \to r, \ell' \to r' \in \mathscr{R}$ and ($\mathscr{F}$-/$\mathscr{V}$-invariant) term homomorphism $\phi : (\mathscr{F} \cup \mathscr{V}) \to (\mathscr{F} \cup \mathscr{V})$ with $\phi(\ell) \to \phi(r) = \ell' \to r'$. Otherwise we call $\ell \to r$ and $\ell' \to r'$ *($\mathscr{V}$-/$\mathscr{F}$-)equivalent*. Clearly a TRS in normal form is already in $\mathscr{V}$-/$\mathscr{F}$-normal form.

Example 2.4 (Cont. of Example 2.1)**.** Both $\phi_1 = \{x \to y, y \mapsto x, c \mapsto c, g \mapsto g, h \mapsto h, f \mapsto f\}$ and $\phi_2 = \{x \to x, y \mapsto y, c \mapsto c, h \mapsto g, g \mapsto h, f \mapsto f\}$ are valid term homomorphisms on $T(\mathscr{F}, \mathscr{V})$ itself, with the former being $\mathscr{F}$-invariant and the latter $\mathscr{V}$-invariant. Moreover, for $\mathscr{R} = \{f(g(x), y) \to h(x), f(h(x), y) \to g(x), f(g(y), x) \to h(y)\}$, $\mathscr{R}$ is neither in $\mathscr{V}$-normal form, $\phi_1(f(g(x), y)) \to \phi_1(h(x)) = f(g(y), x) \to h(y)$ ($f(g(x), y) \to h(x)$ and $f(g(y), x) \to h(y)$ are $\mathscr{V}$-equivalent), nor in $\mathscr{F}$-normal form, $\phi_2(f(g(x), y)) \to \phi_2(h(x)) = f(h(x), y) \to g(x)$ ($f(g(x), y) \to h(x)$ and $f(h(x), y) \to g(x)$ are $\mathscr{F}$-equivalent).

We define notions of isomorphism between TRSs together with the equivalences induced by the term homomorphisms. Variables and function symbols can each be renamed locally, globally, or not at all, resulting in total of eight non-trivial, distinct isomorphism types, after combining all possibilities. Global renaming requires a common symbol-mapping applied to the whole rule set, while local renaming allows separate symbol-mappings for each rule in the set. For global isomorphisms we define variants that require the term homomorphisms to be $\mathscr{F}$- or $\mathscr{V}$-invariant, resp. for the local ones, we define variants, where all local mappings have something in common, e.g. mappings that have a common renaming on the variables, or mappings that have a common renaming on the function symbols.

Definition 2.5 (TRS Isomorphisms)**.** Let $R_i = (\mathscr{F}_i, \mathscr{V}_i, \mathscr{R}_i)$ for $i = 1, 2$ be TRSs. Then R_1 and R_2 are *globally isomorphic*, in terms $R_1 \cong_{\mathbf{GE}} R_2$, if we find a term homomorphism $\phi : (\mathscr{F}_1 \cup \mathscr{V}_1) \to (\mathscr{F}_2 \cup \mathscr{V}_2)$ with $\phi(\mathscr{R}_1) := \{\phi(\ell) \to \phi(r) : \ell \to r \in \mathscr{R}_1\} = \mathscr{R}_2$. Map ϕ is then called a *global TRS isomorphism*.

(a) If ϕ is $\mathscr{F}$-invariant, we say that R_1 and R_2 are *$\mathscr{V}$-globally isomorphic*, $R_1 \cong_{\mathbf{GVE}} R_2$, and call ϕ a *$\mathscr{V}$-global TRS isomorphism*.

Isomorphism	$\mathscr{V}$-inv.	$\mathscr{V}$-global	$\mathscr{V}$-local	$\mathscr{F}$-inv.	$\mathscr{F}$-global	$\mathscr{F}$-local	Requ. normal form
GE		✓			✓		
GVE		✓		✓			
GFE	✓				✓		
VSE		✓				✓	$\mathscr{F}$-normal form
FSE			✓		✓		$\mathscr{V}$-normal form
LE			✓			✓	normal form
LVE			✓	✓			$\mathscr{V}$-normal form
LFE	✓					✓	$\mathscr{F}$-normal form

Table 1: Comparison of TRS isomorphisms

(b) If ϕ is $\mathscr{V}$-invariant, we say that R_1 and R_2 are *$\mathscr{F}$-globally isomorphic*, $R_1 \cong_{\mathbf{GFE}} R_2$, and call ϕ a *$\mathscr{F}$-global TRS isomorphism*.

R_1 and R_2 are *locally isomorphic*, in terms $R_1 \cong_{\mathbf{LE}} R_2$, if they are in normal form and there are term homomorphisms ϕ_i, $1 \leq i \leq n = |\mathscr{R}_1|$, such that $\phi(\mathscr{R}_1) = \{\phi_1(\ell_1) \to \phi_1(r_1), \ldots, \phi_n(\ell_n) \to \phi_n(r_n)\} = \mathscr{R}_2$, where $\phi_i : (\mathscr{F}_1 \cup \mathscr{V}_1) \to (\mathscr{F}_2 \cup \mathscr{V}_2)$ and $\phi = (\phi_1, \ldots, \phi_n)$ denotes the family of such term homomorphisms. Family ϕ is then called a *local TRS isomorphism*.

(c) If R_1 and R_2 are in $\mathscr{F}$-normal form and additionally $\phi_1|_{\mathscr{V}_1} = \ldots = \phi_n|_{\mathscr{V}_1}$, we say that R_1 and R_2 are *$\mathscr{V}$-standard isomorphic*, $R_1 \cong_{\mathbf{VSE}} R_2$, and call ϕ a *$\mathscr{V}$-standard isomorphism*.

