Cumulative Habilitation Thesis

Challenges in Automation of Rewriting

Harald Zankl

Innsbruck, October 10, 2014

Institute of Computer Science

University of Innsbruck

für Michaela

Abstract

This thesis contains selected contributions investigating various aspects regarding the automated analysis of rewrite systems. The main topics of confluence, complexity, termination, and automation itself are addressed. For confluence and complexity analysis, frameworks are presented that admit the combination of different methods, resulting in significant gains in power. Termination criteria whose automation has been open for many years (interpretations using ordinals or exponentiation functions) are considered as well as the uncurrying transformation, which improves the power of first-order termination tools for applicative systems. Since verification is an essential part of automation we discuss our formalization of the decreasing diagrams method in the interactive theorem prover Isabelle. To highlight the results of successful automation efforts we present the tools CSI and KBCV.

Acknowledgments

I am grateful to my co-authors Aart, Bertram, Friedl, Martin, Nao, Sarah, and Thomas for the joyful and enlightening collaboration we had during the conception of the publications that contribute to this habilitation thesis.

Actually, all members of the Computational Logic group have provided a very fertile ground for research. Especially, I want to mention Aart with his incredible knowledge in rewriting and Christian and René for their formalization of rewriting.

Large parts of this work have been supported by FWF (Austrian Science Fund) projects P18763 and P22467.

Finally I thank my friends and family – in particular my companion in life Michaela – for providing the distraction that had been necessary to gain distance, change focus, and obtain a different view on problems that seemed unsolvable beforehand.

Contents

Part I

Preface

There is nothing as practical as a good theory.

Kurt Lewin (1890–1947)

1. Introduction

Term rewriting is a Turing complete model of computation [86]. Equipped with clear semantics it provides the theoretical foundation of declarative programming paradigms. Furthermore, rewriting techniques have found their way into computer algebra systems such as Mathematica [195] and interactive theorem provers like Isabelle [133]. One current trend is to employ rewriting based tools for termination analysis of real world programs written in C [169], Haskell [57], Java [146], or Prolog [158].

This thesis discusses (some) challenges which arise in the automation of rewriting. Here automation means more than just an (efficient) implementation. For a successful automation often the underlying theory must be adjusted accordingly or even developed from scratch to culminate in powerful tools. Furthermore verification also plays an essential role regarding automation because software should behave as described in its specification. So we propose the following equation:

$$\text{automation} = \text{theory} + \text{implementation} + \text{verification}$$

This thesis presents contributions to the above (three) aspects of automation for different properties of rewrite systems. In many crucial places relative rewriting turned out the appropriate concept to succeed.

Termination, being a central important property of programs, ensures that the computations performed by a program eventually yield a final result. However, programs exist that admit incredibly long computation sequences and so far automatic support to capture these systems has been lacking. While the Ackermann function cannot be bounded by a primitive recursive function it can be easily proved terminating by path orders [39, 94], which have been implemented successfully (see e.g. [33]). While primitive recursion is the upper bound for the multiset path order [81], multiple recursion is the theoretical limitation for recursive path orders [190], which has also been the bound for automatic tools [125] for a long time. This thesis describes automation of ordinal interpretations [178], a method that goes far beyond this bound. Furthermore, this effort has practical applications, e.g., it enables an automatic termination proof of Goodstein's theorem [63], which is not provable in Peano arithmetic [96]. The ideas employed to automate ordinal interpretations turned out to be helpful to solve another open challenge concerning automation. While the use of polynomials in termination proofs [112] is well-studied and nowadays strong methods for their implementation are known [36, 49], the situation has been different when considering exponentiation functions. Due to our contributions also exponentiation functions are now supported in automatic termination tools.

The second contribution on termination that has been selected for inclusion in this thesis is concerned with the applicability of first-order termination tools for functional programs. The use of higher-order functions can be modeled in applicative rewrite systems. While first-order termination tools nowadays are very powerful for the majority of systems, applicative rewrite systems lack sufficient structure to be exploited by the dependency pairs method [13, 60, 174], the underlying theoretical framework of these tools. The proposed technique of uncurrying is able to restore the term structure in applicative systems and significantly improves the performance of first-order termination tools on such problems. Moreover, our study of innermost rewriting and derivational complexity aims to further increase applicability of the uncurrying transformation.

As demonstrated by ordinal interpretations, termination alone is often not sufficient. For many practical applications it is also interesting to know how much resources a computation requires. Since recently some tools also support the calculation of upper bounds on the number of computation steps needed to produce a result. While originally upper bounds have been studied to infer the theoretical limits of a base method (see e.g. [77, 114, 190]), recently the focus has changed to restrict the base methods suitably in order to obtain polynomial upper bounds (cf. [15, 126]), which are of interest in the analysis of programs. To overcome the limitations of each single base method we have designed a framework—based on relative rewriting—that supports partial complexity proofs. As a consequence it also admits the combination of different base methods. Because of this combination "the whole is greater than the sum of the parts" since the interaction of different criteria facilitates more successful proofs compared to the setting when all methods are used stand-alone. Furthermore, partial complexity proofs are the key to appropriately address the fact that complexity is an optimization problem, i.e., the reported bounds should be as tight as possible.

Another challenge that has been solved was the automation of a powerful base method. By estimating the frequency and suitably restricting the absolute value of eigenvalues of (parametric) matrices, polynomial upper bounds can be inferred for matrix interpretations [44]. Our criterion strictly encompasses an earlier result [126], and, in contrast to [187] trivially extends to matrix interpretations over rational or real numbers. However, some challenges had to be solved to automate this criterion.

Even if programs terminate (in reasonable time) one further important property must be ensured. Think of the parallel execution of some computation where the result depends on the order in which the instructions are performed. Clearly also in the case of parallel execution the program should report a unique result, called unique normal form in the terminology of rewriting. While only few techniques are known that guarantee unique normal forms, many conditions have been proposed that ensure confluence, a strictly stronger property. However, apart from the cases of termination (where confluence is decidable [100]) or the sufficient condition of orthogonality [152], automatic tool support has been very limited and only in 2009 the first automatic confluence prover was announced (cf. [11]). Despite the presence of many different theoretical results that ensure confluence and the complete (for countable systems) method of decreasing diagrams [143] there has been no unifying framework that allows to combine partial confluence proofs so far. Based on the decreasing diagrams technique, we present the first step in this direction by proposing various labeling functions that can be combined lexicographically. By these means the power of different criteria can be mixed to establish decreasingness of local peaks and hence confluence. The method is fairly easy to implement—based on a (relative) termination prover—and results in a powerful approach for left-linear systems.

Rewrite tools are complex pieces of software and do contain bugs. One way to ensure a high level of certainty that these tools behave correctly is to check their output with the help of a trustable checker (certifier). Such trustable checkers can be obtained by formalizing the proofs of the underlying theorems in an interactive theorem prover, which facilitates the export of trustable executable check functions. While for termination proofs there already is a strong support in this direction (cf. [24, 35, 175, 176]), the picture looks different for confluence proofs (see [128]). Hence, as a major step we have formalized the decreasing diagrams technique in the theorem prover Isabelle [133]. This formalization, which is described in the thesis has already been used in the Confluence Competition 2014 [12] where confluence proofs via decreasing diagrams based on the rule labeling have been confirmed by the trustable checker CeTA.

The best design of a powerful rewrite tool is useless if the underlying implementation is not efficient. As a selection of efficiently coded rewrite tools we included the system descriptions of the confluence tool CSI[1] and the completion tool KBCV.[2] The latter, albeit designed for educational purposes, can compete with other state-of-the-art completion tools. For both tools the main conception has been our responsibility.

[1]`http://cl-informatik.uibk.ac.at/software/csi`

[2]`http://cl-informatik.uibk.ac.at/software/kbcv`

The remainder of this document is organized as follows. In Chapter 2 the contributions of this thesis are described, the main problems that have been solved are explained, and the main impact of the conducted research is envisaged. Further contributions, which are not included in the thesis are also discussed. In addition, an outlook for future directions of research is presented. Afterwards the selected papers are provided, grouped by topic: Confluence (Part II), complexity (Part III), termination (Part IV), and automation (Part V).

