A Verified Decision Procedure for the First-Order Theory of Rewriting for Linear Variable-Separated Rewrite Systems

Alexander Lochmann
Department of Computer Science
University of Innsbruck
Innsbruck, Austria
alexander.lochmann@uibk.ac.at

Aart Middeldorp
Department of Computer Science
University of Innsbruck
Innsbruck, Austria
aart.middeldorp@uibk.ac.at

Fabian Mitterwallner
Department of Computer Science
University of Innsbruck
Innsbruck, Austria
fabian.mitterwallner@uibk.ac.at

Bertram Felgenhauer
Austria
int-e@gmx.de

Abstract

The first-order theory of rewriting is a decidable theory for finite left-linear right-ground rewrite systems, implemented in FORT. We present a formally verified variant of the decision procedure for the class of linear variable-separated rewrite systems. This variant supports a more expressive theory and is based on the concept of anchored ground tree transducers. The correctness of the decision procedure is verified by a formalization in Isabelle/HOL on top of the Isabelle Formalization of Rewriting (IsaFoR).

CCS Concepts: • **Theory of computation → Equational logic and rewriting**; **Logic and verification**; **Tree languages**.

Keywords: first-order theory of rewriting, tree automata, formalization

ACM Reference Format:
Alexander Lochmann, Aart Middeldorp, Fabian Mitterwallner, and Bertram Felgenhauer. 2021. A Verified Decision Procedure for the First-Order Theory of Rewriting for Linear Variable-Separated Rewrite Systems. In *Proceedings of the 10th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP '21), January 18–19, 2021, Virtual, Denmark.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3437992.3439918

We thank Franziska Rapp and T. V. H. Prathamash for contributions in the early stages of this work.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

CPP '21, January 18–19, 2021, Virtual, Denmark
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8299-1/21/01.
https://doi.org/10.1145/3437992.3439918

1 Introduction

Dauchet and Tison [6] proved that the first-order theory of rewriting is decidable for finite ground rewrite systems. In this theory well-known properties like ground-confluence, (weak) normalization and termination are expressible. The decision procedure is based on tree automata techniques [3] and implemented in FORT [12, 13]. To ensure the correctness of FORT, a natural idea is to formalize the underlying decision procedure for the first-order theory and certify the yes/no answers produced by FORT. The contribution of this paper lies in the former; we present a complete formalization in Isabelle/HOL of a variant of the decision procedure for the larger class of linear *variable-separated* rewrite systems.

Felgenhauer et al. [8] formalized operations on ground tree transducers and RR_n automata in order to obtain a verified ground-confluence prover for linear variable-separated systems. More recently, Lochmann and Middeldorp [11] presented formalized proofs of the infinity and normal form predicates. The former is crucial for expressing the termination property. The latter is based on a direct automaton construction due to Comon [2] and gives rise to a more efficient procedure than the one based on the formula $\neg\,\exists\, u\,(t \to u)$.

Formalization efforts are critical for ensuring correctness. An additional outcome is that they may give rise to simpler and more efficient constructions and algorithms. For instance, the formalized proof in [8] that relations accepted by ground tree transducers are effectively closed under transitive closure is considerably simpler than the textbook proof in [3, Theorem 3.2.14]. Also in this paper we revisit the underlying automata theory. We introduce the class of *anchored* ground tree transducers. These are similar to ground tree transducers but have better closure properties, which reduces the number of constructions needed to represent the first-order theory of rewriting. Some of these closure properties are proved (and formalized) using the simple but equivalent

class of *pair automata*. Due to the contributions in this paper, the formalized theory is actually more expressive than the one supported by FORT.

Our formalizations are based on IsaFoR [14],[1] an Isabelle/HOL library containing numerous abstract results and concrete techniques from the rewriting literature. Our own development can be found at

http://cl-informatik.uibk.ac.at/software/fortissimo/cpp2021

Most definitions, theorems, and lemmata in this paper directly correspond to the formalization. These are indicated by the ☑ symbol, which links to an HTML rendering of our formalization in the accompanying supplementary material, for those who like to dive right into the actual Isabelle code. In the running text (traditional) proof details are given.

The remainder of the paper is organized as follows. In the next section we recall basic notions and results on term rewriting, tree automata and ground tree transducers, and the first-order theory of rewriting. The new contributions start in Section 3, where we introduce, in a systematic way, several context closure operations on binary relations that are used to represent the binary predicates in the first-order theory of rewriting. Anchored ground tree transducers and pair automata are introduced in Section 4, and in Section 5 we present closure properties of relations accepted by anchored ground tree transducers. Section 6 relates anchored ground tree transducers to RR_2 automata and presents the (formalized) results on RR_n automata. In Section 7 we show, on a concrete example formula, how the results of the preceding sections are used to verify whether the formula holds for a particular rewrite system. Details of the Isabelle formalization are given in Section 8. We conclude in Section 9.

2 Preliminaries

Familiarity with first-order term rewriting [1] and tree automata [3] will be helpful. Below we recall some definitions and notations.

2.1 Term Rewriting

We assume a finite signature $\mathcal{F}$ containing at least one constant symbol and a disjoint set of variables $\mathcal{V}$. The set of terms built up from $\mathcal{F}$ and $\mathcal{V}$ is denoted by $\mathcal{T}(\mathcal{F},\mathcal{V})$, while $\mathcal{T}(\mathcal{F})$ denotes the (non-empty) set of ground terms. The set of variables occurring in a term t is denoted by $\mathcal{V}\mathrm{ar}(t)$. A term is linear if it does not contain multiple occurrences of the same variable. Positions are strings of positive integers which are used to address subterms. The set of positions in a term t is denoted by $\mathcal{P}\mathrm{os}(t)$ and the root position by ϵ. A substitution is a mapping σ from variables to terms and $t\sigma$ denotes the result of applying σ to a term t. A context C is a term that contains exactly one hole, denoted by the special constant $\Box \notin \mathcal{F}$. We write $C[t]$ for the result of replacing the hole in C by the term t. A term rewrite system (TRS) $\mathcal{R}$ is a set of rules $\ell \to r$ between terms $\ell, r \in \mathcal{T}(\mathcal{F},\mathcal{V})$. A TRS $\mathcal{R}$ is linear if its rewrite rules consist of linear terms. We call $\mathcal{R}$ *variable-separated* if $\mathcal{V}\mathrm{ar}(\ell) \cap \mathcal{V}\mathrm{ar}(r) = \emptyset$ for every $\ell \to r \in \mathcal{R}$. In this paper we are concerned with finite, linear, variable-separated TRSs $\mathcal{R}$ and we consider rewriting on ground terms: $t \to_{\mathcal{R}} u$ for ground terms t, u if there exist a context C, a rewrite rule $\ell \to r \in \mathcal{R}$, and a substitution σ such that $t = C[\ell\sigma]$ and $u = C[r\sigma]$. We write $\to^*_{\mathcal{R}}$ for the reflexive and transitive closure of $\to_{\mathcal{R}}$. Further relations on terms will be introduced in the next section. We drop the subscript $\mathcal{R}$ when it can be inferred from the context. A ground normal form is a ground term t such that $t \to_{\mathcal{R}} u$ for no term u. We write $\mathsf{NF}(\mathcal{R})$ for the set of ground normal forms of $\mathcal{R}$.

Example 2.1. We use the TRS $\mathcal{R}$ consisting of the rewrite rules

$$\mathsf{a} \to \mathsf{b} \qquad \mathsf{f}(\mathsf{a}) \to \mathsf{b} \qquad \mathsf{g}(\mathsf{a}, x) \to \mathsf{f}(\mathsf{a})$$

as leading example in this paper. We have $\mathsf{f}(\mathsf{g}(\mathsf{a},\mathsf{b})) \to^*_{\mathcal{R}} \mathsf{f}(\mathsf{b})$ with ground normal form $\mathsf{f}(\mathsf{b})$.

2.2 Tree Automata

A (finite bottom-up) tree automaton $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ consists of a finite signature $\mathcal{F}$, a finite set Q of states, disjoint from $\mathcal{F}$, a subset $Q_f \subseteq Q$ of final states, and a set of transition rules Δ. Every transition rule has one of the following two shapes: $f(p_1,\dots,p_n) \to q$ with $f \in \mathcal{F}$ and $p_1,\dots,p_n,q \in Q$, or $p \to q$ with $p, q \in Q$. Transition rules of the second shape are called ϵ-transitions. Transition rules can be viewed as rewrite rules between ground terms in $\mathcal{T}(\mathcal{F} \cup Q)$. The induced rewrite relation is denoted by $\to_\Delta$ or $\to_{\mathcal{A}}$. A ground term $t \in \mathcal{T}(\mathcal{F})$ is accepted by $\mathcal{A}$ if $t \to^*_\Delta q$ for some $q \in Q_f$. The set of all accepted terms is denoted by $L(\mathcal{A})$ and a set L of ground terms is regular if $L = L(\mathcal{A})$ for some tree automaton $\mathcal{A}$. A tree automaton $\mathcal{A}$ is deterministic if there are no ϵ-transitions and no two transition rules with the same left-hand side. We say that $\mathcal{A}$ is completely defined if it contains a transition rule with left-hand side $f(p_1,\dots,p_n)$ for every n-ary function symbol f and every combination $p_1,\dots,p_n$ of states. All regular sets are accepted by a completely defined, deterministic tree automaton. The class of regular sets is effectively closed under boolean operations. Moreover, membership and emptiness are decidable.

2.3 Regular Relations

For relations on ground terms two different types of automata are used. The first one is restricted to binary relations. A ground tree transducer (GTT for short) is a pair $\mathcal{G} = (\mathcal{A}, \mathcal{B})$ of tree automata over the same signature $\mathcal{F}$. Let s and t be ground terms in $\mathcal{T}(\mathcal{F})$. We say that the pair (s,t) is accepted by $\mathcal{G}$ if $s \to^*_{\mathcal{A}} u \;{}^*_{\mathcal{B}}\!\leftarrow t$ for some term $u \in \mathcal{T}(\mathcal{F} \cup Q)$. Here Q is the combined set of states of $\mathcal{A}$ and $\mathcal{B}$. The set of all such pairs is denoted by $L(\mathcal{G})$. Observe

[1] http://cl-informatik.uibk.ac.at/isafor/

that $L(\mathcal{G})$ is a binary relation on $\mathcal{T}(\mathcal{F})$. A binary relation $\bowtie$ on ground terms is a GTT relation if there exists a GTT $\mathcal{G}$ such that $\bowtie = L(\mathcal{G})$. The class of GTT relations is effectively closed under composition, inverse, and transitive closure [4]. GTT relations are not closed under the boolean operations complement, intersection, and union.

The second method uses standard tree automata operating on an encoding of the relation as a set of ground terms over a special signature. For a signature $\mathcal{F}$ and $n > 0$ we let $\mathcal{F}^{(n)} = (\mathcal{F} \cup \{\bot\})^n$. Here, $\bot \notin \mathcal{F}$ is a fresh constant. The arity of a symbol $f_1 \cdots f_n \in \mathcal{F}^{(n)}$ is the maximum of the arities of $f_1, \ldots, f_n$. Given n terms $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F})$, the term $\langle t_1, \ldots, t_n \rangle$ is the unique term $u \in \mathcal{T}(\mathcal{F}^{(n)})$ such that $\mathcal{P}\mathrm{os}(u) = \mathcal{P}\mathrm{os}(t_1) \cup \cdots \cup \mathcal{P}\mathrm{os}(t_n)$ and $u(p) = f_1 \cdots f_n$ where $f_i = t_i(p)$ if $p \in \mathcal{P}\mathrm{os}(t_i)$ and $\bot$ otherwise, for all positions $p \in \mathcal{P}\mathrm{os}(u)$.