(d) If R_1 and R_2 are in $\mathscr{V}$-normal form and additionally $\phi_1|_{\mathscr{F}_1} = \ldots = \phi_n|_{\mathscr{F}_1}$, we say that R_1 and R_2 are *$\mathscr{F}$-standard isomorphic*, $R_1 \cong_{\mathbf{FSE}} R_2$, and call ϕ a *$\mathscr{F}$-standard isomorphism*.

(e) R_1 and R_2 are called *$\mathscr{V}$-locally isomorphic*, $R_1 \cong_{\mathbf{LVE}} R_2$, if $R_1 \cong_{\mathbf{FSE}} R_2$ and the ϕ_i's are $\mathscr{F}$-invariant. We then call ϕ a $\mathscr{V}$-local isomorphism.

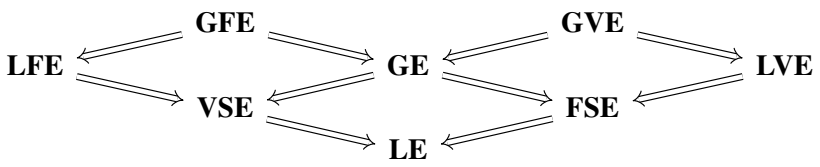
(f) R_1 and R_2 are called *$\mathscr{F}$-locally isomorphic*, $R_1 \cong_{\mathbf{LFE}} R_2$, if $R_1 \cong_{\mathbf{VSE}} R_2$ and the ϕ_i's are $\mathscr{V}$-invariant. We then call ϕ an $\mathscr{F}$-local isomorphism.

A summary concerning all types of TRS isomorphism can be found in Table 1. By extending global isomorphisms to constant families, $\cong_{\mathbf{GE}}$, $\cong_{\mathbf{GFE}}$ and $\cong_{\mathbf{GVE}}$ can be understood as special cases of $\cong_{\mathbf{LE}}$. This is in particular of good use, when composing different types of TRS isomorphisms.

Normal forms are helpful and were introduced in the first place, since they prevent "shrinking" of rewriting rule sets and ensure symmetry and transitivity of the isomorphism-relation. Consider rewriting rule sets $\mathscr{R}_1 = \{f(x) \to c, g(x) \to d, g(c) \to d\}$ and $\mathscr{R}_2 = \{f(x) \to d, g(c) \to d\}$, where the rewriting rules $f(x) \to c$, $g(x) \to d$ and $f(x) \to d$ are $\mathscr{F}$-equivalent. The induced TRSs $R_i = (\mathscr{F}_i, \mathscr{V}_i, \mathscr{R}_i)$ are $\mathscr{F}$-locally isomorphic in just one direction, i.e. one finds $\mathscr{F}$-invariant term homomorphisms $\phi = (\phi_1, \phi_2, \phi_3)$ such that $\phi(\mathscr{R}_1) = \mathscr{R}_2$, but not the other way around due to incompatible cardinalities of said rule sets.

Lemma 2.6. *Every binary relation $\sim$, defined by $R_1 \sim R_2$ iff R_1 and R_2 are ($\mathscr{V}$-$\mathscr{F}$-)locally / globally / standard isomorphic TRSs, is an equivalence relation.*

Proposition 2.7. *By definition $\mathscr{F}$-invariance implies $\mathscr{F}$-globality implies $\mathscr{F}$-locality and $\mathscr{V}$-invariance implies $\mathscr{V}$-globality implies $\mathscr{V}$-locality, and thus the following (strict) implications hold:*



Application and usefulness of the presented TRS isomorphisms may vary, e.g. let $R = (\mathscr{F}, \{x,y\}, \mathscr{R})$ where $\mathscr{F} = \{c,s,f\}$ and $R = \{f(c,x) \to x, f(s(x),y) \to s(f(x,y))\}$. This system can be interpreted as single recursive definition of addition on natural numbers by $\text{add}(0,m) = m$ and $\text{add}(\text{succ}(n),m) = \text{succ}(\text{add}(n,m))$. Introducing $\mathscr{F}' = \{0, \text{succ}, \text{add}\}$ and $\mathscr{V}' = \{n,m\}$ as new function symbol/variable sets, our initial TRS is $\mathscr{F}$-standard isomorphic to new TRS $R' = (\mathscr{F}', \mathscr{V}', \mathscr{R}')$ where $\mathscr{R}' = \{\text{add}(0,m) \to m, \text{add}(\text{succ}(n),m) \to \text{succ}(\text{add}(n,m))\}$, courtesy of $\phi_1 = \{c \mapsto 0, s \mapsto \text{succ}, f \mapsto \text{add}, x \mapsto m, y \mapsto n\}$ and $\phi_2 = \{c \mapsto 0, s \mapsto \text{succ}, f \mapsto \text{add}, x \mapsto n, y \mapsto m\}$. Important to note is that our interpretation only holds, if we (explicitly via term homomorphisms or not) rename f globally, i.e. the f in the second rule has to refer to the same function as in the first. This directly corresponds to our naive conception of terms, where functions are seen as global quantities, while variables are treated as local placeholders. In this sense, $\mathscr{F}$-standard isomorphism is rightly named as the in practice most important type of TRS isomorphism. Efficiently implemented algorithms could aid in (sophisticated) *common subexpression elimination* by checking if selected code-blocks are $\mathscr{F}$-standard isomorphic, i.e. if they offer the same functionality after suitable renaming of functions or methods. In Knuth-Bendix-completion one compares single rules without renaming of function symbols. Thus, $\cong_{\mathbf{LVE}}$ (on singleton TRSs) is the right equality notion.