2. Contributions

This thesis contains contributions to four main streams in the research of rewriting: confluence, complexity, termination, and automation. For each of these themes two publications (addressing different problems and approaches) have been selected for inclusion with their significance and contributions described in the subsequent sections. The text of some conference papers (listed in gray) is not included since these are subsumed by significantly extended journal articles. At the end of this section we briefly discuss related work and sketch an outlook for future research.

2.1. Confluence

Publications

1. Harald Zankl, Bertram Felgenhauer, and Aart Middeldorp, Labelings for Decreasing Diagrams, Journal of Automated Reasoning, accepted for publication, to appear.
   doi: `10.1007/s10817-014-9316-y`
2. Harald Zankl, Bertram Felgenhauer, and Aart Middeldorp, Labelings for Decreasing Diagrams, In Proceedings of the 22nd International Conference on Rewriting Techniques and Applications (RTA 2011), Leibniz International Proceedings in Informatics 10, pp. 377–392, Schloss Dagstuhl, 2011.
   doi: `10.4230/LIPIcs.RTA.2011.377`
3. Harald Zankl, Confluence by Decreasing Diagrams – Formalized, In Proceedings of the 24th International Conference on Rewriting Techniques and Applications (RTA 2013), Leibniz International Proceedings in Informatics 21, pp. 352–367, Schloss Dagstuhl, 2013.
   doi: `10.4230/LIPIcs.RTA.2013.352`

History

Confluence holds for orthogonal systems [152] and is decidable in the presence of termination [100] or (strong) syntactic restrictions (see e.g. [62, 147]). For systems containing (left-)linear rules only, conditions on the critical pairs have been proposed and refined in [84, 137, 143, 148, 179]. The decreasing diagrams technique [143] provides a complete method (in the case of countable systems) for establishing confluence but its automation has been open for many years. By proposing the rule labeling (for linear systems) the first step towards automation was made in [138], and actual implementations of this criterion followed soon (see [4, 70, 71]). In addition, [4] extends the rule labeling to left-linear systems and [70, 71] employ labeling functions based on relative termination for left-linear systems. The significantly more challenging case to use decreasing diagrams for non-left-linear systems is the topic of [7, 97].

Formalizing confluence criteria has a long history in λ-calculus. Huet [85] proved a stronger variant of the parallel moves lemma in Coq. Isabelle was used in [132] to prove the Church-Rosser property of β, η, and $\beta\eta$. For β-reduction the standard Tait/Martin-Löf proof as well as Takahashi's proof [171] were formalized. The first mechanically verified proof of the Church-Rosser property of β-reduction was done using the Boyer-Moore theorem prover [160]. The formalization in Twelf [149] was used to formalize the confluence proof of a specific higher-order rewrite system in [168]. Regarding term rewriting the following has been achieved. Newman's lemma (for abstract rewrite systems) and Knuth and Bendix' critical pair theorem (for first-order rewrite systems) have been proved in [153] using ACL. An alternative proof of the latter in PVS, following the higher-order structure of Huet's proof, is presented in [51]. PVS is also used in the formalization of the lemmas of Newman and Yokouchi in [50]. Knuth and Bendix' criterion has also been formalized in Coq [37] and Isabelle [173].

Contributions

Publication 1 introduces a unifying approach to combine various different labeling functions (rule labeling, source labeling, redex labeling) for the decreasing diagrams technique using relative rewriting. It can be seen as an attempt to unify the findings in [4] (which is subsumed) and [71] (large parts of which are subsumed). Furthermore, new labeling functions (based on a measure of the contracted redex) are introduced. As one key ingredient to the power of the approach we present preconditions (based on relative termination) which ensure that any labeling function applicable to linear rewrite systems (such as the rule labeling) is also sound for left-linear systems. I contributed the overall design and implementation of the incremental labeling approach. Furthermore, I proposed to study the rule labeling in the case of parallel reduction, the technical details of which have been worked out by Bertram Felgenhauer, who also solved the challenges occurring in the implementation of parallel reduction.

Publication 3 is concerned with the soundness of the decreasing diagrams method and describes a formalization of the main results [138, 143] in the theorem prover Isabelle. As a fundamental result for confluence based on abstract rewrite systems, this method can be used as the key to obtain many confluence results for concrete rewrite systems. Hence it is an ideal candidate for a formalization in a theorem prover. The main obstacles when mechanizing textual proofs in theorem provers are often similar. Still, challenges remain to be solved. How to formalize all required notions? How to convince the theorem prover about gaps that are easily believed by humans? Which existing formalizations can be reused/adapted? In contrast to [143], which uses *infinite multisets*, to make the reasoning about sets, finite multisets, and sequences uniform we decided to clearly separate these three notions in the formalization. This allowed to reuse existing Isabelle theories. Still, we had to augment the available formalization of finite multisets in Isabelle with an alternative characterization of the multiset extension of an order. Besides, our formalization introduces a notion of labeled abstract rewriting. For the valley version of decreasing diagrams the proof of [143] could closely be followed in the formalization while for the conversion version some intermediate lemmata were necessary to formalize the approach proposed in [138] in Isabelle.

Impact

Due to the flexible combination of labeling functions our confluence prover CSI is the most powerful tool when restricted to the left-linear systems in the problem database used for the confluence competition [12] and only beaten by ACP [11] when also non-left-linear systems are considered. Hence one natural idea for future work is to extend the framework to labeling functions for non-left-linear systems. Another one is to incorporate sufficient criteria to exclude the need for some local peaks to be shown decreasing. Recent developments regarding this direction have been reported in [141].

As the decreasing diagrams technique is a fundamental result for confluence, its formalization has applications in two different directions. One direction is the formalization of textual proofs for other confluence criteria based on decreasing diagrams, e.g., decomposition results such as the direct sum [144], layer systems [47], or commutation [144]. The other direction is to certify proofs generated by confluence provers. While the Knuth-Bendix' criterion [100] and (weak) orthogonality [152] have already been integrated into the certifier CeTA by Thiemann [173], recently together with Julian Nagele we were able to lift the formalization of decreasing diagrams from abstract rewriting to term rewriting and extend CeTA to confluence proofs based on the rule labeling heuristic [129]. As a consequence in the certification track of the 2014 edition of the confluence competition such proofs have been checked by CeTA. One aim on the way to support certifiable proofs due to the incremental labeling framework is the integration of the main result presented in [71] into CeTA.

2.2. Complexity

Publications

4. Harald Zankl and Martin Korp, Modular Complexity Analysis for Term Rewriting, Logical Methods in Computer Science 10(1:19), 33 pages, 2014. Special issue of RTA 2010.
   doi: `10.2168/LMCS-10(1:19)2014`

5. Harald Zankl and Martin Korp, Modular Complexity Analysis via Relative Complexity, In Proceedings of the 21st International Conference on Rewriting Techniques and Applications (RTA 2010), Leibniz International Proceedings in Informatics 6, pp. 385–400, Schloss Dagstuhl, 2010.
   doi: `10.4230/LIPIcs.RTA.2010.385`

6. Friedrich Neurauter, Harald Zankl, and Aart Middeldorp, Revisiting Matrix Interpretations for Polynomial Derivational Complexity of Term Rewriting, In Proceedings of the 17th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-17), Lecture Notes in Computer Science (Advanced Research in Computing and Software Science) 6397, pp. 550–564, Springer, 2010.
   doi: `10.1007/978-3-642-16242-8_39`

History

For a terminating TRS the derivational complexity relates the size of a term with an upper bound on the length of rewrite sequences (derivations) starting from this term. Studying upper bounds on the (derivational) complexity induced by various termination criteria has a long tradition [77, 114, 190] to grasp the theoretical limitations of these methods. However, although most results establish tight bounds, i.e., those that can actually be reached by some rewrite systems, typically these bounds are too large and hence impractical for complexity analysis of programs, where polynomial upper limits are desired. Based on a suitable restriction of polynomial interpretations, the first method aiming in this direction was proposed in 2001 by Hofbauer [80] and implemented in 2008 by Schnabl [127, 156, 157]. Further polynomial complexity bounds have been established for match-bounds [55] (linear) and matrix interpretations of upper triangular [126] (polynomial in the dimension) and arctic [104] (linear) shape.