Example 2.2. For $\mathcal{F} = \{\mathsf{a}, \mathsf{b}, \mathsf{f}, \mathsf{g}\}$ in Example 2.1 we have

$$\langle \mathsf{g}(\mathsf{a}, \mathsf{f}(\mathsf{b})), \mathsf{f}(\mathsf{a}) \rangle = \mathsf{gf}(\mathsf{aa}, \mathsf{f}\bot(\mathsf{b}\bot)) \in \mathcal{T}(\mathcal{F}^{(2)})$$

$$\langle \mathsf{a}, \mathsf{f}(\mathsf{f}(\mathsf{b})), \mathsf{g}(\mathsf{b}, \mathsf{a}) \rangle = \mathsf{afg}(\bot\mathsf{fb}(\bot\mathsf{b}\bot), \bot\bot\mathsf{a}) \in \mathcal{T}(\mathcal{F}^{(3)})$$

An n-ary relation R on $\mathcal{T}(\mathcal{F})$ is regular if its encoding $\{\langle t_1, \ldots, t_n \rangle \mid (t_1, \ldots, t_n) \in R\}$ is regular. The class of all n-ary regular relations is denoted by RR_n. It is effectively closed under boolean operations. We present three more closure operations. Let R be an n-ary relation over $\mathcal{T}(\mathcal{F})$. If $n \geqslant 2$ and $1 \leqslant i \leqslant n$ then the i-th projection of R is the relation $\Pi_i(R) = \{(t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n) \mid (t_1, \ldots, t_n) \in R\}$. If $1 \leqslant i \leqslant n+1$ then the i-th cylindrification of R is the relation $C_i(R) = \{(t_1, \ldots, t_{i-1}, u, t_i, \ldots, t_n) \mid (t_1, \ldots, t_n) \in R \text{ and } u \in \mathcal{T}(\mathcal{F})\}$. Moreover, if σ is a permutation on $\{1, \ldots, n\}$ then $\sigma(R) = \{(t_{\sigma(1)}, \ldots, t_{\sigma(n)}) \mid (t_1, \ldots, t_n) \in R\}$. The class of regular relations is effectively closed under projection, cylindrification, and permutation. The class of binary regular relations includes all GTT relations, but unlike the latter, it is not closed under composition and transitive closure.

2.4 First-Order Theory of Rewriting

Besides $\to$ and $\to^*$, many other relations on ground terms are of interest: $\mathrel{+\!\!\!\to}$ (parallel rewriting), $\to_\epsilon$ (rewrite step at the root position), and $\to_{>\epsilon}$ (rewrite step below the root), to name a few. These relations constitute predicates in the first-order theory of rewriting. The theory that we consider in this paper is a first-order logic over a language without function symbols that contains (among others) the following predicate symbols:

$$\to \quad \mathrel{+\!\!\!\to} \quad \to_\epsilon \quad \to_{>\epsilon} \quad \to^+ \quad \to^* \quad \leftrightarrow^* \quad \mathsf{INF}_\bowtie$$

As models we consider finite, linear, variable separated TRSs $\mathcal{R}$ whose signature $\mathcal{F}$ contains at least one constant symbol. The non-empty set of ground terms $\mathcal{T}(\mathcal{F})$ serves as domain for the variables in formulas. The binary predicate symbols have their expected meaning. The unary predicate $\mathsf{INF}_\bowtie$ is parameterized by an arbitrary yet regular binary relation $\bowtie$ and holds for a ground term t if the set $\{u \mid t \bowtie u\}$ is infinite. When $\bowtie$ is instantiated to $\to^+$, $\mathsf{INF}_\bowtie$ holds for ground terms with infinitely many different reducts. It is used to characterize the termination property:

$$\neg\, \exists\, t\ (\mathsf{INF}_{\to^+}(t) \ \vee\ t \to^+ t)$$

This formula correctly models termination for every finite variable-separated TRS $\mathcal{R}$, even if $\to_\mathcal{R}$ is not finitely branching. (More precisely, if $\mathcal{R}$ contains a rewrite rule $\ell \to r$ with a variable $x \in \mathcal{V}\mathrm{ar}(r) \setminus \mathcal{V}\mathrm{ar}(\ell)$ then $\mathcal{R}$ is not terminating. By taking any ground instance of ℓ as t, $t \to^+ t$ holds when $r = x$ and $\mathsf{INF}_{\to^+}(t)$ holds when $r \neq x$ because then there must be infinitely many ground terms u such that $t \to^+ r\{x \mapsto u\}$.) We briefly describe the decision procedure for this theory. A more detailed explanation can be found in [12]. Given a formula φ, RR_2 automata are constructed for the atomic subformulas with a binary predicate symbol. Depending on the predicate, intermediate GTTs may need to be constructed before an RR_2 automaton is obtained. Atomic subformulas with the unary $\mathsf{INF}_\bowtie$ predicate are translated into RR_1 automata (which are just standard tree automata). The logical operators of the formula are then mapped to closure operations on RR_n automata. A non-emptiness (emptiness) check of the final automaton reveals whether φ holds (or not).

The formalized variant of the decision procedure that we present in this paper differs in the construction of RR_2 automata for the atomic subformulas with a binary predicate symbol. In FORT, $\to$ is directly represented as an RR_2 automaton whereas for $\to^*$ first a GTT is constructed for $\mathrel{+\!\!\!\to}$, which is subsequently subjected to a transitive closure operation to obtain a representation of $\mathrel{+\!\!\!\to}^+ = \to^*$, and then finally transformed into an RR_2 automaton. For the other binary relations (like $\to^+$, $\to_{>\epsilon}$, and $\leftrightarrow^*$) ad-hoc constructions on GTTs and RR_2 automata are used. The formalized procedure described below generates *all* binary relations from the root step relation $\to_\epsilon$, which is represented by an *anchored* GTT, using a few primitive closure operators. This *uniform* presentation benefits the formalization effort by reducing duplication. As a byproduct, a richer set of relations is supported.

3 Context Operations

In the next few sections we describe formalized automata constructions to decide the first-order theory of rewriting. To save considerable formalization efforts, we introduce a few primitives that operate on binary relations that are accepted by various kinds of tree automata. These primitives are sufficient to generate all binary rewrite relations supported by FORT. For defining the semantics of the primitives, we introduce some context operations on binary relations in this section.

Definition 3.1. Let $\mathcal{F}$ be a signature. A *multi-hole context* is an element of $\mathcal{T}(\mathcal{F} \uplus \{\Box\})$ where $\Box$ is a fresh constant

symbol, called *hole*. If C is a multi-hole context with n holes and $t_1, \ldots, t_n$ are terms in $\mathcal{T}(\mathcal{F})$ then $C[t_1, \ldots, t_n]$ denotes the term in $\mathcal{T}(\mathcal{F})$ obtained from C by replacing the holes from left to right with $t_1, \ldots, t_n$. We write $\mathcal{C}$ for the set of all multi-hole contexts. Given a binary relation $\bowtie$ on ground terms in $\mathcal{T}(\mathcal{F})$ and a set of multi-hole contexts $\mathcal{D} \subseteq \mathcal{C}$, we write $\mathcal{D}(\bowtie)$ for the relation $\{(C[t_1, \ldots, t_n], C[u_1, \ldots, u_n]) \mid C \in \mathcal{D}$ has n holes and $t_i \bowtie u_i$ for all $1 \leqslant i \leqslant n\}$.

We consider two ways to restrict multi-hole contexts: restricting the number of holes and restricting the position of the holes.

- We denote the set of multi-hole contexts with exactly one hole by $\mathcal{C}^1$. The set of multi-hole contexts with at least one hole is denoted by $\mathcal{C}^>$. Moreover $\mathcal{C}^\geqslant$ simply denotes $\mathcal{C}$.
- We denote the set of multi-hole contexts with the property that every hole occurs below the root position by $\mathcal{C}_>$. This includes the set $\mathcal{T}(\mathcal{F})$ of ground terms (which are multi-hole contexts without holes). Similarly, $\mathcal{C}_\epsilon$ denotes the set of multi-hole contexts with the property that every hole occurs at the root position. So $\mathcal{C}_\epsilon = \{\Box\} \cup \mathcal{T}(\mathcal{F})$. Moreover, $\mathcal{C}_\geqslant$ simply denotes $\mathcal{C}$.

By combining both types of restrictions, we obtain nine ways for defining new binary relations.

Definition 3.2. Let $\bowtie$ be a binary relation on $\mathcal{T}(\mathcal{F})$. Given a number constraint $n \in \{\geqslant, 1, >\}$ and a position constraint $p \in \{\geqslant, \epsilon, >\}$, we define the binary relation $\bowtie_p^n$ on $\mathcal{T}(\mathcal{F})$ as $(\mathcal{C}^n \cap \mathcal{C}_p)(\bowtie)$. ☑

Note that $\bowtie_\epsilon^\geqslant = \bowtie^=$ and $\bowtie_\epsilon^1 = \bowtie_\epsilon^> = \bowtie$, for any $\bowtie$.

Example 3.3. Recall the TRS $\mathcal{R}$ from our leading example and consider the multi-hole contexts

$$C_1 = \Box \quad C_2 = \mathsf{f}(\Box) \quad C_3 = \mathsf{g}(\Box, \mathsf{a}) \quad C_4 = \mathsf{g}(\Box, \Box) \quad C_5 = \mathsf{f}(\mathsf{a})$$

We have $C_1, C_2, C_3 \in \mathcal{C}^1$, $C_1, C_2, C_3, C_4 \in \mathcal{C}^>$, $C_1, C_5 \in \mathcal{C}_\epsilon$, and $C_2, C_3, C_4, C_5 \in \mathcal{C}_>$. Moreover, $(C_2[\mathsf{a}], C_2[\mathsf{b}]) \in (\to_\mathcal{R})_>^1$ and $(C_4[\mathsf{a}, \mathsf{a}], C_4[\mathsf{b}, \mathsf{b}]) \notin (\to_\mathcal{R})_>^1$.

Because $\mathcal{C}_\geqslant = \mathcal{C}^\geqslant = \mathcal{C}$, the relation $\bowtie_\geqslant^\geqslant$ is the multi-hole context closure of $\bowtie$. Using the root step relation $\to_\epsilon$ induced by a linear, variable-separated TRS $\mathcal{R}$ as $\bowtie$, we obtain eight different relations for $(\to_\epsilon)_p^n$:

$$(\to_\epsilon)_\geqslant^\geqslant = \twoheadrightarrow \qquad (\to_\epsilon)_\geqslant^1 = \to \qquad (\to_\epsilon)_\geqslant^> = \dot{+}\!\!\to$$
$$(\to_\epsilon)_\epsilon^\geqslant = \to_\epsilon^= \qquad (\to_\epsilon)_\epsilon^1 = \to_\epsilon \qquad (\to_\epsilon)_\epsilon^> = \to_\epsilon$$
$$(\to_\epsilon)_>^\geqslant = \twoheadrightarrow_{>\epsilon} \qquad (\to_\epsilon)_>^1 = \to_{>\epsilon} \qquad (\to_\epsilon)_>^> = \dot{+}\!\!\to_{>\epsilon}$$

Here $\dot{+}\!\!\to$ denotes a non-empty parallel step, $\twoheadrightarrow_{>\epsilon}$ a parallel step where only redexes below the root are contracted, and $\dot{+}\!\!\to_{>\epsilon}$ a non-empty parallel step where only redexes below the root are contracted.

Example 3.4. Consider the term pairs $\pi_1 = (\mathsf{g}(\mathsf{a}, \mathsf{a}), \mathsf{g}(\mathsf{b}, \mathsf{b}))$, $\pi_2 = (\mathsf{g}(\mathsf{a}, \mathsf{a}), \mathsf{f}(\mathsf{a}))$, and $\pi_3 = (\mathsf{g}(\mathsf{a}, \mathsf{a}), \mathsf{g}(\mathsf{a}, \mathsf{a}))$. We have $\pi_1, \pi_2, \pi_3 \in \twoheadrightarrow$, $\pi_1, \pi_2 \in \dot{+}\!\!\to$, $\pi_1 \in \twoheadrightarrow_{>\epsilon} \cap \dot{+}\!\!\to_{>\epsilon}$, and $\pi_3 \notin \dot{+}\!\!\to_{>\epsilon}$.

4 Anchored Ground Tree Transducers

For representing the root step relation $\to_\epsilon$ of a TRS we use the following modification of GTTs.

Definition 4.1. For a GTT $\mathcal{G} = (\mathcal{A}, \mathcal{B})$, the relation $\{(s, t) \mid s \to_\mathcal{A}^* q \,{}_\mathcal{B}^*\!\leftarrow t$ for some $q \in Q\}$ is the *anchored* GTT relation associated with $\mathcal{G}$ and is denoted by $L_a(\mathcal{G})$. ☑

The resulting language class coincides with binary $Rec_\times$ as defined in [3, Section 3.2.1]. The more operational view in Definition 4.1 benefits the developments below.

We obviously have $L_a(\mathcal{G}) \subseteq L(\mathcal{G})$. Anchored GTT relations have the advantage that they can represent the root step relation $\to_\epsilon$. Moreover, they have better closure properties than GTT relations. When we speak of "anchored GTTs", we always have $L_a(\mathcal{G})$ in mind.