3 Local TRS Isomorphisms are in P

We state two explicit polynomial time algorithms which solve **LVE**, **LFE**, **LE**, and later serve to justify polynomial reductions from **FSE** to **GFE** and **VSE** to **GVE**. This method is known as the template-method (an example concerning structural isomorphisms on context-free grammars can be found in [8]) and involves canonical renaming of variables or function symbols or both on a per rule basis, depending on the type of local isomorphism.

Lemma 3.1. *Every TRS $R = (\mathscr{F}, \mathscr{V}, \mathscr{R})$ can be brought into a so-called maximal ($\mathscr{V}$-/$\mathscr{F}$-)normal form $R' = (\mathscr{F}, \mathscr{V}, \mathscr{R}')$. I.e., $\mathscr{R}' \subseteq \mathscr{R}$ and for every $\mathscr{R}' \subsetneq \mathscr{R}'' \subseteq \mathscr{R}$, $(\mathscr{F}, \mathscr{V}, \mathscr{R}'')$ is not in ($\mathscr{V}$-/$\mathscr{F}$-)normal form. This ($\mathscr{V}$-/$\mathscr{F}$-)normal form is unique up to ($\mathscr{V}$-/$\mathscr{F}$-)equivalence of rewriting rules and can be constructed in polynomial time.*

Proof. Let $R = (\mathscr{F}, \mathscr{V}, \mathscr{R})$ be a TRS, $\mathscr{R} = \{\ell_1 \to r_1, \ldots, \ell_n \to r_n\}$, and fix $1 \leq j \leq n$. Set $\mathscr{V}_S = \{x_1, \ldots, x_K\}$, $K = |\mathscr{V}|$, and $\mathscr{F}_S = \{f_{k,l} : l \in L, 1 \leq k \leq p_l\}$, where $L = \{\text{ar}(f) : f \in \mathscr{F}\}$, $p_l = |\{f \in \mathscr{F} : \text{ar}(f) = l\}|$ and $|\mathscr{F}_S| = |\mathscr{F}|$. We call $\mathscr{V}_S$ the *standardised variable set* and $\mathscr{F}_S$ the *standardised function symbol set* of R. For $\mathscr{F}$ and $\mathscr{V}$ from Example 2.4, this results in $\mathscr{F}_S = \{f_{1,0}, f_{1,1}, f_{2,1}, f_{1,3}\}$ and $\mathscr{V}_S = \{x_1, x_2\}$. We define a local remapping $\phi_j : (\mathscr{F} \cup \mathscr{V}) \to (\mathscr{F} \cup \mathscr{V}_S)$ of variables the following way: Consider ℓ_j and replace each occurrence of a variable with x_k, where x_k refers to the kth distinct variable in ℓ_j. Since $\text{Var}(r_j) \subseteq \text{Var}(\ell_j)$ this is already sufficient to state an $\mathscr{F}$-invariant term homomorphism. Analogously, rename function symbols locally via $\phi'_j : (\mathscr{F} \cup \mathscr{V}) \to (\mathscr{F}_S \cup \mathscr{V})$ by replacing each occurrence of a function symbol with $f_{k,l}$, where $f_{k,l}$ is the kth distinct function symbol in $\ell_j r_j$ of arity l (k depends on l), resulting in a $\mathscr{V}$-invariant term homomorphism. Moreover, assign $\phi_{\mathscr{V}} = (\phi_1, \ldots, \phi_n)$ and $\phi_{\mathscr{F}} = (\phi'_1, \ldots, \phi'_n)$. Refer to Algorithms 1 and 2 for implementations. Both TRS isomorphisms $\phi_{\mathscr{V}}$ and $\phi_{\mathscr{F}}$ can be computed in at worst $\mathscr{O}(s|\mathscr{R}||\mathscr{V}||\mathscr{F}|\log(|\mathscr{V}||\mathscr{F}|))$ by linearly parsing over every rewriting rule. We call $\phi_i(\ell_i) \to \phi_i(r_i)$ the *$\mathscr{V}$-template* and $\phi'_i(\ell_i) \to \phi'_i(r_i)$ the *$\mathscr{F}$-template* of rewriting rule $\ell_i \to r_i$.

Before we proceed with the proof, a quick example: Recall Example 2.4. The $\mathscr{V}$-template of $f(g(x),y) \to h(x)$ is $f(g(x_1),x_2) \to h(x_1)$ and the $\mathscr{F}$-template is $f_{1,2}(f_{1,1}(x),y) \to f_{2,1}(x)$. Extend this to rule set $\mathscr{R} = \{f(g(x),y) \to h(x), g(f(x,y)) \to c\}$. Now we see why double indices in the functions symbol set are needed to ensure consistent arity. Renaming function symbols in the same pattern as variables would yield templates $\{f_1(f_2(x),y) \to f_3(x), f_1(f_2(x,y)) \to f_3\}$ which can not be constructed out