Contributions

Publication 4 proposes a general setting to combine different criteria for derivational complexity analysis with the help of relative rewriting. Since the framework is based on derivational complexity it is also applicable to more restricted notions such as (innermost) runtime complexity (see e.g. [75]). Martin Korp designed and implemented match-bounds for relative termination while I contributed the overall design of the modular framework and in particular the findings explained below. Based on relative rewriting, the framework facilitates partial complexity proofs where different base methods can be combined. This results in significant gains in power and we discuss examples where two base methods used stand-alone fail but in the combined framework they succeed to establish an upper bound. Furthermore, we present examples where two base methods used stand-alone cannot establish a tight upper bound on the complexity but when used in the combined framework they can. Due to the support for partial complexity proofs, for the first time complexity analysis is considered as an optimization problem in a non-trivial fashion. After *some* upper bound could be established this bound is tried to be tightened. To this end an existing proof can be inspected and alternative methods can be tested for the parts in the proof that contribute more to the inferred upper bound. Typically these alternative methods induce lower complexity bounds but are more costly to apply, so they are not employed in the first stage. Finally, we have generalized the weight gap principle [75] from weight functions to (suitably restricted) matrix interpretations and explain why similar ideas do not work for arctic interpretations or match-bounds.

Publication 6 deals with a significant improvement of the base method of matrix interpretations. Due to results from linear algebra (spectral radius theory) the eigenvalues of a matrix can be used to determine polynomial growth of the entries in products of this matrix. These findings subsume and extend the results based on upper triangular matrix interpretations [126]. While the usefulness of spectral radius theory has been identified and studied by Friedrich Neurauter, my contributions comprise the automation of the method. The problem is that the search for suitable matrices must be combined with the conditions that ensure polynomial bounds on the complexity. This involves the computation (or approximation) of the absolute values of eigenvalues and their frequency, which is significantly more challenging than ensuring the property of being an upper triangular matrix, i.e., a completely syntactical property. A further challenge is that the matrices are not known a priory, so all the computations must be done on parametric matrices (with unknown coefficients). While the theoretical basis holds for integer domains as well as for rational and real numbers, automation of matrix interpretations using the latter domains has been open for a long time. We tackled these issues by developing appropriate SMT solving techniques [208]. Finally, I contributed examples showing the benefit for the new approach, i.e., the new method strictly subsumes the results on upper triangular matrix interpretations and matrices over the rational numbers may induce tighter bounds than those over the naturals.

Impact

The proposed techniques have been implemented in the automatic complexity prover CaT, which dominated the category on derivational complexity in the termination competition [61] from 2008 until 2013. As a consequence of the power and applicability of the modular framework our approach has found its way into other rewriting based complexity tools (AProVE [134], $\mathsf{T_CT}$ [14]) very quickly.

Recently the connection between (joint) spectral radius theory [131] and the approach for inferring polynomial growth of matrix interpretations based on weighted automata [187] has been investigated (see Publication 17), revealing the equivalence of two results established in different communities via different formalisms.

2.3. Termination

Publications

7. Harald Zankl, Sarah Winkler, and Aart Middeldorp, Beyond Polynomials and Peano Arithmetic – Automation of Elementary and Ordinal Interpretations, Journal of Symbolic Computation, accepted for publication, to appear.
doi: `10.1016/j.jsc.2014.09.033`

8. Sarah Winkler, Harald Zankl, and Aart Middeldorp, Beyond Peano Arithmetic – Automatically Proving Termination of the Goodstein Sequence, In Proceedings of the 24th International Conference on Rewriting Techniques and Applications (RTA 2013), Leibniz International Proceedings in Informatics 21, pp. 335–351, Schloss Dagstuhl, 2013.
doi: `10.4230/LIPIcs.RTA.2013.335`

9. Nao Hirokawa, Aart Middeldorp, and Harald Zankl, Uncurrying for Termination and Complexity, Journal of Automated Reasoning 43(2), 279–315, 2013.
doi: `10.1007/s10817-012-9248-3`

10. Harald Zankl, Nao Hirokawa, and Aart Middeldorp, Uncurrying for Innermost Termination and Derivational Complexity, In Proceedings of the 5th International Conference on Higher-Order Rewriting (HOR 2010), Electronic Proceedings in Theoretical Computer Science 49, pp. 46–57, 2011.
doi: `10.4204/EPTCS.49.4`

11. Nao Hirokawa, Aart Middeldorp, and Harald Zankl, Uncurrying for Termination, In Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-15), Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence) 5330, pp. 667–681, Springer, 2008.
doi: `10.1007/978-3-540-89439-1_46`

History

Termination is a well-studied topic in rewriting and starting with the beginning of the new millennium also many criteria have been implemented successfully, as witnessed by the international competition of termination tools [61]. Due to the dependency pair framework [13, 60, 174] many different criteria can be combined and as a consequence modern termination tools can "almost decide"[1] termination. Still some challenges remain. Schnabl [155] conjectured that the derivational complexity of every rewrite system that can be shown terminating automatically by modern termination tools is limited by a multiple recursive function.

Another problem is to increase the performance of first-order termination tools to rewrite systems that contain some higher-order concepts.

Contributions

Publication 7 solves the problem of automation for two challenging termination criteria. Interpretations using ordinals (as e.g. used by Touzet [178]) enable termination proofs for systems whose derivational complexity cannot be bounded by multiple recursive functions. When using ordinals for automatic termination proofs some problems must be solved. Sarah Winkler proposed the shape of *restricted ordinal expressions* and suitable approximation functions to achieve automation of ordinal interpretations. However, addition and multiplication on ordinals are not strictly monotone, so the popular concept of well-founded monotone algebras cannot be used. Since these functions are weakly monotone valid termination proofs are obtained, provided the interpretation functions are *simple* as shown by Touzet [178] (and Zantema [209]). But the fact that the lexicographic combination

[1] I am grateful to Nao Hirokawa for coining this expression.

of two weakly monotone functions may destroy this property significantly hampers automation. Besides some refinements in the proposed approximation functions one of my contributions was the idea to use relative rewriting to ensure that the overall interpretation functions are weakly monotone and simple. As a consequence our implementation can provide automatic termination proofs for some systems that have been out of reach before, i.e., Touzet's encoding [178] of the battle of Hercules and Hydra due to Kirby and Paris [96], Winkler's encoding [193] of the Goodstein sequence [63], and (a corrected version of) Beklemishev's encoding [21] of the Worm principle, a one-dimensional version of the Hydra battle by Buchholz [28], first introduced by Hamano and Okada [67].

Based on our experience with approximation functions for ordinals we also tackled problem #28 in the the RTA List Of Open Problems[2] proposed by Lescanne in 1991.

> Develop effective methods to decide whether a system decreases with respect to some exponential interpretation.

Little progress has been achieved on this topic (see [115, 117] for preliminary results) despite the fact that the problem has been formulated more than 20 years ago. Furthermore, in a modern reading of this problem statement the *search* for suitable interpretation functions should also be performed automatically and not only the *check* that these interpretations prove a system terminating. Concerning both parts my contributions comprise the design and implementation of the approximation functions as well as the conception and evaluation of (necessary) heuristics for an efficient implementation. As a consequence the motivating examples (encoding the computation of the factorial function or Fibonacci numbers) can now be shown terminating by a direct termination proof using interpretations.