According to the following lemma, the multi-hole context closure of an anchored GTT relation is a GTT relation using the same GTT.

Lemma 4.2. *For every GTT $\mathcal{G}$, $L(\mathcal{G}) = L_a(\mathcal{G})_\geqslant^\geqslant$.* ☑

Proof. Let $\mathcal{G} = (\mathcal{A}, \mathcal{B})$. If $(s, t) \in L(\mathcal{G})$ then there exist a context C with $n \geqslant 0$ holes, terms $s_1, \ldots, s_n, t_1, \ldots, t_n$, and states $q_1, \ldots, q_n$ such that $s = C[s_1, \ldots, s_n]$, $t = C[t_1, \ldots, t_n]$, and $s_i \to_\mathcal{A}^* q_i \,{}_\mathcal{B}^*\!\leftarrow t_i$ for all $1 \leqslant i \leqslant n$. We have $(s_i, t_i) \in L_a(\mathcal{G})$ for all $1 \leqslant i \leqslant n$ by definition of anchored GTTs. Moreover, $C \in \mathcal{C}_\geqslant \cap \mathcal{C}^\geqslant$. Hence $(s, t) \in L_a(\mathcal{G})_\geqslant^\geqslant$. The converse is equally easy. □

In the formalization we also employ an equivalent but more flexible definition.

Definition 4.3. A *pair automaton* is a triple $\mathcal{P} = (Q_2, \mathcal{A}, \mathcal{B})$ where $\mathcal{A}, \mathcal{B}$ are tree automata and $Q_2 \subseteq Q_\mathcal{A} \times Q_\mathcal{B}$. We define $L(\mathcal{P}) = \{(s, t) \mid s \to_\mathcal{A} p$ and $t \to_\mathcal{B} q$ with $(p, q) \in Q_2\}$. ☑

Lemma 4.4. *Anchored GTTs and pair automata are equivalent.* ☑☑

Proof. If $\mathcal{G} = (\mathcal{A}, \mathcal{B})$ is a GTT then $L_a(\mathcal{G}) = L(\mathcal{P})$ for the pair automaton $\mathcal{P} = (Q_2, \mathcal{A}, \mathcal{B})$ with $Q_2 = \{(p, p) \mid p \in Q_\mathcal{A} \cap Q_\mathcal{B}\}$. Conversely, given a pair automaton $\mathcal{P} = (Q_2, \mathcal{A}, \mathcal{B})$, we first rename the states of $\mathcal{B}$ to obtain an equivalent tree automaton $\mathcal{B}'$ such that $\mathcal{A}$ and $\mathcal{B}'$ do not share states. We add an ϵ-transition $p \to q'$ to $\mathcal{A}$ for every $(p, q) \in Q_2$, resulting in the tree automaton $\mathcal{A}'$. Here q' is the (renamed) state in $\mathcal{B}'$ that corresponds to state q in $\mathcal{B}$. The GTT $\mathcal{G} = (\mathcal{A}', \mathcal{B}')$ satisfies $L_a(\mathcal{G}) = L(\mathcal{P})$. □

The above lemma will be used in the sequel without mention. Different constructions (e.g. [4, 5, 8]) exist to prove the following basic result. The other binary relations associated with a TRS $\mathcal{R}$ (like $\twoheadrightarrow_\mathcal{R}$ and $\leftrightarrow_\mathcal{R}^*$) will be obtained from the root step relation $\to_\epsilon$ by automata constructions that operate on anchored GTT relations and RR_2 relations.

Lemma 4.5. *The relation $\to_\epsilon$ is an anchored GTT relation for every linear variable-separated TRS $\mathcal{R}$.* ☑

Example 4.6. The anchored GTT $\mathcal{G} = (\mathcal{A}, \mathcal{B})$ with $\mathcal{A}$ consisting of the transition rules

$$\mathsf{a} \to 0 \qquad \mathsf{b} \to 0 \qquad \mathsf{f}(0) \to 0 \qquad \mathsf{g}(0,0) \to 0$$
$$\mathsf{a} \to 1 \qquad \mathsf{f}(1) \to 2 \qquad \mathsf{g}(1,0) \to 3$$

and $\mathcal{B}$ consisting of the transition rules

$$\mathsf{a} \to 4 \qquad \mathsf{b} \to 1 \qquad \mathsf{b} \to 2 \qquad \mathsf{f}(4) \to 3$$

accepts the root step relation $\to_\epsilon$ of our leading TRS $\mathcal{R}$. For instance, the ground instance $\mathsf{g}(\mathsf{a}, \mathsf{f}(\mathsf{b})) \to \mathsf{f}(\mathsf{a})$ of the third rewrite rule $\mathsf{g}(\mathsf{a}, x) \to \mathsf{f}(\mathsf{a})$ of $\mathcal{R}$ is accepted in state 3:

$$\mathsf{g}(\mathsf{a}, \mathsf{f}(\mathsf{b})) \to^*_{\mathcal{A}} \mathsf{g}(1, \mathsf{f}(0)) \to_{\mathcal{A}} \mathsf{g}(1,0) \to_{\mathcal{A}} 3$$
$$\mathsf{f}(\mathsf{a}) \to_{\mathcal{B}} \mathsf{f}(4) \to_{\mathcal{B}} 3$$

The pair automaton $\mathcal{P} = (\{(1,1),(2,2),(3,3)\}, \mathcal{A}, \mathcal{B})$ accepts the same relation.

5 Closure Properties

After having introduced the basic primitives, we turn to composition and transitive closure.

Definition 5.1. Given tree automata $\mathcal{A}$ and $\mathcal{B}$, $\Delta_\epsilon(\mathcal{A}, \mathcal{B})$ is the set of ϵ-transitions $\rightsquigarrow$ defined by the inference rules in Figure 1.

First we recall the result of [8, Lemma 4.2]. Lemma 5.2 below is presented as a definition in [3, 4].

Lemma 5.2. $\Delta_\epsilon(\mathcal{A}, \mathcal{B}) = \{(p,q) \mid p \;{}^*_{\mathcal{A}}\!\leftarrow t \to^*_{\mathcal{B}} q$ *for some ground term* $t\}$. □

Example 5.3. For the (anchored) GTT $\mathcal{G} = (\mathcal{A}, \mathcal{B})$ of Example 4.6 the set $\Delta_\epsilon(\mathcal{A}, \mathcal{B})$ consists of the following six ϵ-transitions:

$$0 \rightsquigarrow 4 \qquad (0 \;{}_{\mathcal{A}}\!\leftarrow \mathsf{a} \to_{\mathcal{B}} 4)$$
$$1 \rightsquigarrow 4 \qquad (1 \;{}_{\mathcal{A}}\!\leftarrow \mathsf{a} \to_{\mathcal{B}} 4)$$
$$0 \rightsquigarrow 1 \qquad (0 \;{}_{\mathcal{A}}\!\leftarrow \mathsf{b} \to_{\mathcal{B}} 1)$$
$$0 \rightsquigarrow 2 \qquad (0 \;{}_{\mathcal{A}}\!\leftarrow \mathsf{b} \to_{\mathcal{B}} 2)$$
$$0 \rightsquigarrow 3 \qquad (0 \;{}_{\mathcal{A}}\!\leftarrow \mathsf{f}(0) \rightsquigarrow \mathsf{f}(4) \to_{\mathcal{B}} 3)$$
$$2 \rightsquigarrow 3 \qquad (2 \;{}_{\mathcal{A}}\!\leftarrow \mathsf{f}(1) \rightsquigarrow \mathsf{f}(4) \to_{\mathcal{B}} 3)$$

Since $\mathcal{G}$ does not contain ϵ-transitions, only the congruence rule [c] is used here.

Lemma 5.4. *Anchored GTT relations are effectively closed under composition.* ☑

Proof. Let $\mathcal{P}_1 = (Q_2^1, \mathcal{A}_1, \mathcal{B}_1)$ and $\mathcal{P}_2 = (Q_2^2, \mathcal{A}_2, \mathcal{B}_2)$ be pair automata (operating on terms over the same signature). We construct the pair automaton $\mathcal{P} = (Q_2, \mathcal{A}_1, \mathcal{B}_2)$ with

$$Q_2 = Q_2^1 \circ \Delta_\epsilon(\mathcal{B}_1, \mathcal{A}_2) \circ Q_2^2$$

We claim that $L(\mathcal{P}) = L(\mathcal{P}_1) \circ L(\mathcal{P}_2)$. First let $(s,t) \in L(\mathcal{P})$. We have $s \to^*_{\mathcal{A}_1} p$ and $t \to^*_{\mathcal{B}_2} q$ for some $(p,q) \in Q_2$. The definition of Q_2 yields states p' and q' such that $(p,p') \in Q_2^1$, $(p',q') \in \Delta_\epsilon(\mathcal{B}_1, \mathcal{A}_2)$, and $(q',q) \in Q_2^2$. According to Lemma 5.2 there exists a ground term u such that $u \to^*_{\mathcal{B}_1} p'$ and $u \to^*_{\mathcal{A}_2} q'$. Hence $(s,u) \in L(\mathcal{P}_1)$ and $(u,t) \in L(\mathcal{P}_2)$ and thus $(s,t) \in L(\mathcal{P}_1) \circ L(\mathcal{P}_2)$.

For the converse, let $(s,t) \in L(\mathcal{P}_1) \circ L(\mathcal{P}_2)$. So there exists a ground term u such that $(s,u) \in L(\mathcal{P}_1)$ and $(u,t) \in L(\mathcal{P}_2)$. Hence there are pairs $(p_1,q_1) \in Q_2^1$ and $(p_2,q_2) \in Q_2^2$ such that $s \to^*_{\mathcal{A}_1} p_1$, $u \to^*_{\mathcal{B}_1} q_1$, $u \to^*_{\mathcal{A}_2} p_2$, and $t \to^*_{\mathcal{B}_2} q_2$. Lemma 5.2 yields $(q_1,p_2) \in \Delta_\epsilon(\mathcal{B}_1, \mathcal{A}_2)$. Hence $(p_1,q_2) \in Q_2$ and therefore $(s,t) \in L(\mathcal{P})$. □

Lemma 5.5. *Anchored GTT relations are effectively closed under transitive closure.* ☑

Proof. Let $\mathcal{P} = (Q_2, \mathcal{A}, \mathcal{B})$ be a pair automaton. We construct the pair automaton $\mathcal{P}_+ = (\Delta_+(\mathcal{P}), \mathcal{A}, \mathcal{B})$ where $\Delta_+(\mathcal{P})$ is the binary relation on states defined by the inference rules in Figure 2. We claim that $L(\mathcal{P}_+) = L(\mathcal{P})^+$. From the first inference rule we immediately obtain $L(\mathcal{P}) \subseteq L(\mathcal{P}_+)$. The second inference rule, together with the definition of Q_2 in the proof of Lemma 5.4, yields $L(\mathcal{P}_+) \circ L(\mathcal{P}_+) \subseteq L(\mathcal{P}_+)$. Hence $L(\mathcal{P})^+ \subseteq L(\mathcal{P}_+)$.