Algorithm 1: Computation of $\mathcal{V}$-template

input : TRS $R = (\mathcal{F}, \mathcal{V}, \mathrm{ar}, \mathcal{R})$ where $\mathcal{R} = \{\ell_1 \to r_1, \ldots, \ell_n \to r_n\}$
output : $\phi_\mathcal{V} = (\phi_1, \ldots, \phi_n)$ family of $\mathcal{F}$-invariant term homomorphisms, canonical renaming of variables on a per rule basis
runtime: $\mathcal{O}(s|\mathcal{R}||\mathcal{V}||\mathcal{F}|\log(|\mathcal{V}||\mathcal{F}|))$
Initialise new variable set $\mathcal{V}_S \leftarrow \{x_1, \ldots, x_{|\mathcal{V}|}\}$
for $j \leftarrow 1$ **to** n **do** // $\mathcal{O}(s|\mathcal{R}|\log|\mathcal{V}|)$
  Initialise partial map $\phi_j : \mathcal{F} \cup \mathcal{V} \to \mathcal{F} \cup \mathcal{V}_S$
  Let $\gamma_1 \ldots \gamma_m = \ell_j \in (\mathcal{F} \cup \mathcal{V})^*$ // store terms without symbols (,), ,
  Set $k \leftarrow 1$
  for $i \leftarrow 1$ **to** m **do** // rename variables from left to right // $\mathcal{O}(s\log|\mathcal{V}|)$
    if $\gamma_i \in \mathcal{V}$ *and* $\phi_j(\gamma_i)$ *undefined* **then** // $\mathcal{O}(\log|\mathcal{V}|)$
      Set $\phi_j(\gamma_i) \leftarrow x_k$
      $k \leftarrow k+1$
// extend ϕ_j to all of $\mathcal{F} \cup \mathcal{V}$
for $x \in \mathcal{V}$ **do** // $\mathcal{O}(|\mathcal{V}|\log|\mathcal{V}|)$
  if $\phi_j(x)$ *undefined* **then** // $\mathcal{O}(\log|\mathcal{V}|)$
    Set $\phi_j(x) \leftarrow x_k$
    $k \leftarrow k+1$
for $f \in \mathcal{F}$ **do** Set $\phi_j(f) \leftarrow f$ // $\mathcal{O}(|\mathcal{F}|\log|\mathcal{F}|)$
return $\phi_\mathcal{V} = (\phi_1, \ldots, \phi_n)$

of one singular term set due to both f_1 and f_2 appearing with arity 1 and 2. Moreover, it is not sufficient to just trivially increase the first index, i.e. remapping $f(g(x),y) \to h(x)$ to $f_{1,2}(f_{2,1}(x),y) \to f_{3,1}(x)$. The resulting template set would be $\{f_{1,2}(f_{2,1}(x),y) \to f_{3,1}(x), f_{1,1}(f_{2,2}(x,y)) \to f_{3,0}\}$, which requires an underlying function symbol set of atleast order six, namely $\{f_{1,1}, f_{1,2}, f_{2,1}, f_{2,2}, f_{3,0}, f_{3,1}\}$. This prevents bijectivity of ϕ'_j's, and thus them being well-defined term homomorphisms.

We consider $\mathcal{V}$-normal forms and $\mathcal{V}$-templates (the $\mathcal{F}$-case is analogous). $\mathcal{V}$-equivalence of distinct rewriting rules $\ell \to r, \ell' \to r'$ is equivalent to $\mathcal{V}$-local isomorphism of corresponding singleton rule sets $\{\ell \to r\}$ and $\{\ell' \to r'\}$. Corresponding equivalence classes can be determined once we have $\phi_\mathcal{V}$, since by Lemma 2.6, $[\ell_i \to r_i] = \{\ell_j \to r_j \in \mathcal{R} : \phi_i(\ell_i) \to \phi_i(r_i) = \phi_j(\ell_j) \to \phi_j(r_j)\}$. Thus, $\phi_\mathcal{V}(\mathcal{R})$ directly implies a partitioning of $\mathcal{R}$ into $1 \leq m \leq n$ sets of rewriting rules of the same $\mathcal{V}$-template. Then every representation system $\mathcal{R}' = \{\ell_{i_1} \to r_{i_1}, \ldots, \ell_{i_m} \to r_{i_m}\}$ of this equivalence relation is already in maximal $\mathcal{V}$-normal form. In the case of normal forms, consider $\psi = (\psi_1, \ldots, \psi_n)$, $\psi_i = (\phi'_i|_\mathcal{F} \sqcup \mathrm{id}_{\mathcal{V}_S}) \circ \phi_i$, well-defined due to $\mathcal{F}$-invariance of ϕ_i for each $1 \leq i \leq n$. We call $\psi_i(\ell_i) \to \psi_i(r_i)$ the *template* of rewriting rule $\ell_i \to r_i \in \mathcal{R}$. We sequentially apply Algorithms 1 and 2, i.e. we take the maximal $\mathcal{F}$-normal form of the maximal $\mathcal{V}$-normal form of R. The claim then follows by the same reasoning as above. □

The notion of ($\mathcal{V}$-/$\mathcal{F}$-)template is extended to all of TRS R, yielding new TRSs by collecting ($\mathcal{V}$-/$\mathcal{F}$-)template sets and (possibly) standardised symbol sets. We immediately conclude:

Corollary 3.2. *A TRS R in ($\mathcal{V}$-/$\mathcal{F}$-)normal form is ($\mathcal{V}$-/$\mathcal{F}$-)locally isomorphic to its ($\mathcal{V}$-/$\mathcal{F}$-)template. Moreover, as direct consequence of Lemma 2.6, two TRSs R_1 and R_2 in ($\mathcal{V}$-/$\mathcal{F}$-)normal form are ($\mathcal{V}$-/$\mathcal{F}$-)locally isomorphic iff they have the same ($\mathcal{V}$-/$\mathcal{F}$-)template, after possibly artificially extending variable/function to ensure one-to-one correspondences.*