Publication 9 is concerned with applicability of termination tools for functional programs. It addresses the challenge for (first-order) termination provers to handle applicative systems, which model some higher-order concepts. Compared to its power on plain first-order problems, the dependency pair method [13, 60, 174] performs rather poorly on applicative systems. The inverse operation of currying (called uncurrying) allows to reconstruct the structure of applicative systems and recovers the power of the dependency pair method on the transformed systems. In contrast to earlier attempts [9], the version proposed by Nao Hirokawa—where the uncurrying rules are explicitly added to the rewrite system—does not rely on the presence of types. Also two dependency pair processors have been proposed by my co-authors. Besides the support for head variables in right-hand sides of rewrite

[2]`http://www.cs.tau.ac.il/~nachum/rtaloop/problems/28.html`

rules, uncurrying also preserves minimality when used in the dependency pair setting. This makes uncurrying much more powerful than the transformation $\mathcal{A}$ [59], which pursues the same idea. My contributions comprise the automation of the method besides the study how the transformation behaves for innermost rewriting and derivational complexity. Both extensions aim to improve the applicability of the method to real-world problems. The key insight in establishing that uncurrying preserves and reflects termination for innermost rewriting is that applicative terms may only be uncurried partially, such that the corresponding rewrite step of the uncurried system remains innermost. By the same argument, also soundness and completeness of dependency pair processors for uncurrying regarding the innermost rewriting strategy could be proved. Due to the additional rewrite steps needed to perform the uncurrying, rewrite sequences can only become longer in the transformed system. Hence uncurrying preserves upper bounds on the length of derivations. The useful result towards complexity analysis is that uncurrying also reflects polynomial complexities, i.e., the uncurrying steps are negligible compared to the steps performed due to the original system.

Impact

Despite the incredible progress that has been achieved regarding the automation of termination some (hard) problems have been open for many years. In Publication 7 we have solved two of them, the automation of ordinal interpretations and—by similar ideas—the search for exponential functions.

Another area where termination tools typically lack power are systems with higher-order ingredients. Nowadays dedicated methods and tools (THOR [25], Wanda [102]) for such problems are available but the existence of very powerful first-order termination provers justifies a transformation method to enhance support for problems that have some higher-order flavors. Since uncurrying preserves and reflects polynomial complexities it is also useful in the complexity analysis of rewrite systems, which is not yet supported by dedicated higher-order tools.

2.4. Automation

Publications

12. Harald Zankl, Bertram Felgenhauer, and Aart Middeldorp, CSI – A Confluence Tool, In Proceedings of the 23rd International Conference on Automated Deduction (CADE 2011), Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence) 6803, pp. 499–505, Springer, 2011.
    doi: `10.1007/978-3-642-22438-6_38`

13. Thomas Sternagel and Harald Zankl, KBCV – Knuth-Bendix Completion Visualizer, In Proceedings of the 6th International Joint Conference on Automated Reasoning (IJCAR 2012), Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence) 7364, pp. 530–536, Springer, 2012. doi: `10.1007/978-3-642-31365-3_41`

History

While automated termination analysis has advanced very well over the years, regarding the aspect of automation other important properties of rewriting have been neglected. Despite the vast number of different confluence criteria available in the literature, the first tool (ACP [11]) has been announced in 2009 only. In 2010 Hirokawa and Middeldorp proposed the seminal idea to employ the source labeling together with relative termination for decreasing diagrams [70, 71], which clearly gave another boost to automatic confluence analysis.

In the field of automatic completion, the tool Slothrop [189] made its appearance in 2006, with the use of external termination tools as a novel feature. Other powerful tools have been proposed in subsequent years. MKBTT [192] considers multiple options to orient an equation into a rewrite rule, while Maxcomp [98] tries to maximize the number of critical pairs that can be joined when orienting an equation. None of these tools allows interaction during the process of completion.

Contribution

Publication 12 presents the confluence tool CSI. While originally planned as a stand-alone tool developed from scratch, soon this design choice was abandoned and the tool was built on top of the termination tool $\mathsf{T_TT_2}$ [107] instead. Some reasons can be found in the setup of Publication 1, which uses various relative termination techniques for labeling diagrams. While the Knuth-Bendix' criterion [100] as well as the approach by Hirokawa and Middeldorp [70, 71] may rely on external (relative) termination tools, our setup of incremental labeling collaborates very closely with the relative termination criteria, e.g., the coefficients of a matrix interpretation are exploited for computing the labels of the diagrams. The benefit of this close coupling is a significant increase in power. CSI also supports a decomposition result based on ordered sorts, which has been proposed, studied, and implemented by Bertram Felgenhauer [47].

Publication 13 presents KBCV, an interactive tool for Knuth-Bendix completion [100]. Provided with an intuitive graphical user interface it displays the sets of equations and rules for the current state of completion. Besides the basic and efficient (following Huet [87]) completion procedures from [17], where the user only has to select the desired orientation of an equation into a rewrite rule, the tool also allows to perform completion via choosing the inference rules (on selections of equations or rules) according to the approach by Bachmair and Dershowitz [18]. In both settings the duty of maintaining a termination proof for the oriented rewrite rules is performed by the tool unless the user specifies the desired precedence for the lexicographic path order. At any stage KBCV allows the user to step backward (and forward again) in the completion process. Equipped with a fully automatic mode for completion as well it can compete with other state-of-the art completion tools such as Maxcomp [98], MKBTT [154], and Slothrop [189]. While Thomas Sternagel implemented KBCV my contributions comprised the overall design and several hints for performance related issues. Further contributions going beyond those listed in the system description are mentioned below.

Impact

With CSI as the third confluence tool being announced, a small but sufficiently large set of participants for a confluence competition [12] had been available. The first edition of this competition was conducted in 2012 and since then it has been held every year. The stimulating effects this competition has triggered on the community are detailed in the next section.

Actually KBCV is more than just a completion tool. Due to an appropriate recording of the applied inference rules [167], KBCV can also generate equational logic proofs and provide checkable completion certificates. The latter sparked the motivation of the CeTA team to tackle the certification of completion proofs [166, 167], paving the way for checkable confluence proofs [173]. This enabled the installation of a certified track in the confluence competition.

2.5. Further Contributions

I have co-authored the following further publications after receiving my PhD:

14. Harald Zankl, Decreasing Diagrams, Archive of Formal Proofs, 2013. Formal Proof Development
url: `afp.sourceforge.net/entries/Decreasing-Diagrams.shtml`

15. Sarah Winkler, Harald Zankl, and Aart Middeldorp, Ordinals and Knuth-Bendix Orders, In Proceedings of the 18th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-18), Lecture Notes in Computer Science (Advanced Research in Computing and Software Science) 7180, pp. 420–434, Springer, 2012.
doi: `10.1007/978-3-642-28717-6_33`

16. Bertram Felgenhauer, Harald Zankl, and Aart Middeldorp, Layer Systems for Proving Confluence, In Proceedings of the 31st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2011), Leibniz International Proceedings in Informatics 13, pp. 288–299, Schloss Dagstuhl, 2011.
    doi: `10.4230/LIPIcs.FSTTCS.2011.288`

17. Aart Middeldorp, Georg Moser, Friedrich Neurauter, Johannes Waldmann, and Harald Zankl, Joint Spectral Radius Theory for Automated Complexity Analysis of Rewrite Systems, In Proceedings of the 4th International Conference on Algebraic Informatics (CAI 2011), Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence) 6742, pp. 1–20, Springer, 2011.
    doi: `10.1007/978-3-642-21493-6_1`

18. Friedrich Neurauter, Aart Middeldorp, and Harald Zankl, Monotonicity Criteria for Polynomial Interpretations over the Naturals, In Proceedings of the 5th International Joint Conference on Automated Reasoning (IJCAR 2010), Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence) 6173, pp. 502–517, Springer, 2010.
    doi: `10.1007/978-3-642-14203-1_42`

19. Harald Zankl and Aart Middeldorp, Satisfiability of Non-Linear (Ir)rational Arithmetic, In Proceedings of the 16th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-16), Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence) 6355, pp. 481–500, Springer, 2010.
    doi: `10.1007/978-3-642-17511-4_27`

Next some highlights of these publications are described and some recent activities are mentioned.