For the converse, let $(s,t) \in L(\mathcal{P}_+)$. So there exists a pair $p \rightsquigarrow q$ such that $s \to^*_{\mathcal{A}} p$ and $t \to^*_{\mathcal{B}} q$. We prove $(s,t) \in L(\mathcal{P})^+$ by induction on the derivation of $p \rightsquigarrow q$. If $(p,q) \in Q_2$ then $(s,t) \in L(\mathcal{P})$. Suppose $p \rightsquigarrow p'$, $(p',q') \in \Delta_\epsilon(\mathcal{B}, \mathcal{A})$, and $q' \rightsquigarrow q$. According to Lemma 5.2 there exists a ground term u such that $u \to^*_{\mathcal{B}} p'$ and $u \to^*_{\mathcal{A}} q'$. The induction hypothesis yields $(s,u) \in L(\mathcal{P})^+$ and $(u,t) \in L(\mathcal{P})^+$. Hence also $(s,t) \in L(\mathcal{P})^+$. □

Example 5.6. For the pair automaton $\mathcal{P} = (Q_2, \mathcal{A}, \mathcal{B})$ associated with the anchored GTT $\mathcal{G} = (\mathcal{A}, \mathcal{B})$ of Example 4.6 we have $Q_2 = \{(1,1),(2,2),(3,3)\}$. Moreover $\Delta_\epsilon(\mathcal{B}, \mathcal{A}) = \Delta_\epsilon(\mathcal{A}, \mathcal{B})^- = \{(4,0),(4,1),(1,0),(2,0),(3,0),(3,2)\}$ according to the computation in Example 5.3. Hence we obtain the pair automaton $\mathcal{P}_+ = (\Delta_+(\mathcal{P}), \mathcal{A}, \mathcal{B})$ with $\Delta_+(\mathcal{P}) = \{(1,1),(2,2),(3,3),(3,2)\}$. We have $\mathsf{g}(\mathsf{a},\mathsf{b}) \to_\epsilon \mathsf{f}(\mathsf{a}) \to_\epsilon \mathsf{b}$ and the pair $(\mathsf{g}(\mathsf{a},\mathsf{b}), \mathsf{b})$ is accepted by $\mathcal{P}_+$: $\mathsf{g}(\mathsf{a},\mathsf{b}) \to^*_{\mathcal{A}} 3$ and $\mathsf{b} \to_{\mathcal{B}} 2$ with $(3,2) \in \Delta_+(\mathcal{P})$. Furthermore, $\mathsf{g}(\mathsf{a},\mathsf{b}) \to_\epsilon \mathsf{f}(\mathsf{a}) \to \mathsf{f}(\mathsf{b})$ but $\mathsf{g}(\mathsf{a},\mathsf{b}) \to^+_\epsilon \mathsf{f}(\mathsf{b})$ does not hold, and one readily checks that the pair $(\mathsf{g}(\mathsf{a},\mathsf{b}), \mathsf{f}(\mathsf{b}))$ is not accepted by $\mathcal{P}_+$.

Two further closure operations on anchored GTT relations are inverse and union. Recall that GTT relations are not closed under union.

Lemma 5.7. *Anchored GTT relations are effectively closed under inverse and union.* ☑☑

Proof. Given a pair automaton $\mathcal{P} = (Q_2, \mathcal{A}, \mathcal{B})$, we have $L(\mathcal{P})^- = L(\mathcal{P}^-)$ for the pair automaton $\mathcal{P}^- = (Q_2^-, \mathcal{B}, \mathcal{A})$. Here $Q_2^- = \{(q,p) \mid (p,q) \in Q_2\}$. Given pair automata $\mathcal{P}_1 = (Q_2^1, \mathcal{A}_1, \mathcal{B}_1)$ and $\mathcal{P}_2 = (Q_2^2, \mathcal{A}_2, \mathcal{B}_2)$ without common

$$\frac{q \;\mathcal{A}\!\leftarrow p \quad p \rightsquigarrow r}{q \rightsquigarrow r}\ [\text{a}] \qquad \frac{p \rightsquigarrow q \quad q \rightarrow_{\mathcal{B}} r}{p \rightsquigarrow r}\ [\text{b}] \qquad \frac{p \;\mathcal{A}\!\leftarrow f(p_1,\ldots,p_n) \quad p_1 \rightsquigarrow q_1 \;\cdots\; p_n \rightsquigarrow q_n \quad f(q_1,\ldots,q_n) \rightarrow_{\mathcal{B}} q}{p \rightsquigarrow q}\ [\text{c}]$$

Figure 1. $\Delta_\epsilon(\mathcal{A},\mathcal{B})$: ϵ-transitions for (anchored) GTT composition.

$$\frac{(p,q) \in Q_2}{p \rightsquigarrow q} \qquad \frac{p \rightsquigarrow q \quad (q,q') \in \Delta_\epsilon(\mathcal{B},\mathcal{A}) \quad q' \rightsquigarrow r}{p \rightsquigarrow r}$$

Figure 2. $\Delta_+(\mathcal{P})$: ϵ-transitions for transitive closure of pair automata.

states, $L(\mathcal{P}_1) \cup L(\mathcal{P}_2) = L(\mathcal{P})$ for the pair automaton $\mathcal{P} = (Q_1^2 \cup Q_2^2, \mathcal{A}_1 \cup \mathcal{A}_2, \mathcal{B}_1 \cup \mathcal{B}_2)$. □

Lemma 5.8. *The composition of an anchored GTT relation and a GTT relation is an anchored GTT relation.*

Proof. Let $\mathcal{P} = (Q_2, \mathcal{A}_1, \mathcal{B}_1)$ be a pair automaton and $\mathcal{G} = (\mathcal{A}_2, \mathcal{B}_2)$ a GTT. Without loss of generality we assume that $\mathcal{P}$ and $\mathcal{G}$ do not share states. Define the pair automaton

$$\mathcal{P}' = (Q_2, \mathcal{A}_1, \mathcal{B}_1 \cup \Delta_\epsilon(\mathcal{A}_2,\mathcal{B}_1) \cup \mathcal{B}_2)$$

We claim that $L(\mathcal{P}') = L(\mathcal{P}) \circ L(\mathcal{G})$. First let $(s,t) \in L(\mathcal{P}')$. So $s \rightarrow^*_{\mathcal{A}_1} p$ and $t \rightarrow^*_{\mathcal{B}'} q$ with $(p,q) \in Q_2$ and $\mathcal{B}'$ abbreviating $\mathcal{B}_1 \cup \Delta_\epsilon(\mathcal{A}_2,\mathcal{B}_1) \cup \mathcal{B}_2$. Because $\mathcal{P}$ and $\mathcal{G}$ do not share states, the sequence $t \rightarrow^*_{\mathcal{B}'} q$ can be rearranged as follows:

$$\begin{aligned} t = C[t_1,\ldots,t_n] &\rightarrow^*_{\mathcal{B}_2} C[q_1,\ldots,q_n] \\ &\rightarrow^*_{\Delta_\epsilon(\mathcal{A}_2,\mathcal{B}_1)} C[r_1,\ldots,r_n] \rightarrow^*_{\mathcal{B}_1} q \end{aligned}$$

Here C is a multi-hole context with $n \geqslant 0$ holes. Using Lemma 5.2 we obtain ground terms $u_1,\ldots,u_n$ such that $u_i \rightarrow^*_{\mathcal{A}_2} q_i$ and $u \rightarrow^*_{\mathcal{B}_1} r_i$ for all $1 \leqslant i \leqslant n$. Define the term $u = C[u_1,\ldots,u_n]$. We have $u \rightarrow^*_{\mathcal{B}_1} C[r_1,\ldots,r_n] \rightarrow^*_{\mathcal{B}_1} q$ and thus $(s,u) \in L(\mathcal{P})$. Furthermore, $u \rightarrow^*_{\mathcal{A}_2} C[q_1,\ldots,q_n]$ and thus also $(u,t) \in L(\mathcal{G})$. Hence $(s,t) \in L(\mathcal{P}) \circ L(\mathcal{G})$.

For the converse direction, let $(s,t) \in L(\mathcal{P})$ and $(t,u) \in L(\mathcal{G})$. So $s \rightarrow^*_{\mathcal{A}_1} p$ and $t \rightarrow^*_{\mathcal{B}_1} q$ with $(p,q) \in Q_2$. Moreover, there exists a multi-hole context C with $n \geqslant 0$ holes, terms $t_1,\ldots,t_n,u_1,\ldots,u_n$, and states $r_1,\ldots,r_n$ such that $t = C[t_1,\ldots,t_n]$, $u = C[u_1,\ldots,u_n]$, and $t_i \rightarrow^*_{\mathcal{A}_2} r_i$ and $u_i \rightarrow^*_{\mathcal{B}_2} r_i$ for all $1 \leqslant i \leqslant n$. The sequence $t \rightarrow^*_{\mathcal{B}_1} q$ can be written as $t = C[t_1,\ldots,t_n] \rightarrow^*_{\mathcal{B}_1} C[q_1,\ldots,q_n] \rightarrow^*_{\mathcal{B}_1} q$ for some states $q_1,\ldots,q_n$. By Lemma 5.2, $r_i \rightarrow q_i$ is a transition rule in $\Delta_\epsilon(\mathcal{A}_2,\mathcal{B}_1)$. Hence

$$\begin{aligned} u = C[u_1,\ldots,u_n] &\rightarrow^*_{\mathcal{B}_2} C[r_1,\ldots,r_n] \\ &\rightarrow^*_{\Delta_\epsilon(\mathcal{A}_2,\mathcal{B}_1)} C[q_1,\ldots,q_n] \rightarrow^*_{\mathcal{B}_1} q \end{aligned}$$

and thus $(s,u) \in L(\mathcal{P}')$ as desired. □

The construction in the above proof gives rise to a modified composition operation $\mathbin{\widehat{\circ}}$ on anchored GTT relations $\bowtie_1$ and $\bowtie_2$:

$$\bowtie_1 \mathbin{\widehat{\circ}} \bowtie_2 \;=\; \bowtie_1 \circ (\bowtie_2)^{\geqslant}_{\geqslant} \cup (\bowtie_1)^{\geqslant}_{\geqslant} \circ \bowtie_2$$

Here $\circ$ denotes standard relational composition. The construction $L(\mathcal{P}) \times L(\mathcal{G}) \mapsto L(\mathcal{P}')$ in the proof of Lemma 5.8 and its symmetric counterpart $L(\mathcal{G}) \times L(\mathcal{P}) \mapsto L(\mathcal{P}')$ in connection with Lemma 5.7 ensure that $\bowtie_1 \mathbin{\widehat{\circ}} \bowtie_2$ is an anchored GTT relation. We have $(\bowtie_1 \mathbin{\widehat{\circ}} \bowtie_2)^{\geqslant}_{\geqslant} = (\bowtie_1)^{\geqslant}_{\geqslant} \circ (\bowtie_2)^{\geqslant}_{\geqslant}$.

GTT relations are closed under transitive closure, which is the reason they were developed in the first place, but the construction is different from the one for anchored GTT relations and the correctness proof is considerably more involved (cf. [3, 8]). The construction in [8] employs the set $\Delta_+(\mathcal{A},\mathcal{B})$ consisting of ϵ-transitions $p \rightsquigarrow q$ that are computed by the inference rules in Figure 3. The transitive closure of a GTT relation $L(\mathcal{G})$ is then accepted by the GTT $\mathcal{G}_+ = (\mathcal{A}_+,\mathcal{B}_+) = (\mathcal{A} \cup \Delta_+(\mathcal{B},\mathcal{A}), \mathcal{B} \cup \Delta_+(\mathcal{A},\mathcal{B}))$.

If we apply this construction to an anchored GTT $\mathcal{G}$ then the anchored GTT relation $L_a(\mathcal{G}_+)$ associated with $\mathcal{G}_+$ satisfies

$$L(\mathcal{G}_+) \;=\; L_a(\mathcal{G}_+)^{\geqslant}_{\geqslant} \;=\; (L_a(\mathcal{G})^{\geqslant}_{\geqslant})^+ \;=\; L(\mathcal{G})^+$$

So the transitive closure of a GTT relation $(L(\mathcal{G})^+)$ is the multi-hole context closure of the transitive closure of its anchored counterpart $(L_a(\mathcal{G}^+)^{\geqslant}_{\geqslant})$. This result will be explained below.

Lemma 5.9. *Let $\mathcal{G} = (\mathcal{A},\mathcal{B})$ be an anchored GTT. If $\mathcal{G}_+ = (\mathcal{A}_+,\mathcal{B}_+)$ then $L_a(\mathcal{G}_+) = L(\mathcal{G})^+ \circ L_a(\mathcal{G}) \circ L(\mathcal{G})^+$.*

The proof of Lemma 5.9 relies on two preliminary results. The first one is from [8, Theorem 4.7].