Algorithm 2: Computation of $\mathscr{F}$-template

input : TRS $R = (\mathscr{F}, \mathscr{V}, \mathrm{ar}, \mathscr{R})$ where $\mathscr{R} = \{\ell_1 \to r_1, \ldots, \ell_n \to r_n\}$
output : $\phi_{\mathscr{F}} = (\phi'_1, \ldots, \phi'_n)$ family of $\mathscr{V}$-invariant term homomorphisms, canonical renaming of function symbols on a per rule basis
runtime: $\mathscr{O}(s|\mathscr{R}||\mathscr{V}||\mathscr{F}|\log(|\mathscr{V}||\mathscr{F}|))$

```
for l ∈ {ar(f) : f ∈ F} do let p_l ← 0                                   // O(|F| log |F|)
Initialise new function symbol set F_S ← ∅
for j ← 1 to n do                                                        // O(s|R| log |F|)
    Initialise partial map φ'_j : F ∪ V → F_S ∪ V
    Let γ_1 ... γ_m = ℓ_j r_j ∈ (F ∪ V)* // store terms without symbols (, ), ,
    for i ← 1 to m do          // rename function symbols                // O(s log |F|)
        if γ_i ∈ F and φ'_j(γ_i) undefined then
            Let l ← ar(γ_i)                                              // O(log |F|)
            Set p_l ← p_l + 1
            Set F_S ← F_S ∪ {f_{p_l,l}}
            Set φ'_j(γ_i) ← f_{p_l,l}                                    // O(log |F|)
// extend φ'_j to all of F ∪ V
for f ∈ F do                                                             // O(|F| log |F|)
    if φ'_j(f) undefined then
        Let l ← ar(f)                                                    // O(log |F|)
        Set p_l ← p_l + 1
        Set F_S ← F_S ∪ {f_{p_l,l}}
        Set φ'_j(γ_i) ← f_{p_l,l}                                        // O(log |F|)
for x ∈ V do Set φ'_j(x) ← x                                             // O(|V| log |V|)
return φ_F = (φ'_1, ..., φ'_n)
```

The last assumption ensures intended isomorphism despite ill-defined symbol sets, which are needed to ensure rigorous mathematical modelling and are often defined in retrospective after stating a consistent system of rewriting rules. Consider rewriting rules $\mathscr{R}_1 = \{f(x) \to c, f(x) \to h(x)\}$ and $\mathscr{R}_2 = \{f(x) \to c, g(x) \to h(x)\}$, implicitly extended to TRSs $R_i = (\mathscr{F}_i, \mathscr{V}_i, \mathscr{R}_i)$. Clearly, both posses the same $\mathscr{F}$-template set $\{f_{1,1}(x) \to f_{1,0}, f_{1,1}(x) \to f_{2,1}(x)\}$, but we cannot find valid term homomorphisms on underlying term sets $T(\mathscr{F}_1, \mathscr{V}_1)$ and $T(\mathscr{F}_2, \mathscr{V}_2)$, since $\mathscr{F}_1 = \{f, h\}$ and $\mathscr{F}_2 = \{f, g, h\}$ do not have the same number of function symbols. This is no problem however, we just artificially extend $\mathscr{F}_1$ by one unary "dummy" function symbol $f_{1,1}$. This procedure can be generalised to handle arbitrary TRSs as long as they posses the same $\mathscr{F}$-template sets, simply by comparing the standardised function symbol sets (which can be done in polynomial time). The same concept also holds for ($\mathscr{V}$-local) templates. Now we are equipped to prove the initial statement from the start of the section.

Theorem 3.3 (Local TRS Isomorphisms are in **P**). *Isomorphism* **LE***,* **LVE** *and* **LFE** *can be decided in polynomial time.*

Proof. We only consider the case of **LE**, the other cases are shown analogously. By Corollary 3.2, two TRSs R_1 and R_2 in normal form are locally isomorphic iff their templates R'_1 and R'_2 are the same. But this is already the case iff the sets of rewriting rules of R'_1 and R'_2 are the same. Both, construction

of templates and verification of set equality can be done in polynomial time, with latter being possible in at most most $\mathcal{O}(s_1 n_1 \log n_1 + s_2 n_2 \log n_2)$, where $n_i = |\mathscr{R}_i|$ is the number of rewriting rules and $s_i = \max\{|\ell r| : \ell \to r \in \mathscr{R}_i\}$ is an upper bound on the length of rules in R_i, $i = 1,2$. □

4 GI-Completeness of Global TRS Isomorphisms

Global, non-invariant, renaming of atleast one symbol set, immediately results in **GI**-completeness of the corresponding TRS isomorphism-decision problem:

Theorem 4.1 (**GI**-Completeness of Global TRS Isomorphisms)**.** *The following sets are polynomially equivalent:*

(*i*) $\mathbf{GI} = \{(G_1, G_2) : G_1, G_2$ *isomorphic graphs*$\}$
(*ii*) $\mathbf{GVE} = \{(R_1, R_2) : R_1, R_2$ $\mathscr{V}$*-globally isomorphic TRS*$\}$
(*iii*) $\mathbf{GFE} = \{(R_1, R_2) : R_1, R_2$ $\mathscr{F}$*-globally isomorphic TRS*$\}$
(*iv*) $\mathbf{GE} = \{(R_1, R_2) : R_1, R_2$ *globally isomorphic TRS*$\}$
(*v*) $\mathbf{FSE} = \{(R_1, R_2) : R_1, R_2$ $\mathscr{F}$*-standard isomorphic TRS*$\}$
(*vi*) $\mathbf{VSE} = \{(R_1, R_2) : R_1, R_2$ $\mathscr{V}$*-standard isomorphic TRS*$\}$

To prove Theorem 4.1, we discuss an encoding from TRSs into forests of ordered, labelled directed trees. For the rest of the paper, assume $\mathscr{F} \cup \mathscr{V}$ and $\mathbb{N}_0 \cup \{H, Z, T, F, C\}$ to be disjoint (symbol) sets.