The formalization of decreasing diagrams in the theorem prover Isabelle [197] has been submitted to the Archive of Formal Proofs (Publication 14), a large collection of formalizations in the theorem prover Isabelle. By making the formalization publicly available others can build on it.

The use of ordinals for the Knuth-Bendix order [118], in addition with the result that actually fairly small ordinals suffice for such an order [108], ignited my interest in ordinals. Together with Sarah Winkler and Aart Middeldorp in Publication 15 we show that as far as termination proving power is concerned, the Knuth-Bendix order does not benefit from ordinal domains at all. Furthermore, we propose a simplification in the definition of this order which does not affect its theoretical power (variables can be chosen to have weight one).

Besides my involvement in the tools CSI and KBCV, whose system descriptions are included in this thesis, I have also co-developed other tools for rewriting (the termination tool $\mathsf{T_TT_2}$ [107] and the complexity tool $\mathsf{C\!_a\!T}$ [205]) as well the SMT solver for non-linear integer/rational/real arithmetic MiniSmt (see Publication 19). These tools have participated in various competitions (confluence competition [12], SMT-COMP [34], termination competition [61]) of logic tools and have achieved top rankings in several categories.

Together with Takahito Aoto and Nao Hirokawa I have designed and organized the first three editions of the confluence competition (CoCo 2012, CoCo 2013, and CoCo 2014, see [12]). The last edition was part of the *FLoC Olympic Games 2014*,[3] a merger of 14 different competitions of logic tools. These games, held during the Vienna Summer of Logic aim to increase the visibility of the tools and foster progress. For the confluence competition, the progress is evident. In 2014 there have been three new entrants, CONCON and CO3 for the new track of conditional confluence, and the commutation tool CoLL, participating in the main track for confluence.[4] The other tracks for finding (non-)confluence proofs and certification thereof have been considered since 2012, with tools progressing every year. Here we list two distinguishing features of the confluence competition. The first is that it runs live during a workshop (as part of a presentation where the setup of the competition is explained), which makes the competition thrilling and comprehensible for people who did not participate by submitting a tool. The second is the organization of the confluence problems (Cops)[5] considered for the competition. Together with Nao Hirokawa we have developed an interface that allows the easy administration, submission, tagging, and query of such problems. Especially the (automatic) tagging mechanism (which indicates properties that the problems do satisfy or do not satisfy) has proved itself extremely useful in combination with the querying functionality that allows to search for problems satisfying Boolean combinations of these tags.

2.6. Related and Future Work

Nowadays logic based tools are becoming strong enough to be useful in practice. Due to the myriads of different approaches it is only possible to give a small piece of the big picture with the principal focus on rewriting based approaches. As a case study for termination and complexity analysis of programs we also compare to a selection of tools which do not have their foundations in rewriting.

[3]`http://vsl2014.at/olympics/`
[4]Further information on the tools is available from the competition website.
[5]`http://cl-informatik.uibk.ac.at/users/hzankl/cops`

The tool TERMINATOR supports termination analysis of low-level programs written in C (e.g. device drivers) and works by synthesizing linear ranking functions [150]. Recently its successor, T2[6] has been announced. Supporting user-defined data structures, the tool Speed [66] can compute symbolic complexity bounds for C/C++ programs. Regarding Java (Bytecode), the tool Costa [3] aims at automatic cost and termination analysis while Julia [161] allows termination and non-termination analysis, among others. The findings from [82] allow to infer upper bounds expressed by multivariate polynomials for a first-order fragment of OCaml (Resource Aware ML). Rewriting based termination tools are applicable to various programming languages via transformations (see e.g. [57, 146, 158, 169]). The international competition of termination tools [61] gives details about the performance on concrete testbeds involving some of the approaches mentioned above.

The use of rewriting based tools for termination (or resource analysis in general) has benefits and disadvantages at the same time. The transformation based approach(es) employing rewriting based tools benefit from the highly advanced termination criteria which have been developed for rewrite tools and hence often succeed where challenging termination arguments are required. On the contrary, the tools supporting a direct approach usually scale better on large programs where relatively simple arguments suffice to conclude termination. Another benefit of the rewriting based approach is that large parts of the theory in rewriting have already been formalized in interactive theorem provers (see e.g. the IsaFoR/CeTA project [176]).[7] As a consequence, the output of several tools can be confirmed to be correct by trustable checkers, making this approach very appealing for areas where soundness is critical.

Also automation of confluence techniques does have applications, albeit we are currently not aware that confluence tools are employed for the analysis of programs. We mention the use of such tools for showing soundness of abstract forms of reduction in solving the typing problem [170].

Interactive theorem provers like Isabelle simplify proof obligations according to a user-controlled set of rewrite rules (the simplifier). However, these rules are not subject of internal confluence/termination checks and the theorem prover leaves this important task to the user. To perform these tests automatically strong termination/confluence/completion tools are required. An approach to ensure confluence and termination of the rules in the simplifier is reported in [93]. It remains to be seen if first-order tools in combination with suitable transformations or dedicated higher-order tools are better suited for this purpose. While for termination some higher-order tools have already made their appearance (THOR [25], Wanda [102]), such tools are still lacking for confluence and completion.

[6]`http://research.microsoft.com/en-us/projects/t2/`

[7]`http://cl-informatik.uibk.ac.at/software/ceta/`

Part II

Confluence

3. Labelings for Decreasing Diagrams

Publication Details

Harald Zankl, Bertram Felgenhauer, and Aart Middeldorp, Labelings for Decreasing Diagrams, Journal of Automated Reasoning, accepted for publication, to appear.
doi: `10.1007/s10817-014-9316-y`

Harald Zankl, Bertram Felgenhauer, and Aart Middeldorp, Labelings for Decreasing Diagrams, In Proceedings of the 22nd International Conference on Rewriting Techniques and Applications (RTA 2011), Leibniz International Proceedings in Informatics 10, pp. 377–392, Schloss Dagstuhl, 2011.
doi: `10.4230/LIPIcs.RTA.2011.377`

Abstract

This article is concerned with automating the decreasing diagrams technique of van Oostrom for establishing confluence of term rewrite systems. We study abstract criteria that allow to lexicographically combine labelings to show local diagrams decreasing. This approach has two immediate benefits. First, it allows to use labelings for linear rewrite systems also for left-linear ones, provided some mild conditions are satisfied. Second, it admits an incremental method for proving confluence which subsumes recent developments in automating decreasing diagrams. The techniques proposed in the article have been implemented and experimental results demonstrate how, e.g., the rule labeling benefits from our contributions.

3.1. Introduction

Confluence is an important property of rewrite systems since it ensures unique normal forms. It is decidable in the presence of termination [100] and implied by orthogonality [152] or restricted joinability conditions on the critical pairs [84, 137, 143, 148, 179]. Recently, there is a renewed interest in confluence research, with a strong emphasis on automation. As one application we mention [170], where automated confluence tools are employed for proving soundness of abstract forms of reduction in solving the typing problem.

The decreasing diagrams technique of van Oostrom [139] is a complete method for showing confluence of countable abstract rewrite systems. The main idea of the approach is to show confluence by establishing local confluence under the side condition that rewrite steps of the joining sequences must *decrease* with respect to some well-founded order. For term rewrite systems however, the main problem for automation of decreasing diagrams is that in general infinitely many local peaks must be considered. To reduce this problem to a finite set of local peaks one can label rewrite steps with functions that satisfy special properties. In [138] van Oostrom presented the rule labeling that allows to conclude confluence of *linear* rewrite systems by checking decreasingness of the critical peaks (those emerging from critical overlaps). The rule labeling has been implemented by Aoto [4] and Hirokawa and Middeldorp [71]. Already in [138] van Oostrom presented constraints that allow to apply the rule labeling to *left-linear* systems. This approach has been implemented and extended by Aoto [4]. Our framework subsumes the above ideas.