Lemma 5.10. *Let $\mathcal{G} = (\mathcal{A},\mathcal{B})$ be a GTT. Let $\mathcal{G}_+ = (\mathcal{A}_+,\mathcal{B}_+)$. If $s \rightarrow^*_{\mathcal{A}_+} q$ then $t \rightarrow^*_{\mathcal{A}} q$ for some ground term t with $(s,t) \in L(\mathcal{G})^+$.* □

Lemma 5.11. *Let $\mathcal{G} = (\mathcal{A},\mathcal{B})$ be a GTT. If $\mathcal{G}_+ = (\mathcal{A}_+,\mathcal{B}_+)$ then $\Delta_\epsilon(\mathcal{A}_+,\mathcal{B}_+) = \Delta_+(\mathcal{A},\mathcal{B})$.*

Proof. We first show $\Delta_\epsilon(\mathcal{A}_+,\mathcal{B}_+) \subseteq \Delta_+(\mathcal{A},\mathcal{B})$ via induction on the relation $\rightsquigarrow$ defined by the inference rules in Figure 1. We proceed by case analysis, so assume $(p,q) \in \Delta_\epsilon(\mathcal{A}_+,\mathcal{B}_+)$ is derived from a congruence step [c]. Hence we obtain $(p,q) \in \Delta_+(\mathcal{A},\mathcal{B})$ by a congruence step [c] of Figure 3, the fact that the constructions only add ϵ-transitions, and the induction hypothesis. Next assume that we derived $(q,r) \in \Delta_\epsilon(\mathcal{A}_+,\mathcal{B}_+)$ by an ϵ-step [a]. So $p \rightarrow_{\mathcal{A}_+} q$ and $p \rightsquigarrow r$. We have $\mathcal{A}_+ = \mathcal{A} \cup \Delta_+(\mathcal{B},\mathcal{A})$. The result trivially follows for

$$\frac{q \;{}_{\mathcal{A}}\!\leftarrow p \quad p \rightsquigarrow r}{q \rightsquigarrow r}\;[\mathrm{a}] \qquad \frac{p \rightsquigarrow q \quad q \rightarrow_{\mathcal{B}} r}{p \rightsquigarrow r}\;[\mathrm{b}] \qquad \frac{p \rightsquigarrow q \quad q \rightsquigarrow r}{p \rightsquigarrow r}\;[\mathrm{t]}$$

$$\frac{p \;{}_{\mathcal{A}}\!\leftarrow f(p_1,\ldots,p_n) \quad p_1 \rightsquigarrow q_1 \;\cdots\; p_n \rightsquigarrow q_n \quad f(q_1,\ldots,q_n) \rightarrow_{\mathcal{B}} q}{p \rightsquigarrow q}\;[\mathrm{c}]$$

Figure 3. $\Delta_+(\mathcal{A},\mathcal{B})$: ϵ-transitions for GTT transitive closure.

$p \rightarrow_{\mathcal{A}} q$. So let $(p,q) \in \Delta_+(\mathcal{B},\mathcal{A})$. Hence $(q,p) \in \Delta_+(\mathcal{A},\mathcal{B})$. The induction hypothesis yields $(p,r) \in \Delta_+(\mathcal{A},\mathcal{B})$ and therefore $(q,r) \in \Delta_+(\mathcal{A},\mathcal{B})$ using the transitivity rule [t]. The ϵ-step [b] case is obtained in the same way.

For the reverse inclusion we use induction on the relation $\rightsquigarrow$ defined by the inference rules in Figure 3 and argue in a similar fashion. Hence $\Delta_\epsilon(\mathcal{A}_+,\mathcal{B}_+) = \Delta_+(\mathcal{A},\mathcal{B})$. □

Proof of Lemma 5.9. First let $(s,t) \in L_a(\mathcal{G}_+)$. So there exists a state q such that $s \rightarrow^*_{\mathcal{A}_+} q$ and $t \rightarrow^*_{\mathcal{B}_+} q$. Lemma 5.10 yields a ground term u such that $u \rightarrow^*_{\mathcal{A}} q$ and $(s,u) \in L(\mathcal{G})^+$. Applied to $\mathcal{G}^- = (\mathcal{B},\mathcal{A})$, Lemma 5.10 yields a ground term v such that $v \rightarrow^*_{\mathcal{B}} q$ and $(t,v) \in L(\mathcal{G}^-)^+$. Hence $(u,v) \in L_a(\mathcal{G})$ and $(v,t) \in L(\mathcal{G})^+$. Consequently, $(s,t) \in L(\mathcal{G})^+ \circ L_a(\mathcal{G}) \circ L(\mathcal{G})^+$.

For the other direction we apply the modified composition operation $\mathbin{\widehat{\circ}}$ with $\bowtie_1 = \bowtie_2 = L_a(\mathcal{G}_+)$ and obtain

$$\begin{aligned} L_a(\mathcal{G}_+) \circ L(\mathcal{G}_+) \cup L(\mathcal{G}_+) \circ L_a(\mathcal{G}_+) &\subseteq L_a(G_+) \mathbin{\widehat{\circ}} L_a(G_+) \\ &= L_a(\mathcal{G}_+) \end{aligned}$$

with the help of Lemma 5.11. Note that we do not get equality, as one direction in the proof of Lemma 5.8 requires disjoint state sets. Since $L_a(\mathcal{G}) \subseteq L_a(\mathcal{G}_+)$ we also have

$$L_a(\mathcal{G}) \circ L(\mathcal{G}_+) \cup L(\mathcal{G}_+) \circ L_a(\mathcal{G}) \subseteq L_a(\mathcal{G}_+)$$

At this point we can use the following well-known result in Kleene algebra

$$A \subseteq X \;\wedge\; B \circ X \subseteq X \;\wedge\; X \circ C \subseteq X \implies B^* \circ A \circ C^* \subseteq X$$

with $A = L_a(\mathcal{G})$, $B = C = L(\mathcal{G})$, and $X = L_a(\mathcal{G}_+)$. Since $L(\mathcal{G})^* = L(\mathcal{G})^+$, we are done. □

The (constructive proof of) the above lemma gives rise to the following modified transitive closure operation $\widehat{+}$ on anchored GTT relations $\bowtie$:

$$\bowtie^{\widehat{+}} \;=\; (\bowtie^{\geqslant}_{\geqslant})^+ \circ \bowtie \circ (\bowtie^{\geqslant}_{\geqslant})^+$$ ☑

We have $(\bowtie^{\widehat{+}})^{\geqslant}_{\geqslant} = (\bowtie^{\geqslant}_{\geqslant})^+$.

Example 5.12. Computing $\mathcal{G}_+ = (\mathcal{A}_+,\mathcal{B}_+)$ for the GTT $\mathcal{G} = (\mathcal{A},\mathcal{B})$ of Example 4.6 adds the pairs of $\Delta_+(\mathcal{B},\mathcal{A}) = \{(4,0),(4,1),(1,0),(2,0),(3,0),(3,2)\}$ as ϵ-transitions to $\mathcal{A}$ and the pairs of $\Delta_+(\mathcal{A},\mathcal{B}) = \Delta_+(\mathcal{B},\mathcal{A})^-$ to $\mathcal{B}$. We have $(\mathrm{g}(\mathrm{a},\mathrm{b}),\mathrm{f}(\mathrm{b})) \in L_a(\mathcal{G}_+)$ as $\mathrm{g}(\mathrm{a},\mathrm{b}) \rightarrow^*_{\mathcal{A}_+} 3$ and $\mathrm{f}(\mathrm{b}) \rightarrow_{\mathcal{B}_+} \mathrm{f}(1) \rightarrow_{\mathcal{B}_+} \mathrm{f}(4) \rightarrow_{\mathcal{B}_+} 3$. The term pair $(\mathrm{f}(\mathrm{a}),\mathrm{f}(\mathrm{b}))$ does not belong to $L_a(\mathcal{G}_+)$.

The final operation on anchored GTT relations that we consider is complement. This requires the determinization of pair automata.

Lemma 5.13. *For every pair automaton $\mathcal{P} = (Q_2,\mathcal{A},\mathcal{B})$ there exist deterministic tree automata $\mathcal{A}'$ and $\mathcal{B}'$ and a binary relation Q_2^d such that $L(\mathcal{P}) = L(Q_2^d,\mathcal{A}',\mathcal{B}')$.* ☑

Proof. We use the subset construction to determinize $\mathcal{A}$ and $\mathcal{B}$ into equivalent deterministic tree automata $\mathcal{A}'$ and $\mathcal{B}'$. As the binary state relation we take $Q_2^d = \{(X,Y) \mid (p,q) \in Q_2$ for some $p \in X \subseteq Q_{\mathcal{A}}$ and $q \in Y \subseteq Q_{\mathcal{B}}\}$. We have $L(\mathcal{P}) = L(Q_2^d,\mathcal{A}',\mathcal{B}')$ by the correctness of the subset construction. □

Lemma 5.14. *Anchored GTT relations are effectively closed under complement.* ☑☑☑

Proof. Let $\mathcal{G}$ be an anchored GTT. According to Lemma 5.13 we may assume that $L(\mathcal{G})$ is accepted by a deterministic pair automaton $\mathcal{P} = (Q_2,\mathcal{A},\mathcal{B})$. Without loss of generality we may further assume that $\mathcal{A}$ and $\mathcal{B}$ do not share any state and are completely defined. It follows that $L(\mathcal{P})^c = (Q_2^c,\mathcal{A},\mathcal{B})$ where $Q_2^c = (Q_{\mathcal{A}} \times Q_{\mathcal{B}}) \setminus Q_2$. □

It is worth noting that GTT relations are *not* closed under complement [3, Exercise 3.4].

Example 5.15. For the pair automaton $\mathcal{P} = (Q_2,\mathcal{A},\mathcal{B})$ associated with the anchored GTT $\mathcal{G} = (\mathcal{A},\mathcal{B})$ of Example 4.6 we have $Q_2 = \{(1,1),(2,2),(3,3)\}$. Determinizing $\mathcal{A}$ yields the tree automaton $\mathcal{A}'$ with the following transition rules:

$$\mathrm{a} \rightarrow A \qquad \mathrm{f}(X) \rightarrow \begin{cases} C & \text{if } X = A \\ B & \text{otherwise} \end{cases}$$

$$\mathrm{b} \rightarrow B \qquad \mathrm{g}(X,Y) \rightarrow \begin{cases} D & \text{if } X = A \\ B & \text{otherwise} \end{cases}$$

for all $X,Y \in \{A,B,C,D\}$. Here $A = \{0,1\}$, $B = \{0\}$, $C = \{0,2\}$, and $D = \{0,3\}$. Next we rename $(1 \mapsto 5,\ 2 \mapsto 6,\ 3 \mapsto 7)$ the states of $\mathcal{B}$ and apply determinization to obtain the tree automaton $\mathcal{B}'$ consisting of the following transition rules:

$$\mathrm{a} \rightarrow E \quad \mathrm{b} \rightarrow F \quad \mathrm{f}(X) \rightarrow \begin{cases} G & \text{if } X = E \\ H & \text{otherwise} \end{cases} \quad \mathrm{g}(X,Y) \rightarrow H$$

for all $X,Y \in \{E,F,G,H\}$. Here $E = \{4\}$, $F = \{5,6\}$, $G = \{7\}$, and $H = \varnothing$. The transition rules for g are added to make

$\mathcal{B}'$ completely defined. Now the complement $L(\mathcal{G})^c$ of $L(\mathcal{G})$ is accepted by the pair automaton $(Q_2', \mathcal{A}', \mathcal{B}')$ with

$$Q_2' = (\{A, B, C, D\} \times \{E, F, G, H\}) \setminus \{(A, F), (C, F), (D, G)\}$$

The final closure property of anchored GTT relations that we mention is intersection.

Lemma 5.16. *Anchored GTT relations are effectively closed under intersection.*

Proof. This follows from Lemmata 5.7 and 5.14. □

A more efficient product construction, which avoids the subset construction of the complement, is easily conceived.

6 Regular Relations

We continue on the level of regular relations. The following lemma takes care of transforming anchored GTT relations into binary regular (i.e., RR_2) relations.

Lemma 6.1. *Every anchored GTT relation is an* RR_2 *relation.* ☑

Proof. The first step in the (formalized) proof of [8, Theorem 5.3], where it is shown that GTT relations are RR_2 relations, constructs an RR_2 automaton for a relation $\mathcal{G}_\epsilon$. Since $\mathcal{G}_\epsilon$ is an anchored GTT relation, the result follows. □

We illustrate the underlying (product) construction on our leading example.