Our graph model of choice is the following: A *labelled directed graph (LDG)* is a tuple $G = (V, E, L, \mathrm{lab})$, where V is a finite set of vertices, $E \subseteq V \times V \times L$ are directed labelled edges between vertices, L is a finite set of labels, and $\mathrm{lab} : V \to L$ is a labelling function. Vertex $v \in V$ is called a *root*, if it has no incoming edges. Graph G is called *connected*, if we can find a path between any two vertices in the undirected graph corresponding to G. Vertex $v \in V$ is called *initial*, if every other node $w \in V \setminus \{v\}$ is reachable from v. We call G *rooted*, if G is connected and has an unique, initial root. If additionally there is at most one path between any two vertices in G, then G is a *tree*. Two LDGs $G_i = (V_i, E_i, L_i, \mathrm{lab}_i)$ are *isomorphic* w.r.t. one-to-one correspondence $\psi : L_1 \to L_2$, $G_1 \cong G_2$ (ψ is suppressed), if we find a bijection $\phi : V_1 \to V_2$ such that adjacency and label-equivalence classes are preserved, i.e. $(v, w, l) \in E_1$ iff $(\phi(v), \phi(w), \psi(l)) \in E_2$, and $\psi(\mathrm{lab}_1(v)) = \mathrm{lab}_2(\phi(v))$ for $v \in V_1$. Map ϕ is called a *graph isomorphism*. If both label sets agree and ψ can be chosen as identity between them, we call G_1 and G_2 *strongly isomorphic*, $G_1 \cong_S G_2$, and ϕ a *strong graph isomorphism*. In this case, we omit ψ entirely in our notation.

We can canonically encode a term into a tree by respecting the term structure. For example, $f(x,y)$ is a tree with f-labelled root and two child-vertices, labelled with x and y respectively. However, the order of arguments is of utmost importance, since function symbols are ranked (we do not want to treat terms $f(x,y)$ and $f(y,x)$ as isomorphic). That is why we use OOLDG-trees: An *outgoing-ordered labelled directed graph (OOLDG)* [9] is a special case of an LDG, where edge-labels of outgoing edges are unique, i.e. if $(w, v, l), (w, v', l)$ are included edges, then already $v = v'$. In essence, we identify TRSs by a forest of unconnected OOLDG-trees with uniquely numbered edges, where every connected component directly corresponds to a single rewriting rule. Dependent on the type of TRS isomorphism, we modify this forest by adding additional vertices and edges or replacing labels. Of particular interest are **GFE**, **GVE** and **GE**, where, for the former, we explicitly state such a polynomial reduction into OOLDGs. The remaining isomorphisms can then be reduced to those three cases, saving unnecessary constructions.

We use the following construction: Suppose $T_1, \ldots, T_n$ are OOLDG-trees with distinct (labelled) roots $v_1, \ldots, v_n$. Graph $\mathrm{Join}(v, l, T_1, \ldots, T_n)$ is the OOLDG-tree with l-labelled root v and ordered subtrees $T_1, \ldots, T_n$, i.e. extend the union of $T_1, \ldots, T_n$ to a new OOLDG-tree by introducing new edges (v, v_i, i),

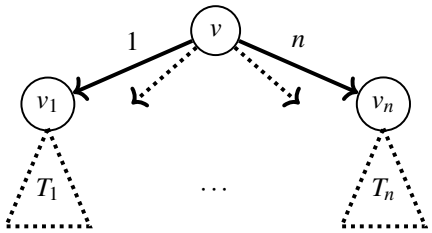


Figure 1: Visualisation of $\mathrm{Join}(v, l, T_1, \ldots, T_n)$

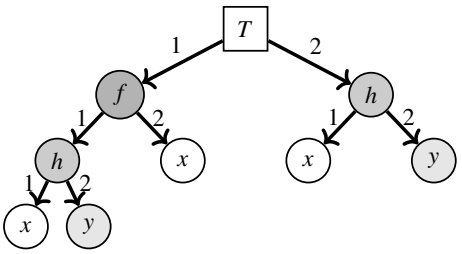


Figure 2: Term tree $\mathrm{Tree}(f(h(x,y),x) \to h(x,y))$

$1 \leq i \leq n$. An example is in Fig. 1. For a term $t \in T(\mathscr{F}, \mathscr{V})$, its *term tree* $\mathrm{Tree}(t)$ is an OOLDG-tree, inductively defined by: If $t = x$ or $t = c$ for $x \in \mathscr{V}$ or $c \in \mathscr{F}$, then $\mathrm{Tree}(t)$ is a single vertex with respective label. If $t = f(t_1, \ldots, t_{\mathrm{ar}(f)})$ for $f \in \mathscr{F}$ and terms $t_1, \ldots, t_{\mathrm{ar}(f)} \in T(\mathscr{F}, \mathscr{V})$, then $\mathrm{Tree}(t) = \mathrm{Join}(v, f, \mathrm{Tree}(t_1), \ldots, \mathrm{Tree}(t_{\mathrm{ar}(f)}))$, where v is a fresh vertex (see Fig. 2 for an example). For a rewriting rule $\ell \to r$, we generalise the definition to $\mathrm{Tree}(\ell \to r) = \mathrm{Join}([\ell \to r], T, \mathrm{Tree}(\ell), \mathrm{Tree}(r))$, where $[\ell \to r]$ is a fresh T-labelled root. For a TRS $R = (\mathscr{F}, \mathscr{V}, \mathscr{R})$, we call $G(R) = \bigcup_{\ell \to r \in \mathscr{R}} \mathrm{Tree}(\ell \to r)$, the *TRS-forest* of R. It can be constructed in time $\mathscr{O}(s|\mathscr{R}|)$, where s is an upper bound on the length of rewriting rules, if we, for example, use doubly linked lists by linearly parsing each rewriting rule $\ell \to r$.

The following lemma highlights the correlation between (global) isomorphism of TRSs and isomorphism between corresponding TRS-forests.