The contributions of this article comprise the extraction of abstract constraints on a labeling such that for a (left-)linear rewrite system decreasingness of the (parallel) critical peaks ensures confluence. We show that the rule labeling adheres to our constraints and present additional labeling functions. Furthermore such labeling functions can be combined lexicographically to obtain new labeling functions satisfying our constraints. This approach allows the formulation of an abstract criterion that makes virtually every labeling function for linear rewrite systems also applicable to left-linear systems. Consequently, confluence of the TRS in Example 3.1 can be established automatically, e.g., by the rule labeling, while current approaches based on the decreasing diagrams technique [4, 71] as well as other confluence criteria like Knuth and Bendix' criterion or orthogonality (and its refinements) fail.

Example 3.1. Consider the TRS $\mathcal{R}$ (Cops #60)[1] consisting of the rules

$$\begin{array}{rrcl@{\qquad}rrcl}
1\colon & x+(y+z) & \to & (x+y)+z & 6\colon & x\times y & \to & y\times x \\
2\colon & (x+y)+z & \to & x+(y+z) & 7\colon & \mathsf{s}(x)+y & \to & x+\mathsf{s}(y) \\
3\colon & \mathsf{sq}(x) & \to & x\times x & 8\colon & x+\mathsf{s}(y) & \to & \mathsf{s}(x)+y \\
4\colon & \mathsf{sq}(\mathsf{s}(x)) & \to & (x\times x)+\mathsf{s}(x+x) & 9\colon & x\times\mathsf{s}(y) & \to & x+(x\times y) \\
5\colon & x+y & \to & y+x & 10\colon & \mathsf{s}(x)\times y & \to & (x\times y)+y
\end{array}$$

This system is locally confluent since all its 34 critical pairs are joinable.

[1]COnfluence ProblemS, see `http://coco.nue.riec.tohoku.ac.jp/problems/`.

The remainder of this article is organized as follows. After recalling preliminaries in Section 3.2 we present constraints (on a labeling) such that decreasingness of the critical peaks ensures confluence for (left-)linear rewrite systems in Section 3.3. Three of these constraints are based on relative termination while the fourth employs persistence. We focus on parallel rewriting in Section 3.4. The merits of these approaches are assessed in Section 3.5 by discussing the relationship to the recent literature. Implementation issues are addressed in Section 3.6 before Section 3.7 gives an empirical evaluation of our results. Section 3.8 concludes.

This article is an updated and extended version of [201], which presents the first incremental approach for labeling decreasing diagrams. Besides a number of small improvements, the article contains three new major contributions:

- Section 3.3.2, presenting a new labeling measuring the contracted redex,
- Section 3.3.2, which uses persistence to enhance the applicability of L-labelings for left-linear systems,
- Section 3.4, which studies parallel rewriting to make any weak LL-labeling applicable to showing confluence of left-linear systems without additional (relative termination) constraints.

The latter generalizes and incorporates recent findings from [46], which studies the rule labeling for parallel rewriting.

3.2. Preliminaries

We assume familiarity with term rewriting [17, 172].

Let $\mathcal{F}$ be a signature and let $\mathcal{V}$ be a set of variables disjoint from $\mathcal{F}$. By $\mathcal{T}(\mathcal{F},\mathcal{V})$ we denote the set of terms over $\mathcal{F}$ and $\mathcal{V}$. The expression $|t|_x$ indicates how often variable x occurs in term t. Positions are strings of natural numbers, i.e., elements of $\mathbb{N}_+^*$. The set of positions of a term t is defined as $\mathcal{P}\mathsf{os}(t) = \{\epsilon\}$ if t is a variable and as $\mathcal{P}\mathsf{os}(t) = \{\epsilon\} \cup \{iq \mid 1 \leqslant i \leqslant n \text{ and } q \in \mathcal{P}\mathsf{os}(t_i)\}$ if $t = f(t_1,\ldots,t_n)$. We write $p \leqslant q$ if $q = pp'$ for some position p', in which case $q \backslash p$ is defined to be p'. Furthermore $p < q$ if $p \leqslant q$ and $p \neq q$. Finally, $p \parallel q$ if neither $p \leqslant q$ nor $q < p$. Positions are used to address subterm occurrences. The subterm of t at position $p \in \mathcal{P}\mathsf{os}(t)$ is defined as $t|_p = t$ if $p = \epsilon$ and as $t|_p = t_i|_q$ if $p = iq$. We write $s \trianglelefteq t$ if s is a subterm of t and $s[t]_p$ for the result of replacing $s|_p$ with t in s. The set of function symbol positions $\mathcal{P}\mathsf{os}_{\mathcal{F}}(t)$ is $\{p \in \mathcal{P}\mathsf{os}(t) \mid t|_p \notin \mathcal{V}\}$ and $\mathcal{P}\mathsf{os}_{\mathcal{V}}(t) = \mathcal{P}\mathsf{os}(t) \setminus \mathcal{P}\mathsf{os}_{\mathcal{F}}(t)$. The set of variables occurring in a term t is denoted by $\mathcal{V}\mathsf{ar}(t)$. We let $t|_P = \{t|_p \mid p \in P\}$ if t is a term and P a set of positions.

A rewrite rule is a pair of terms (l, r), written $l \to r$, such that l is not a variable and all variables in r are contained in l. A rewrite rule $l \to r$ is duplicating if $|l|_x < |r|_x$ for some $x \in \mathcal{V}$. A term rewrite system (TRS) is a signature together with a finite set of rewrite rules over this signature. In the sequel signatures are implicit. By $\mathcal{R}_{\mathsf{d}}$ and $\mathcal{R}_{\mathsf{nd}}$ we denote the duplicating and non-duplicating rules of a TRS $\mathcal{R}$, respectively. A rewrite relation is a binary relation on terms that is closed under contexts and substitutions. For a TRS $\mathcal{R}$ we define $\to_{\mathcal{R}}$ to be the smallest rewrite relation that contains $\mathcal{R}$. As usual $\to^{=}$, $\to^{+}$, and $\to^{*}$ denotes the reflexive, transitive, and reflexive and transitive closure of $\to$, respectively.

A relative TRS $\mathcal{R}/\mathcal{S}$ is a pair of TRSs $\mathcal{R}$ and $\mathcal{S}$ with the induced rewrite relation $\to_{\mathcal{R}/\mathcal{S}} = \to_{\mathcal{S}}^{*} \cdot \to_{\mathcal{R}} \cdot \to_{\mathcal{S}}^{*}$. Sometimes we identify a TRS $\mathcal{R}$ with the relative TRS $\mathcal{R}/\varnothing$ and vice versa. A TRS $\mathcal{R}$ (relative TRS $\mathcal{R}/\mathcal{S}$) is terminating if $\to_{\mathcal{R}}$ ($\to_{\mathcal{R}/\mathcal{S}}$) is well-founded. Two relations $\geqslant$ and $>$ are called compatible if $\geqslant \cdot > \cdot \geqslant \subseteq >$. A monotone reduction pair $(\geqslant, >)$ consists of a preorder $\geqslant$ and a well-founded order $>$ such that $\geqslant$ and $>$ are compatible and closed under contexts and substitutions. A reduction pair $(\geqslant, >)$ is called simple if $f(s_1, \ldots, s_n) \geqslant s_i$ for all $1 \leqslant i \leqslant n$. We recall how to prove relative termination incrementally according to Geser [56].