Example 6.2. For the anchored GTT $\mathcal{G}$ of Example 4.6 we obtain the RR_2 automaton $(\{\mathsf{a}, \mathsf{b}, \mathsf{f}, \mathsf{g}\}^{(2)}, Q, Q_f, \Delta)$ with $Q = (\{0, 1, 2, 3, \bot\} \times \{1, 2, 3, 4, \bot\}) \setminus \{\bot\bot\}$, $Q_f = \{11, 22, 33\}$, and Δ consisting of the following transition rules:

$$\begin{array}{llll}
\mathsf{aa} \to 04 & \mathsf{ba} \to 04 & \mathsf{fa}(0\bot) \to 04 & \mathsf{a}\bot \to 0\bot \\
\mathsf{aa} \to 14 & \mathsf{bb} \to 01 & \mathsf{fa}(1\bot) \to 24 & \mathsf{b}\bot \to 0\bot \\
\mathsf{ab} \to 01 & \mathsf{bb} \to 02 & \mathsf{fb}(0\bot) \to 01 & \bot\mathsf{b} \to \bot1 \\
\mathsf{ab} \to 02 & \mathsf{af}(\bot4) \to 03 & \mathsf{fb}(0\bot) \to 02 & \mathsf{a}\bot \to 1\bot \\
\mathsf{ab} \to 11 & \mathsf{af}(\bot4) \to 13 & \mathsf{fb}(1\bot) \to 21 & \bot\mathsf{a} \to \bot4 \\
\mathsf{ab} \to 12 & \mathsf{bf}(\bot4) \to 03 & \mathsf{fb}(1\bot) \to 22 & \bot\mathsf{b} \to \bot2 \\
\multicolumn{2}{r}{\mathsf{ga}(0\bot, 0\bot) \to 04} & \multicolumn{2}{r}{\mathsf{gf}(04, 0\bot) \to 03} \\
\multicolumn{2}{r}{\mathsf{ga}(1\bot, 0\bot) \to 34} & \multicolumn{2}{r}{\mathsf{gf}(14, 0\bot) \to 33} \\
\multicolumn{2}{r}{\mathsf{gb}(0\bot, 0\bot) \to 01} & \multicolumn{2}{r}{\mathsf{g}\bot(0\bot, 0\bot) \to 0\bot} \\
\multicolumn{2}{r}{\mathsf{gb}(0\bot, 0\bot) \to 02} & \multicolumn{2}{r}{\mathsf{g}\bot(1\bot, 0\bot) \to 3\bot} \\
\multicolumn{2}{r}{\mathsf{gb}(1\bot, 0\bot) \to 31} & \multicolumn{2}{r}{\mathsf{f}\bot(0\bot) \to 0\bot} \\
\multicolumn{2}{r}{\mathsf{gb}(1\bot, 0\bot) \to 32} & \multicolumn{2}{r}{\mathsf{f}\bot(1\bot) \to 2\bot} \\
\multicolumn{2}{r}{\bot\mathsf{f}(\bot4) \to \bot3} & \mathsf{ff}(04) \to 03 & \mathsf{ff}(14) \to 23
\end{array}$$

We have

$$\begin{aligned}\langle \mathsf{g}(\mathsf{a}, \mathsf{f}(\mathsf{b})), \mathsf{f}(\mathsf{a})\rangle = \mathsf{gf}(\mathsf{aa}, \mathsf{f}\bot(\mathsf{b}\bot)) &\to_\Delta^* \mathsf{gf}(14, \mathsf{f}\bot(0\bot)) \\ &\to_\Delta \mathsf{gf}(14, 0\bot) \to_\Delta 33\end{aligned}$$

The various context closure operations are taken care of in the following general result.

Lemma 6.3. *If* $\bowtie$ *is an* RR_2 *relation then* $\bowtie_p^n$ *is an* RR_2 *relation, for all* $n \in \{\geqslant, 1, >\}$ *and* $p \in \{\geqslant, \epsilon, >\}$. ☑☑☑☑☑☑☑☑☑

Proof. Let $\mathcal{A} = (\mathcal{F}^{(2)}, Q, Q_f, \Delta)$ be the RR_2 automaton that accepts $\bowtie$. We add two new states $\star$ and $\checkmark$. In the former the encoding of the identity relation on ground terms will be accepted. The latter will serve as the unique final state. This is achieved by extending Δ with the transitions $ff(\star, \ldots, \star) \to \star$ for every $f \in \mathcal{F}$ and $q \to \checkmark$ for every $q \in Q_f$. The resulting automaton $\mathcal{A}' = (\mathcal{F}^{(2)}, Q \cup \{\checkmark, \star\}, \{\checkmark\}, \Delta')$ is equivalent to $\mathcal{A}$ and the starting point for the various context closure operations. We present a few illustrative cases.

- For instance, for $n = 1$ and $p = \geqslant$ we extend Δ with all rules of the form $ff(\star, \ldots, \star, \checkmark, \star, \ldots, \star) \to \checkmark$.
- For $n = p = \geqslant$ we add $ff(q_1, \ldots, q_n) \to \checkmark$ for all $f \in \mathcal{F}$ and $q_1, \ldots, q_n \in \{\checkmark, \star\}$.
- For $p = >$ we need a new final state $\checkmark'$ to ensure that the surrounding context is non-empty:

  $$ff(\star, \ldots, \star, \checkmark, \star, \ldots, \star) \to \checkmark'$$
  $$ff(\star, \ldots, \star, \checkmark', \star, \ldots, \star) \to \checkmark'$$

  This is sufficient for $n = 1$. For $n = >$ we add the single ϵ-transition $\checkmark \to \star$ and for $n = \geqslant$ we additionally add a new final state $\star'$ together transition rules ensuring that the accepted relation is reflexive:

  $$ff(\star', \ldots, \star') \to \star'$$

Full details can be found in the formalization. □

Example 6.4. The following transition rules are added to the RR_2 automaton of Example 6.2 to model the relation $L_a(\mathcal{G})_>^\geqslant = \dot{\to}_{>\epsilon}$:

$$\begin{array}{llllll}
\mathsf{aa} \to \star & 11 \to \checkmark & \mathsf{ff}(\checkmark) \to \checkmark' & \mathsf{ff}(\checkmark') \to \checkmark' \\
\mathsf{bb} \to \star & 22 \to \checkmark & \mathsf{gg}(\checkmark, \star) \to \checkmark' & \mathsf{gg}(\checkmark', \star) \to \checkmark' \\
\mathsf{ff}(\star) \to \star & 33 \to \checkmark & \mathsf{gg}(\star, \checkmark) \to \checkmark' & \mathsf{gg}(\star, \checkmark') \to \checkmark' \\
 & \checkmark \to \star & \mathsf{gg}(\star, \star) \to \star &
\end{array}$$

The encoding of the term pair $(\mathsf{g}(\mathsf{f}(\mathsf{a}), \mathsf{f}(\mathsf{a})), \mathsf{g}(\mathsf{b}, \mathsf{f}(\mathsf{b})))$ is accepted:

$$\begin{aligned}\mathsf{gg}(\mathsf{fb}(\mathsf{a}\bot), \mathsf{ff}(\mathsf{ab})) &\to \mathsf{gg}(\mathsf{fb}(1\bot), \mathsf{ff}(11)) \to \mathsf{gg}(22, \mathsf{ff}(\checkmark)) \\ &\to \mathsf{gg}(\checkmark, \checkmark') \to \mathsf{gg}(\star, \checkmark') \to \checkmark'\end{aligned}$$

In [11] formalizations of the following results are reported. The first result is well-known. The formalization is based on the direct construction by Comon [2]. The second result goes back to a technical report by Dauchet and Tison [7].

Theorem 6.5. *The sets* $\mathsf{NF}(\mathcal{R})$ *for every left-linear TRS* $\mathcal{R}$ *and* $\mathsf{INF}_\bowtie$ *for every* RR_2 *relation* $\bowtie$ *are regular.* □

Table 1. Binary predicates as RR_2 relations.

$\to$	$= (\to_\epsilon)^1_{\geqslant}$	$\leftarrow$	$= ((\to_\epsilon)^1_{\geqslant})^-$
$\to_\epsilon$	$= (\to_\epsilon)^1_\epsilon$	$\to^+$	$= ((\to_\epsilon)^{\widehat{+}})^>_{\geqslant}$
$\to_{>\epsilon}$	$= (\to_\epsilon)^1_>$	$\to^*_{>\epsilon}$	$= ((\to_\epsilon)^{\widehat{+}})^{\geqslant}_>$
$\twoheadrightarrow$	$= (\to_\epsilon)^{\geqslant}_{\geqslant}$	$\to^*$	$= ((\to_\epsilon)^{\widehat{+}})^{\geqslant}_{\geqslant}$
$\to^+_\epsilon$	$= ((\to_\epsilon)^+)^1_\epsilon$	$\leftrightarrow^*$	$= (((\to_\epsilon)^- \cup \to_\epsilon)^{\widehat{+}})^{\geqslant}_{\geqslant}$
$\leftrightarrow$	$= ((\to_\epsilon)^- \cup \to_\epsilon)^1_{\geqslant}$	$\downarrow$	$= ((\to_\epsilon)^{\widehat{+}} \mathbin{\widehat{\circ}} (\to_\epsilon^-)^{\widehat{+}})^{\geqslant}_{\geqslant}$
$\to^!$	$= ((\to_\epsilon)^{\widehat{+}})^{\geqslant}_{\geqslant} \cap (\mathcal{T}(\mathcal{F}) \times \mathsf{NF})$		

We present two more operations that turn regular sets into RR_2 relations. Here $=_T$ consists of all pairs (t, t) with $t \in T$. The easy proofs are omitted.

Lemma 6.6. *If $T \subseteq \mathcal{T}(\mathcal{F})$ is a regular set of ground terms then $T \times T$ and $=_T$ are RR_2 relations.* ☑☑

We summarize the results obtained so far by listing all formalized closure operations for the predicates in the first-order theory of rewriting by means of the grammar below. Here A are anchored GTT relations, R are RR_2 relations, and T are regular sets of ground terms:

$$A ::= \to_\epsilon \mid A^- \mid A \cup A \mid A^+ \mid A^{\widehat{+}} \mid A \circ A \mid A \mathbin{\widehat{\circ}} A \mid A^c \mid A \cap A$$
$$R ::= A \mid R^n_p \mid R \cup R \mid R \cap R \mid R^- \mid T \times T \mid =_T$$
$$T ::= \mathcal{T}(\mathcal{F}) \mid \mathsf{NF} \mid \mathsf{INF}_R \mid T \cup T \mid T \cap T \mid T^c \mid \pi_1(R) \mid \pi_2(R)$$
$$n ::= \geqslant \mid 1 \mid > \qquad p ::= \geqslant \mid \epsilon \mid >$$

In Table 1 we show how some of the binary predicates in the first-order theory of rewriting are represented as RR_2 relations using the above constructs.

The logical structure of formulas in the first-order theory of rewriting is taken care of by the closure operations on regular relations mentioned in Section 2.3. These have been formalized in the setting of [8]. However, since Isabelle's code generation facility cannot automatically compute the sets used in the underlying automata constructions, the formalization of projection and cylindrification in [8] is not executable. We present an executable formalization of these results, which builds lists of rules and epsilon transition directly.

Theorem 6.7. *The class of regular relations is effectively closed under projection, cylindrification, and permutation.* ☑☑

7 Example

Stemming from the constructions of $(\to_\epsilon)^n_p$, relations such as $\to^*_{>\epsilon}$ can be used to model properties which were not expressible in FORT. One example of such a formula is the *root-stability* predicate $\mathsf{RS}(t) \iff \forall u\,(t \to^*_{>\epsilon} u \implies \neg\,\exists v\,(u \to_\epsilon v))$, which can be modeled as described below.