Lemma 4.2. *Let R_1, R_2 be two TRSs and consider the following two statements:*

(i) *R_1 and R_2 are ($\mathscr{V}$-/$\mathscr{F}$-)globally isomorphic.* (ii) *$G(R_1)$ and $G(R_2)$ are isomorphic.*

$(i) \Longrightarrow (ii)$ always holds. The reverse implication is true if edge-labels are invariant and the symbol-relationship is preserved, i.e. if $G(R_1)$ and $G(R_2)$ are isomorphic w.r.t. $\psi : L_1 \to L_2$, then $\psi|_{\mathbb{N}} = \mathrm{id}_{\mathbb{N}}$ and $\psi(\mathscr{V}_1) = \mathscr{V}_2$ (and $\psi(\mathscr{F}_1) = \mathscr{F}_2$) (for $\mathscr{V}$-/$\mathscr{F}$-global isomorphism, we demand ψ to be invariant on $\mathscr{F}$/$\mathscr{V}$).

The proof of Theorem 4.1 follows by a series of polynomial reductions, and is schematically depicted in Fig. 3 where arrows $L_1 \Rightarrow L_2$ mean that there is a polynomial reduction from L_1 to L_2, $L_1 \preceq^{\mathbf{P}} L_2$.

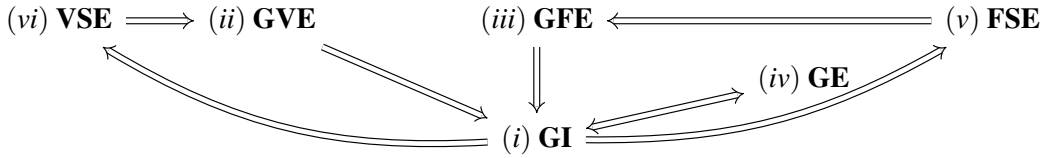


Figure 3: Proof of Theorem 4.1

Reductions **GVE** $\preceq^{\mathbf{P}}$ **GI**, **GFE** $\preceq^{\mathbf{P}}$ **GI** and **GE** $\preceq^{\mathbf{P}}$ **GI** follow the same basic pattern. The TRS-forest $G(R)$ of given TRS R, is transformed into a new OOLDG, referencing vertex-labels by introducing uniquely identifiable pointer-vertices, which represent, possibly different, classes of symbols. Strong isomorphism on these newly generated OOLDGs is then equivalent to isomorphism on initial TRS-forests, with additional constraints mirroring requirements of the reverse implication in Lemma 4.2. This in turn equals ($\mathscr{F}$-/$\mathscr{V}$-)global isomorphism on underlying TRSs, depending on the chosen encoding. Below an example encoding for the case **GFE** $\preceq^{\mathbf{P}}$ **GI**. Other cases follows analogously.

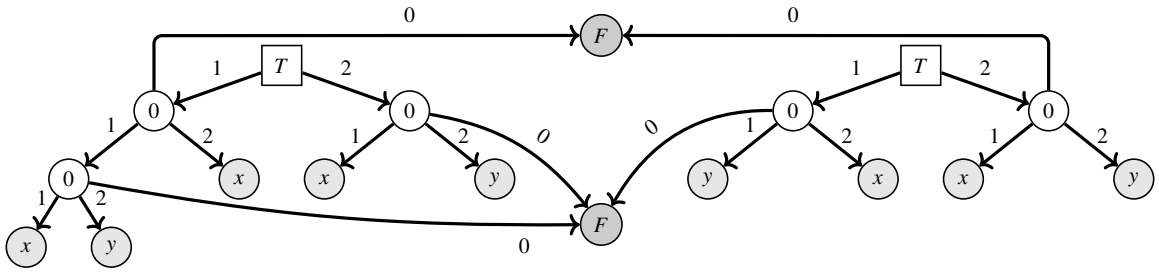


Figure 4: Example of encoding $\text{Graph}_{\mathscr{F}}(R)$ for TRS $\mathscr{R} = \{f(h(x,y),x) \to h(x,y), h(y,x) \to f(x,y)\}$

Example 4.3. Consider a TRS R based on rewriting rule set $\{f(h(x,y),x) \to h(x,y), h(y,x) \to f(x,y)\}$. For each appearing function symbol (here: f,h) introduce a fresh F-labelled vertex. Encode function symbol labels by edges, pointing to corresponding function-vertex, after replacing them with (global) uniform label 0. Refer to Fig. 4. By decoding the so generated OOLDG $\text{Graph}_{\mathscr{F}}(R)$, we extract that our TRS consists exactly of rewriting rules of the form $\square(\blacksquare(x,y),x) \to \blacksquare(x,y)$ and $\blacksquare(y,x) \to \square(x,y)$, where $\square$ and $\blacksquare$ are distinct (global) placeholders for functions, while variables x,y are explicitly used. This can be generalised, i.e. TRSs R_1 and R_2 are $\mathscr{F}$-globally isomorphic iff $\text{Graph}_{\mathscr{F}}(R_1) \cong_S \text{Graph}_{\mathscr{F}}(R_2)$.

Regarding **FSE** $\preceq^{\mathbf{P}}$ **GFE**: Two TRS $(\mathscr{F}_i, \mathscr{V}_i, \mathscr{R}_i)$, $i = 1,2$, in $\mathscr{V}$-normal form are $\mathscr{F}$-standard isomorphic iff their $\mathscr{V}$-templates $(\mathscr{F}_i, (\mathscr{V}_i)S, \phi_{\mathscr{V}_i}(\mathscr{R}_i))$, $i = 1,2$ are $\mathscr{F}$-global isomorphic. Instead of handling local variable renaming by an $\mathscr{F}$-standard isomorphism, we rename variables canonically, according to the $\mathscr{F}$-template algorithm. Existence of appropriate term homomorphisms guarantees that in both cases both $\mathscr{V}$-templates posses the same standardised variable set, i.e. the following diagram is commutative:

$$\begin{array}{ccc} (\mathscr{F}_1, \mathscr{V}_1, \mathscr{R}_1) & \xrightarrow{\phi\ \mathscr{F}\text{-standard isomorphism}} & (\mathscr{F}_2, \mathscr{V}_2, \mathscr{R}_2) \\ \phi_{\mathscr{V}_1}^{-1} \uparrow \downarrow \phi_{\mathscr{V}_1} & & \phi_{\mathscr{V}_2}^{-1} \uparrow \downarrow \phi_{\mathscr{V}_2} \\ \big(\mathscr{F}_1, (\mathscr{V}_1)_S, \phi_{\mathscr{V}_1}(\mathscr{R}_1)\big) & \xrightarrow{\phi'\ \mathscr{F}\text{-global isomorphism}} & \big(\mathscr{F}_2, (\mathscr{V}_2)_S, \phi_{\mathscr{V}_2}(\mathscr{R}_2)\big) \end{array}$$

Recall the TRS-addition example at the end of Section 2 and consider $\mathscr{V}$-template rule sets $\{\text{add}(0,x_1) \to x_1,\ \text{add}(\text{succ}(x_1),x_2) \to \text{succ}(\text{add}(x_1,x_2))\}$, $\{f(c,x_1) \to x_1, f(s(x_1),x_2) \to s(f(x_1,x_2))\}$. Now the restriction of ϕ to $\mathscr{F}$, i.e. $\phi = \{c \mapsto 0, s \mapsto \text{succ}, f \mapsto \text{add}\}$ can naturally be extended to an $\mathscr{F}$-global isomorphism between $\mathscr{V}$-templates of $\mathscr{R}$ and $\mathscr{R}'$. Reduction **VSE** $\preceq^{\mathbf{P}}$ **GVE** applies the same concept, just switch notions of $\mathscr{F}$-standard/global isomorphism and $\mathscr{V}$-standard/global isomorphism.

5 Conclusion

We introduced eight notions of syntactic equality up to renaming of function symbols and/or variables for TRSs. We have shown for each of them that they are either efficiently decidable or they are **GI**-complete. This clarifies the complexity of the equalities. Instead of explicitly stating graph encodings for every global isomorphism case, we took advantage of the general structure of TRSs in form of templates and, in particular the relationship between variables and function symbols. Presented template-generating-algorithms can easily be generalised to handle more than two disjoint symbol sets and can then be applied for reductions or polynomial solvability proofs in the context of of new isomorphisms if they demand symbol-relationship to be preserved. In fact, we saw that under canonical remapping of symbols, proof of local isomorphisms reduces to simple set comparison. Runtime of those algorithms can be greatly improved by choice of of more appropriate data structures.

References

[1] David A. Basin (1994): *A term equality problem equivalent to graph isomorphism*. Information Processing Letters 51(2), pp. 61–66, doi:https://doi.org/10.1016/0020-0190(94)00084-0.

[2] Kellogg S. Booth & Charles J. Colbourn (1979): *Problems polynomially equivalent to graph isomorphism*. Technical Report CS-77-04, University of Waterloo. Available at `https://cs.uwaterloo.ca/research/tr/1977/CS-77-04.pdf`.

[3] Marc Brockschmidt, Richard Musiol, Carsten Otto & Jürgen Giesl (2012): *Automated Termination Proofs for Java Programs with Cyclic Data*. In P. Madhusudan & Sanjit A. Seshia, editors: Computer Aided Verification - 24th International Conference, CAV 2012, Berkeley, CA, USA, July 7-13, 2012 Proceedings, Lecture Notes in Computer Science 7358, Springer, pp. 105–122, doi:10.1007/978-3-642-31424-7_13.

[4] Michael Christian Fink Amores & David Sabel (2021): *Complexity of Deciding Syntactic Equivalence up to Renaming for Term Rewriting Systems (Extended Version)*. CoRR abs/2106.13520. Available at `https://arxiv.org/abs/2106.13520`.

[5] Jürgen Giesl, Matthias Raffelsieper, Peter Schneider-Kamp, Stephan Swiderski & René Thiemann (2011): *Automated termination proofs for haskell by term rewriting*. ACM Trans. Program. Lang. Syst. 33(2), pp. 7:1–7:39, doi:10.1145/1890028.1890030.

[6] Martin Grohe & Pascal Schweitzer (2020): *The Graph Isomorphism Problem*. Commun. ACM 63(11), p. 128–134, doi:10.1145/3372123.

[7] Johannes Köbler, Uwe Schöning & Jacobo Torán (1992): *Graph Isomorphism is Low for PP*. Comput. Complex. 2, pp. 301–330, doi:10.1007/BF01200427.

[8] Daniel J. Rosenkrantz & Harry B. Hunt III (1985): *Testing for Grammatical Coverings*. Theor. Comput. Sci. 38, pp. 323–341, doi:10.1016/0304-3975(85)90226-9.

[9] Manfred Schmidt-Schauß, Conrad Rau & David Sabel (2013): *Algorithms for Extended Alpha-Equivalence and Complexity*. In Femke van Raamsdonk, editor: 24th International Conference on Rewriting Techniques and Applications, RTA 2013, June 24-26, 2013, Eindhoven, The Netherlands, LIPIcs 21, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 255–270, doi:10.4230/LIPIcs.RTA.2013.255.

[10] Uwe Schöning (1988): *Graph Isomorphism is in the Low Hierarchy*. J. Comput. Syst. Sci. 37(3), pp. 312–323, doi:10.1016/0022-0000(88)90010-4.

[11] Viktor N. Zemlyachenko, Nickolai M. Korneenko & Regina I. Tyshkevich (1985): *Graph isomorphism problem*. J. Math. Sci. (N. Y.) 29, pp. 1426–1481, doi:10.1007/BF02104746.