Theorem 3.2. *A relative TRS $\mathcal{R}/\mathcal{S}$ is terminating if $\mathcal{R} = \varnothing$ or there exists a monotone reduction pair $(\geqslant, >)$ such that $\mathcal{R} \cup \mathcal{S} \subseteq \geqslant$ and $(\mathcal{R} \setminus >)/(\mathcal{S} \setminus >)$ is terminating.* □

A critical overlap $(l_1 \to r_1, p, l_2 \to r_2)_\mu$ of a TRS $\mathcal{R}$ consists of variants of rewrite rules $l_1 \to r_1$ and $l_2 \to r_2$ in $\mathcal{R}$ without common variables, a position $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(l_2)$, and a most general unifier μ of l_1 and $l_2|_p$. If $p = \epsilon$ then we require that $l_1 \to r_1$ and $l_2 \to r_2$ are not variants. From a critical overlap $(l_1 \to r_1, p, l_2 \to r_2)_\mu$ we obtain a critical peak $l_2\mu[r_1\mu]_p \leftarrow l_2\mu \to r_2\mu$ and a critical pair $l_2\mu[r_1\mu]_p \leftarrow\!\rtimes\!\to r_2\mu$.

If $l \to r \in \mathcal{R}$ and p is a position, we call the pair $\pi = \langle p, l \to r \rangle$ a redex pattern, and write l_π, r_π, p_π for its left-hand side, right-hand side, and position, respectively. We write $\to^{\pi}$ (or $\to^{p_\pi, l_\pi \to r_\pi}$) for a rewrite step at position p_π using the rule $l_\pi \to r_\pi$. A redex pattern π matches a term t if $t|_{p_\pi}$ is an instance of l_π. If π matches t, there is a unique reduct t^π with $t \to^\pi t^\pi$.

Let π_1 and π_2 be redex patterns that match a common term. They are called parallel ($\pi_1 \parallel \pi_2$) if $p_{\pi_1} \parallel p_{\pi_2}$. If $p_{\pi_2} \leqslant p_{\pi_1}$ and $p_{\pi_1} \backslash p_{\pi_2} \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(l_{\pi_2})$ then π_1 and π_2 overlap critically; otherwise they are called orthogonal ($\pi_1 \perp \pi_2$). Note that $\pi_1 \parallel \pi_2$ implies $\pi_1 \perp \pi_2$. We write $P \perp Q$ if $\pi \perp \pi'$ for all $\pi \in P$ and $\pi' \in Q$ and similarly $P \parallel Q$ if $\pi \parallel \pi'$ for all $\pi \in P$ and $\pi' \in Q$. If P is a set of pairwise parallel redex patterns matching a term t, we denote by $t \mathrel{+\!\!\!\!\to}^{P} t'$ the parallel rewrite step from t to t' by P, where $t' = t^{\pi_1 \cdots \pi_n}$ if $P = \{\pi_1, \ldots, \pi_n\}$. We allow P to be abbreviated to a set of positions in $t \mathrel{+\!\!\!\!\to}^{P} t'$.

We write $\langle A, \{\rightarrow_\alpha\}_{\alpha \in I}\rangle$ to denote the ARS $\langle A, \rightarrow\rangle$ where $\rightarrow$ is the union of $\rightarrow_\alpha$ for all $\alpha \in I$. Let $\langle A, \{\rightarrow_\alpha\}_{\alpha \in I}\rangle$ be an ARS and let $\geqslant$ and $>$ be relations on I. We write $\rightarrow_{\vee\alpha_1\cdots\alpha_n}$ for the union of $\rightarrow_\beta$ where $\beta < \alpha_i$ for some $1 \leqslant i \leqslant n$. We call $\rightarrow_\alpha$ and $\rightarrow_\beta$ *decreasing* (with respect to $\geqslant$ and $>$) if

$$\underset{\alpha}{\leftarrow} \cdot \underset{\beta}{\rightarrow} \subseteq \xrightarrow[\vee\alpha]{*} \cdot \xrightarrow[\vee\!\!/\beta]{=} \cdot \xrightarrow[\vee\alpha\beta]{*} \cdot \xleftarrow[\vee\alpha\beta]{*} \cdot \xleftarrow[\vee\!\!/\alpha]{=} \cdot \xleftarrow[\vee\beta]{*}$$

An ARS $\langle A, \{\rightarrow_\alpha\}_{\alpha \in I}\rangle$ is *decreasing* if there exists a preorder $\geqslant$ and a well-founded order $>$ such that $\geqslant$ and $>$ are compatible and $\rightarrow_\alpha$ and $\rightarrow_\beta$ are decreasing for all $\alpha, \beta \in I$ with respect to $\geqslant$ and $>$.

The following theorem is a reformulation of a result obtained by van Oostrom [139] (where $\geqslant$ is the identity relation). While allowing a preorder $\geqslant$ does not add power, it is more convenient for our purposes.

Theorem 3.3. *Every decreasing ARS is confluent.* $\square$

3.3. Labelings for Rewrite Steps

In this section we present constraints (on a labeling) such that decreasingness of the critical peaks ensures confluence of linear (Section 3.3.1) and left-linear (Section 3.3.2) TRSs. Furthermore, we show that if two labelings satisfy these conditions then also their lexicographic combination satisfies them.

For a local peak

$$t = s[r_1\sigma]_p \leftarrow s[l_1\sigma]_p = s = s[l_2\sigma]_q \rightarrow s[r_2\sigma]_q = u \tag{1}$$

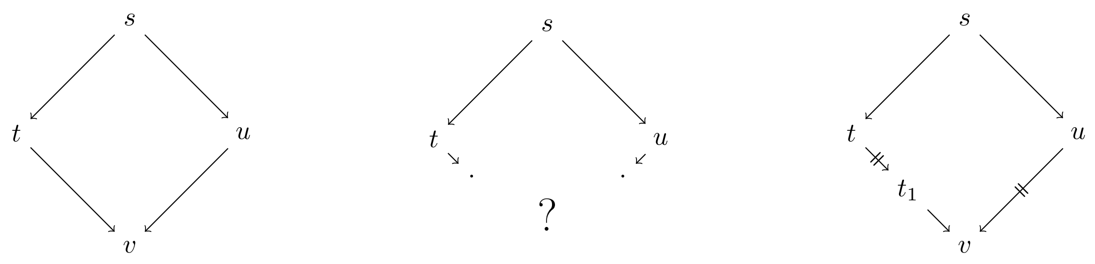
there are three possibilities (modulo symmetry):

(a) $p \parallel q$ (parallel),

(b) $q \leqslant p$ and $p \backslash q \in \mathcal{P}\mathsf{os}_\mathcal{F}(l_2)$ (critical overlap),

(c) $q < p$ and $p \backslash q \notin \mathcal{P}\mathsf{os}_\mathcal{F}(l_2)$ (variable overlap).

These cases are visualized in Figure 3.1. Figure 3.1(a) shows the shape of a local peak where the steps take place at parallel positions. Here we have $s \rightarrow^{p, l_1 \rightarrow r_1} t$ and $u \rightarrow^{p, l_1 \rightarrow r_1} v$ as well as $s \rightarrow^{q, l_2 \rightarrow r_2} u$ and $t \rightarrow^{q, l_2 \rightarrow r_2} v$, i.e., the steps drawn at opposing sides in the diagram are due to the same rules. The question mark in Figure 3.1(b) conveys that joinability of critical overlaps may depend on auxiliary rules. Variable overlaps (Figure 3.1(c)) can again be joined by the rules involved in the diverging step. More precisely, if q' is the unique position in $\mathcal{P}\mathsf{os}_\mathcal{V}(l_2)$ such that



(a) (parallel) (b) (critical overlap) (c) (variable overlap)

Figure 3.1.: Three kinds of local peaks.

$qq' \leqslant p$, $x = l_2|_{q'}$, $|l_2|_x = m$, and $|r_2|_x = n$ then we have $t \to_{l_1 \to r_1}^{m-1} t_1$, $t_1 \to_{l_2 \to r_2} v$, and $u \to_{l_1 \to r_1}^{n} v$.

Labelings are used to compare rewrite steps. In the sequel we denote the set of all rewrite steps for a TRS $\mathcal{R}$ by $\Lambda_{\mathcal{R}}$ and elements from this set by capital Greek letters Γ and Δ. Furthermore if $\Gamma = s \to^{p, l \to r} t$ then $C[\Gamma\sigma]$ denotes the rewrite step $C[s\sigma] \to^{p'p, l \to r} C[t\sigma]$ for any substitution σ and context C with $C|_{p'} = \Box$.