We construct automata for the subterms of the formula in a bottom up fashion. From Lemma 4.5 it follows that for any linear variable-separated TRS we can construct an anchored GTT G_1 accepting the language:

$$\text{GTT } G_1 \qquad L_a(G_1) = \{\,(u, v) \mid u \to_\epsilon v\,\}$$

Since anchored GTT relations are also RR_2 relations we can construct an equivalent RR_2 automaton A_1:

$$\mathsf{RR}_2\ A_1 = (G_1)^1_\epsilon \qquad L(A_1) = \{\,\langle u, v\rangle \mid u \to_\epsilon v\,\}$$

The second projection (Π_2) to construct A_2 followed by the complement construction results in the RR_1 automaton A_3, which represents the right-hand side of the implication:

$$\mathsf{RR}_1\ A_2 = \Pi_2(A_1) \qquad L(A_2) = \{\,\langle u\rangle \mid \exists v\,(u \to_\epsilon v)\,\}$$
$$\mathsf{RR}_1\ A_3 = (A_2)^c \qquad L(A_3) = \{\,\langle u\rangle \mid \neg\,\exists v\,(u \to_\epsilon v)\,\}$$

Reusing the GTT G_1 with closure under $\widehat{+}$ followed by an application of Lemma 6.3, the RR_2 automaton A_4 is constructed, representing the left-hand side of the implication:

$$\text{GTT } G_2 = (G_1)^{\widehat{+}} \qquad L_a(G_2) = \{\,(u, v) \mid u \to^* \cdot \to_\epsilon \cdot \to^* v\,\}$$
$$\mathsf{RR}_2\ A_4 = (G_2)^{\geqslant}_> \qquad L(A_4) = \{\,\langle t, u\rangle \mid t \to^*_{>\epsilon} u\,\}$$

To then represent the implication we first take the complement of A_4, resulting in A_5. Before the conjunction in $\neg\,(t \to^*_{>\epsilon} u) \vee \neg\,\exists v\,(u \to_\epsilon v)$ can be constructed, the arities of the RR_1 automaton A_3 and RR_2 automaton A_5 have to match. With this goal A_3 is cylindrified (C_1) to construct the RR_2 A_6. Here care has to be taken that not only the arities match, but also that terms, taking the place of variables shared by both formulas, have to be at the same position i in the encoding $\langle t_1, \ldots, t_i, \ldots, t_n\rangle$ of both automata. After this, the union with A_5 results in the RR_2 automaton A_7 that models modeling the implication:

$$\mathsf{RR}_2\ A_5 = (A_4)^c \qquad L(A_5) = \{\,\langle t, u\rangle \mid \neg\,(t \to^*_{>\epsilon} u)\,\}$$
$$\mathsf{RR}_2\ A_6 = C_1(A_3) \qquad L(A_6) = \{\,\langle t, u\rangle \mid \neg\,\exists v\,(u \to_\epsilon v)\,\}$$
$$\mathsf{RR}_2\ A_7 = A_5 \cup A_6$$
$$L(A_7) = \{\,\langle t, u\rangle \mid \neg\,(t \to^*_{>\epsilon} u) \vee \neg\,\exists v\,(u \to_\epsilon v)\,\}$$
$$= \{\,\langle t, u\rangle \mid t \to^*_{>\epsilon} u \implies \neg\,\exists v\,(u \to_\epsilon v)\,\}$$

Similarly to the implication, the universal quantifier is not directly modeled. To match the currently constructed formula to the logically equivalent formula

$$\neg\,\exists u\,(\neg\,(t \to^*_{>\epsilon} u \implies \neg\,\exists v\,(u \to_\epsilon v)))$$

the complement operation followed by the first projection and another complement operation are used. This results in

the RR_1 automaton A_{10}:

$$\mathsf{RR}_2\ A_8 = (A_7)^c$$
$$L(A_8) = \{\langle t, u\rangle \mid \neg\,(t \to_{>\epsilon}^* u \implies \neg\,\exists v\,(u \to_\epsilon v))\}$$
$$\mathsf{RR}_1\ A_9 = \Pi_1(A_8)$$
$$L(A_9) = \{\langle t\rangle \mid \exists u\,(\neg\,(t \to_{>\epsilon}^* u \implies \neg\,\exists v\,(u \to_\epsilon v)))\}$$
$$\mathsf{RR}_1\ A_{10} = (A_9)^c$$
$$L(A_{10}) = \{\langle t\rangle \mid \forall u\,(t \to_{>\epsilon}^* u \implies \neg\,\exists v\,(u \to_\epsilon v))\}$$

As can be clearly seen, the RR_1 automaton A_{10} accepts the language $L(A_{10}) = \mathsf{RS}$.

This example also shows how automata relate to individual subformulas, and how the closure properties can be used in a stepwise construction. The approach of relating formulas to automata, and annotating individual steps by the used properties, will be adapted into a formal certificate language. In turn this can be checked by a trustworthy tool, code generated from the executable formalization. This is ongoing work.

8 Formalization

In this section we discuss some aspects of our formalization. Since we deal exclusively with finite automata over finite terms, we decided to use *finite sets* as our foundation:

typedef *'a fset* = {*A* :: *'a set*. *finite A*}
morphisms *fset Abs_fset* **by** *auto*
setup_lifting *type_definition_fset*

The *fset* type is part of the standard HOL library[2] but several important operations and results on relations are absent, including transitive closure and relational composition. We addressed this issue with the help of the lifting and transfer package by Huffman and Kunčar [9]. This package provides a modular theory that provides the functionality *lift_definition* to lift definitions of related types, in our setting from *set* to *fset*. For this purpose, definitions must be shown to preserve finiteness. Most of the required constructions and lemmata could without further ado be converted with the help of this package. Problems emerged with the lifting of $\bigcup$ the big union on sets. Most lemmata for $\bigcup$ were derived from the complete lattice type class:

class *complete_lattice* = *lattice* + *Inf* + *Sup* + *bot* + *top* +
assumes *Inf_lower*: $x \in A \Longrightarrow \bigsqcap A \leq x$
and *Inf_greatest*: $(\bigwedge x.\ x \in A \Longrightarrow z \leq x) \Longrightarrow z \leq \bigsqcap A$
and *Sup_upper*: $x \in A \Longrightarrow x \leq \bigsqcup A$
and *Sup_least*: $(\bigwedge x.\ x \in A \Longrightarrow x \leq z) \Longrightarrow \bigsqcup A \leq z$
and *Inf_empty* [*simp*]: $\bigsqcap \{\} = \top$
and *Sup_empty* [*simp*]: $\bigsqcup \{\} = \bot$

However, finite sets in general do not possess a $\top$ element and therefore cannot be made an instance of this class. We solved this in an ad-hoc fashion by reproving the required lemmata for *fset*'s big union. A better solution would be to split the class into more fine-grained parts, so that lemmata not requiring a $\top$ element can be used independently. We intend do this in the near future.

Tree automata in our setting are represented as follows:

datatype (*'q*, *'f*) *ta_rule* = *TA_rule 'f* (*'q list*) *'q*
datatype (*'q*, *'f*) *ta* =
TA ((*'q*, *'f*) *ta_rule fset*) ((*'q* × *'q*) *fset*)

The states of the tree automaton are left implicit as they can be computed from the transition rules and epsilon transitions. Since we use finite sets, it follows that the set of states of a tree automaton is always finite. Note that we did not include final states in our definition. This allows us to create more general lemmata for languages associated with tree automata. However, we provide a wrapper called *reg* that deals with final states:

datatype (*'q*, *'f*) *reg* = *Reg* (*'q fset*) ((*'q*, *'f*) *ta*)

This is more convenient when dealing with tree automata constructions which work on final states directly.

Another reason for adopting the finite set representation for tree automata is to ensure that the formalization more accurately reflects the results on tree automata from the literature. Additionally, it allows more unconditional simplification lemmata. One of these is the correctness of the relabeling function which maps the states of a tree automaton to natural numbers without changing the accepted language:

lemma *gta_lang* (*relabel_Q*$_f$ *Q A*) (*relabel_ta A*) =
gta_lang Q A

The concrete implementation is listed in Figure 4. The function *map_fset_to_nat* is injective, a necessary condition for preserving the accepted language, only if sets are finite.

It is worth mentioning the tree automata formalization by Lammich [10], which covers basic results on tree automata. It does not include GTTs, RR_n automata, and the various context closures of regular languages, which is the reason why we decided to use the IsaFoR formalization of tree automata as our base. It covers all results in [10] and allows to reason about mixed terms (i.e. terms in $\mathcal{T}(\mathcal{F}, \mathcal{Q})$). We define (anchored) ground tree transducers

type_synonym (*'q*, *'f*) *gtt* = (*'q*, *'f*) *ta* × (*'q*, *'f*) *ta*

and their respective languages over ground terms

inductive *gtt_accept* :: (*'q*, *'f*) *gtt* ⇒ *'f gterm*
⇒ *'f gterm* ⇒ *bool* **for** *G* **where**
gtt_accept G t t
| *q* |∈| *gta_der* (*fst G*) *s* ⟹ *q* |∈| *gta_der* (*snd G*) *t*
⟹ *gtt_accept G s t*
| *length ss* = *length ts* ⟹ ∀ *i* < *length ss*. *gtt_accept G*
(*ss* ! *i*) (*ts* ! *i*) ⟹ *gtt_accept G* (*GFun f ss*) (*GFun f ts*)

definition *agtt_lang* :: (*'q*, *'f*) *gtt* ⇒ *'f gterm rel* **where**
agtt_lang G = {(*t*, *u*) |*t u q*. *q* |∈| *gta_der* (*fst G*) *t* ∧

[2] http://isabelle.in.tum.de/library/HOL/HOL-Library/FSet.html

```
lift_definition sorted_list_of_fset :: ('a :: linorder) fset ⇒ 'a list is sorted_list_of_set

definition map_fset_to_nat :: ('a :: linorder) fset ⇒ 'a ⇒ nat where
  map_fset_to_nat X = (λx. the (mem_idx x (sorted_list_of_fset X)))

definition relabel_ta :: ('q :: linorder, 'f) ta ⇒ (nat, 'f) ta where
  relabel_ta 𝒜 = fmap_states_ta (map_fset_to_nat (𝒬 𝒜)) 𝒜

definition relabel_Q_f :: ('q :: linorder) fset ⇒ ('q :: linorder, 'f) ta ⇒ nat fset where
  relabel_Q_f Q 𝒜 = map_fset_to_nat (𝒬 𝒜) |`| (Q |∩| 𝒬 𝒜)
```

Figure 4. Relabeling of tree automata.

```
  q |∈| gta_der (snd 𝒢) u}
```

We now turn our attention to the closure properties of (anchored) GTTs. Most of these require that the state sets of the participating GTTs are disjoint. Proving properties that require such invariants involves additional work. Hence we removed the necessity of this invariant by the equivalent (Lemma 4.4) definition of pair automata (Definition 4.3).

```
definition pair_at_lang :: ('q, 'f) gtt ⇒ ('q × 'q) fset ⇒
  'f gterm rel where
  pair_at_lang 𝒢 Q = {(s, t) | s t p q. q |∈| gta_der (fst 𝒢) s
    ∧ p |∈| gta_der (snd 𝒢) t ∧ (q, p) |∈| Q}
lemma agtt_lang 𝒢 =
  pair_at_lang 𝒢 (fId_on (gtt_interface 𝒢))
lemma pair_at_lang 𝒢 Q =
  agtt_lang (pair_at_to_agtt' 𝒢 Q)
```

This leads to shorter and more elegant proofs. Moreover, from the definition of agtt_lang it is not obvious that the language is closed under complement. Looking at the pair automata definition, it follows that the complement can be constructed via determinization and complement of tree automata.

Formalizing operations on multihole contexts proved to be challenging. Multihole contexts are defined as follows:

```
datatype ('f, 'v) mctxt = MVar 'v | MHole
  | MFun 'f (('f, 'v) mctxt list)
```

A multihole context is filled by partitioning the given term list depending on the number of holes:

```
fun num_holes :: ('f, 'v) mctxt ⇒ nat where
  num_holes (MVar _) = 0 |
  num_holes MHole = 1 |
  num_holes (MFun _ ctxts) = sum_list
    (map num_holes ctxts)

fun fill_holes :: ('f, 'v) mctxt ⇒ ('f, 'v) Term.term list
  ⇒ ('f, 'v) Term.term where
  fill_holes (MVar x) _ = Term.Var x |
  fill_holes MHole [t] = t |
  fill_holes (MFun f cs) ts = Fun f (map
    (λ i. fill_holes (cs ! i)
      (partition_by ts (map num_holes cs) ! i))
    [0 ..< length cs])
```

Many proofs require (de)composing multihole contexts, which is too complex for the automated tactics of Isabelle/HOL. This is due to the invariant, that the number of holes of a multihole context must match the length of the given list. Such proofs regularly exhibit similar structures. These can be captured and extracted using higher-order predicates. An example of such a higher-order predicate can be seen in the following lemma proving function closure over multihole context closed relations.

```
lemma (⋀f n. (f, n) ∈ ℱ ⟹ n ≠ 0 ⟹
  P (GMFun f (replicate n GMHole))) ⟹ (⋀C Ds.
    P C ⟹ num_gholes C = length Ds ⟹
    0 < num_gholes C ⟹ ∀ D∈set Ds. P D ⟹
    P (fill_gholes_gmctxt C Ds)) ⟹
  function_closed ℱ (gmctxtex_onp P ℛ)
```

This allows us to prove function closure for different relations, by reasoning solely over the predicate. This greatly improves the success rate of the automated proof tactics.

The rewrite relations are defined by restricting the relations from IsaFoR to ground terms:

```
definition grrstep :: ('f, 'v) trs ⇒ 'f gterm rel where
  grrstep ℛ = inv_image (rrstep ℛ) term_of_gterm

definition gnrrstep :: ('f, 'v) trs ⇒ 'f gterm rel where
  gnrrstep ℛ = inv_image (nrrstep ℛ) term_of_gterm

definition grstep :: ('f, 'v) trs ⇒ 'f gterm rel where
  grstep ℛ = inv_image (rstep ℛ) term_of_gterm

definition gpar_rstep :: ('f, 'v) trs ⇒ 'f gterm rel where
  gpar_rstep ℛ = inv_image (par_rstep ℛ) term_of_gterm
```

For a clear separation of the tree automata theory and the decision procedure for the first-order theory of rewriting, we abstract from the concrete constructions of relations. We achieve this by defining primitives directly and interpreting closure properties as operations on relations as seen in Figure 5. This setting allows us to prove the correctness of our decision procedure without any knowledge of tree automata. This means that we obtain a stronger result, namely

```
primrec eval_gtt_rel :: ('f × nat) set ⇒ ('f, 'v) trs list ⇒ ftrs gtt_rel ⇒ 'f gterm rel where
  eval_gtt_rel F Rs (ARoot is) = Restr (grrstep (is_to_trs Rs is)) (T_G F)
| eval_gtt_rel F Rs (GInv g) = prod.swap ‘ (eval_gtt_rel F Rs g)
| eval_gtt_rel F Rs (AUnion g1 g2) = (eval_gtt_rel F Rs g1) ∪ (eval_gtt_rel F Rs g2)
| eval_gtt_rel F Rs (ATrancl g) = (eval_gtt_rel F Rs g)^+
| eval_gtt_rel F Rs (AComp g1 g2) = (eval_gtt_rel F Rs g1) O (eval_gtt_rel F Rs g2)
| eval_gtt_rel F Rs (GTrancl g) = gtrancl_rel F (eval_gtt_rel F Rs g)
| eval_gtt_rel F Rs (GComp g1 g2) = gcomp_rel F (eval_gtt_rel F Rs g1) (eval_gtt_rel F Rs g2)

primrec eval_rr1_rel :: ('f × nat) set ⇒ ('f, 'v) trs list ⇒ ftrs rr1_rel ⇒ 'f gterm set
    and eval_rr2_rel :: ('f × nat) set ⇒ ('f, 'v) trs list ⇒ ftrs rr2_rel ⇒ 'f gterm rel where
  eval_rr1_rel F Rs R1Terms = (T_G F)
| eval_rr1_rel F Rs (R1Union R S) = (eval_rr1_rel F Rs R) ∪ (eval_rr1_rel F Rs S)
| eval_rr1_rel F Rs (R1Inter R S) = (eval_rr1_rel F Rs R) ∩ (eval_rr1_rel F Rs S)
| eval_rr1_rel F Rs (R1Diff R S) = (eval_rr1_rel F Rs R) − (eval_rr1_rel F Rs S)
| eval_rr1_rel F Rs (R1Proj n R) = (case n of 0 ⇒ fst ‘ (eval_rr2_rel F Rs R)
                                      | _ ⇒ snd ‘ (eval_rr2_rel F Rs R))
| eval_rr1_rel F Rs (R1NF is) = NF (Restr (grstep (is_to_trs Rs is)) (T_G F)) ∩ (T_G F)
| eval_rr1_rel F Rs (R1Inf R) = {s. infinite (eval_rr2_rel F Rs R `` {s})}
| eval_rr2_rel F Rs (R2GTT_Rel A W X) = lift_root_step F W X (eval_gtt_rel F Rs A)
| eval_rr2_rel F Rs (R2Inv R) = prod.swap ‘ (eval_rr2_rel F Rs R)
| eval_rr2_rel F Rs (R2Union R S) = (eval_rr2_rel F Rs R) ∪ (eval_rr2_rel F Rs S)
| eval_rr2_rel F Rs (R2Inter R S) = (eval_rr2_rel F Rs R) ∩ (eval_rr2_rel F Rs S)
| eval_rr2_rel F Rs (R2Diff R S) = (eval_rr2_rel F Rs R) − (eval_rr2_rel F Rs S)
| eval_rr2_rel F Rs (R2Comp R S) = (eval_rr2_rel F Rs R) O (eval_rr2_rel F Rs S)
| eval_rr2_rel F Rs (R2Diag R) = Id_on (eval_rr1_rel F Rs R)
| eval_rr2_rel F Rs (R2Prod R S) = (eval_rr1_rel F Rs R) × (eval_rr1_rel F Rs S)
```

Figure 5. Semantics of GTT, RR_1 and RR_2 relations.

any functions that can compute *eval_gtt_rel*, *eval_rr1_rel*, and *eval_rr2_rel* are suitable as a decision procedure for the first-order theory of rewriting. The equations in Table 1 are proved correct in lemmata of the following shape (☑):

```
lemma eval_rr2_rel F Rs (R2Step ts) =
  Restr (grstep (is_to_trs Rs ts)) (T_G F)
lemma eval_rr2_rel F Rs (R2StepEq ts) =
  Restr ((grstep (is_to_trs Rs ts))^=) (T_G F)
lemma eval_rr2_rel F Rs (R2ParStep ts) =
  Restr (gpar_rstep (is_to_trs Rs ts)) (T_G F)
```

The correctness of the tree automata constructions then follows from the correctness of the primitives $\rightarrow_\epsilon$ (*ARoot*, from Figure 5), $\mathcal{T}(\mathcal{F})$ (*R1Terms*), NF (*R1NF*), and INF (*R1Inf*) as well as correctness of the closure operations (☑).

Felgenhauer *et al.* [8, Section 7] pointed out the limitation of Isabelle's code generation mechanism to obtain executable code for inductively defined sets, and introduced a convenient abstract *Horn inference system* for sets that are finite. We use this framework to obtain executable code for the following constructions:

- reachable and productive states of a tree automaton, ☑☑☑☑
- rules and states of tree automata obtained by the subset construction, ☑☑☑
- epsilon transitions for the composition and transitive closure constructions of (anchored) GTTs,
- an inductive set needed for the tree automaton for the infinity predicate. ☑☑

We conclude this section by providing some statistics of our formalization in Table 2. The numbers in parentheses refer to re-formalizations (partly due to the new *fset* foundation for tree automata) of existing results reported in [8, 11]. The formalization is divided into the following components:

Utility files This material covers three parts. First, lemmata for various kinds of basic operations that were required in the formalization. For example, the image of a Cartesian product can be rewritten as an image over a tuple. Second, it deals with all missing concepts for finite set type (e.g., transitive closure). A third part concerns the saturation process described in [11].

Horn inference system This is the basic building block to obtain executable definitions. Felgenhauer *et al.* [8] provide a detailed explanation.

Table 2. Formalization data.

topics	lines	facts	defs
Utility files	616 (+1659)	47 (+183)	2 (+ 22)
Horn inference system	(+ 462)	(+ 39)	(+ 17)
Ground constructions	902 (+1939)	75 (+242)	6 (+ 39)
Regular relations	529 (+3495)	35 (+239)	9 (+ 56)
FORT	2563 (+ 655)	206 (+ 65)	37 (+ 6)
Implementation files	419 (+1237)	26 (+ 75)	12 (+ 32)
total	5029 (+9447)	389 (+843)	66 (+172)

Ground constructions This component is concerned with ground instances of terms, contexts and multi-hole contexts. Additionally, we provide higher-order properties to obtain results for closing relations under (multi-hole) contexts as described in Section 3.

Regular relations This part of the formalization deals with all topics concerning regular relations, as described in Sections 4, 5 and 6.

FORT Here we start the foundation of our ongoing work, to certify the first-order theory of rewriting for linear variable-separated TRSs. Currently it contains the anchored GTT construction of the root-step relation, the normal form automaton ported from [11], and various kinds of multi-hole closures of regular relations (cf. Lemma 6.3).

Implementation files This last part contains the required code equations for Isabelle's code generation facility to produce executable Haskell code from our constructions.

9 Conclusion and Future Work

In this paper we presented an executable formalization in Isabelle/HOL of a decision procedure for the first-order theory of rewriting of (finite) linear, variable separated TRSs. The decision procedure uses closure operations on anchored GTTs and RR_n automata. We are in the process of developing a certificate language that corresponds one-to-one to these operations. A reincarnation of FORT will then output certificates in this language and these certificates are subsequently verified by executable code that is generated from the formalization.

Upgrading FORT to support the full formalized first-order theory as described in this paper is another obvious task. In the other direction, [13] describes useful extensions to the first-order theory for which we would like to provide executable formalizations and corresponding primitives in the certificate language. Specifically, support for multiple TRSs, which is used to express properties like commutation, and support for many-sorted TRSs will be added.

Acknowledgments

This work is supported by FWF (Austrian Science Fund) project P30301.

References

[1] Franz Baader and Tobias Nipkow. 1998. *Term Rewriting and All That.* Cambridge University Press. https://doi.org/10.1017/CBO9781139172752

[2] Hubert Comon. 2000. Sequentiality, Monadic Second-Order Logic and Tree Automata. *Information and Computation* 157, 1-2 (2000), 25–51. https://doi.org/10.1006/inco.1999.2838

[3] Hubert Comon, Max Dauchet, Rémi Gilleron, Christof Löding, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. 2008. Tree Automata Techniques and Applications. http://tata.gforge.inria.fr/

[4] Max Dauchet, Thierry Heuillard, Pierre Lescanne, and Sophie Tison. 1990. Decidability of the Confluence of Finite Ground Term Rewriting Systems and of Other Related Term Rewriting Systems. *Information and Computation* 88, 2 (1990), 187–201. https://doi.org/10.1016/0890-5401(90)90015-A

[5] Max Dauchet and Sophie Tison. 1985. Decidability of Confluence for Ground Term Rewriting Systems. In *Proc. 5th Fundamentals of Computation Theory (Lecture Notes in Computer Science, Vol. 199).* 80–84. https://doi.org/10.1007/BFb0028794

[6] Max Dauchet and Sophie Tison. 1990. The Theory of Ground Rewrite Systems is Decidable. In *Proc. 5th IEEE Symposium on Logic in Computer Science.* 242–248. https://doi.org/10.1109/LICS.1990.113750

[7] Max Dauchet and Sophie Tison. 1990. *The Theory of Ground Rewrite Systems is Decidable (Extended Version).* Technical Report I.T. 197. LIFL.

[8] Bertram Felgenhauer, Aart Middeldorp, T. V. H. Prathamesh, and Franziska Rapp. 2019. A Verified Ground Confluence Tool for Linear Variable-Separated Rewrite Systems in Isabelle/HOL. In *Proc. 8th ACM SIGPLAN International Conference on Certified Programs and Proofs*, Assia Mahboubi and Magnus O. Myreen (Eds.). 132–143. https://doi.org/10.1145/3293880.3294098

[9] Brian Huffman and Ondřej Kunčar. 2013. Lifting and Transfer: A Modular Design for Quotients in Isabelle/HOL. In *Proc. 3th ACM SIGPLAN International Conference on Certified Programs and Proofs (Lecture Notes in Computer Science, Vol. 8307).* 131–146. https://doi.org/10.1007/978-3-319-03545-1_9

[10] Peter Lammich. 2009. Tree Automata. *Archive of Formal Proofs* (Nov. 2009). https://isa-afp.org/entries/Tree-Automata.html, Formal proof development.

[11] Alexander Lochmann and Aart Middeldorp. 2020. Formalized Proofs of the Infinity and Normal Form Predicates in the First-Order Theory

of Rewriting. In *Proc. 26th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (Lecture Notes in Computer Science, Vol. 12079)*, Armin Biere and Dave Parker (Eds.). 178–194. https://doi.org/10.1007/978-3-030-45237-7_11

[12] Franziska Rapp and Aart Middeldorp. 2016. Automating the First-Order Theory of Left-Linear Right-Ground Term Rewrite Systems. In *Proc. 1st International Conference on Formal Structures for Computation and Deduction (Leibniz International Proceedings in Informatics, Vol. 52)*, Delia Kesner and Brigitte Pientka (Eds.). 36:1–36:12. https://doi.org/10.4230/LIPIcs.FSCD.2016.36

[13] Franziska Rapp and Aart Middeldorp. 2018. FORT 2.0. In *Proc. 9th International Joint Conference on Automated Reasoning (LNAI, Vol. 10900)*, Didier Galmiche, Stephan Schulz, and Roberto Sebastiani (Eds.). 81–88. https://doi.org/10.1007/978-3-319-94205-6_6

[14] René Thiemann and Christian Sternagel. 2009. Certification of Termination Proofs using CeTA. In *Proc. 22nd International Conference on Theorem Proving in Higher Order Logics (Lecture Notes in Computer Science, Vol. 5674)*, Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel (Eds.). 452–468. https://doi.org/10.1007/978-3-642-03359-9_31