Definition 3.4. Let $\mathcal{R}$ be a TRS. A *labeling function* $\ell\colon \Lambda_{\mathcal{R}} \to W$ is a mapping from rewrite steps into some set W. A *labeling* $(\ell, \geqslant, >)$ for $\mathcal{R}$ consists of a labeling function ℓ, a preorder $\geqslant$, and a well-founded order $>$ such that $\geqslant$ and $>$ are compatible and for all rewrite steps $\Gamma, \Delta \in \Lambda_{\mathcal{R}}$, contexts C and substitutions σ:

1. $\ell(\Gamma) \geqslant \ell(\Delta)$ implies $\ell(C[\Gamma\sigma]) \geqslant \ell(C[\Delta\sigma])$, and
2. $\ell(\Gamma) > \ell(\Delta)$ implies $\ell(C[\Gamma\sigma]) > \ell(C[\Delta\sigma])$.

All labelings we present satisfy $> \subseteq \geqslant$, which allows to avoid tedious case distinctions, and we assume this property henceforth. We do so without loss of generality, because $((> \cup \geqslant)^*, >)$ satisfies the conditions of Definition 3.4 if $(\geqslant, >)$ does.

In the sequel W, $\geqslant$, and $>$ are left implicit when clear from the context and a labeling is identified with the labeling function ℓ. We use the terminology that a labeling ℓ is *monotone* and *stable* if properties 1 and 2 of Definition 3.4 hold. Abstract labels, i.e., labels that are unknown, are represented by lowercase Greek letters α, β, γ, and δ. We write $s \to_{\alpha}^{\pi} t$ (or simply $s \to_{\alpha} t$) if $\ell(s \to^{\pi} t) = \alpha$. Often we leave the labeling ℓ implicit and just attach labels to arrows. A local peak $t \leftarrow s \to u$ is called *decreasing for* ℓ if there are labels α and β such that $t \;{}_{\alpha}\!\leftarrow s \to_{\beta} u$, and $\to_{\alpha}$ and $\to_{\beta}$ are decreasing with respect to $\geqslant$ and $>$. To employ Theorem 3.3 for TRSs, decreasingness of the ARS $\langle \mathcal{T}(\mathcal{F}, \mathcal{V}), \{\to_w\}_{w \in W} \rangle$ must be shown.

$$\begin{tikzcd}
& s \arrow[dl, "\alpha"'] \arrow[dr, "\beta"] & \\
t \arrow[dr, "\delta"'] & & u \arrow[dl, "\gamma"] \\
& v &
\end{tikzcd}$$

(a) (parallel)

$$\begin{tikzcd}
& s \arrow[dl, "\alpha"'] \arrow[dr, "\beta"] & \\
t \arrow[dr, "\delta"'] & & u \arrow[dl, "="', "\gamma"] \\
& v &
\end{tikzcd}$$

(b) (variable-linear)

$$\begin{tikzcd}
& s \arrow[dl, "\alpha"'] \arrow[dr, "\beta"] & \\
t \arrow[dr, "\delta"'] & & u \arrow[dl, "\parallel" description, "\overline{\gamma}"] \\
& v &
\end{tikzcd}$$

(c) (variable-left-linear)

Figure 3.2.: Labeled local peaks.

In this article we investigate conditions on a labeling such that local peaks according to (parallel) and (variable overlap) are decreasing automatically. This is desirable since in general there are infinitely many local peaks corresponding to these cases (even if the underlying TRS has finitely many rules). There are also infinitely many local peaks according to (critical overlap) in general, but for a finite TRS they are captured by the finitely many critical overlaps. Still, it is undecidable if they are decreasingly joinable [71].

For later reference, Figure 3.2 shows labeled local peaks for the case (parallel) (Figure 3.2(a)) and (variable overlap) if the rule $l_2 \to r_2$ in local peak (1) is linear (Figure 3.2(b)) and left-linear (Figure 3.2(c)), respectively. In Figure 3.2(c) the expression $\overline{\gamma}$ denotes a sequence of labels $\gamma_1, \ldots, \gamma_n$. In the subsequent analysis we will always use the fact that the local peaks in Figure 3.2 can be closed by the rules involved in the peak (applied at opposing sides in the diagram).

3.3.1. Linear TRSs

The next definition presents sufficient abstract conditions on a labeling such that local peaks according to the cases (parallel) and (variable-linear) in Figure 3.2 are decreasing. We use the observation that for linear TRSs the (parallel) case can be seen as an instance of the (variable-linear) case to shorten proofs.

Definition 3.5. Let ℓ be a labeling for a TRS $\mathcal{R}$. We call ℓ an *L-labeling (for $\mathcal{R}$)* if for local peaks according to (parallel) and (variable-linear) we have $\alpha \geqslant \gamma$ and $\beta \geqslant \delta$ in Figures 3.2(a) and 3.2(b), respectively.

The local diagram in Figure 3.3(a) visualizes the conditions on an L-labeling more succinctly. We will use L-labelings also for left-linear TRSs, where no conditions are required for local peaks different from (parallel) and (variable-linear). We call the critical peaks of a TRS $\mathcal{R}$ Φ-*decreasing* if there exists a Φ-labeling ℓ

for $\mathcal{R}$ such that the critical peaks of $\mathcal{R}$ are decreasing for ℓ. In the sequel we will introduce further labelings, e.g., LL-labelings and weak LL-labelings. The placeholder Φ avoids the need for repeating the definition of decreasingness for these labelings.

The next theorem states that L-labelings may be used to show confluence of linear TRSs.

Theorem 3.6. *Let $\mathcal{R}$ be a linear TRS. If the critical peaks of $\mathcal{R}$ are L-decreasing then $\mathcal{R}$ is confluent.*

Proof. By assumption the critical peaks of $\mathcal{R}$ are decreasing for some L-labeling ℓ. We establish confluence of $\mathcal{R}$ by Theorem 3.3, i.e., show decreasingness of the ARS $\langle \mathcal{T}(\mathcal{F},\mathcal{V}), \to_{\mathcal{R}} \rangle$ where rewrite steps are labeled according to ℓ. Since $\mathcal{R}$ is linear, local peaks have the shape (parallel), (variable-linear), or (critical overlap). By definition of an L-labeling the former two are decreasing. Now consider a local peak according to (critical overlap), i.e., for the local peak (1) we have $q \leqslant p$ and $p \backslash q \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(l_2)$. Let $p' = p\backslash q$. Then $t|_q \leftarrow s|_q \to u|_q$ must be an instance of a critical peak $l_2\mu[r_1\mu]_{p'} \leftarrow l_2[l_1\mu]_{p'} = l_2\mu \to r_2\mu$ which is decreasing by assumption. By monotonicity and stability of ℓ we obtain decreasingness of the local peak (1). □

We recall the rule labeling of van Oostrom [138], parametrized by a mapping $i\colon \mathcal{R} \to \mathbb{N}$. Often i is left implicit. The rule labeling satisfies the constraints of an L-labeling.

Lemma 3.7. *Let $\mathcal{R}$ be a TRS and $\ell^i_{\mathrm{rl}}(s \to^{\pi} t) = i(l_\pi \to r_\pi)$. Then $(\ell^i_{\mathrm{rl}}, \geqslant_{\mathbb{N}}, >_{\mathbb{N}})$ is an L-labeling for $\mathcal{R}$.*

Proof. First we show that $(\ell^i_{\mathrm{rl}}, \geqslant_{\mathbb{N}}, >_{\mathbb{N}})$ is a labeling. The preorder $\geqslant_{\mathbb{N}}$ and the well-founded order $>_{\mathbb{N}}$ are compatible. Furthermore $\ell^i_{\mathrm{rl}}(s \to^{\pi} t) = i(l_\pi \to r_\pi)$ which ensures monotonicity and stability of ℓ^i_{rl}. Hence $(\ell^i_{\mathrm{rl}}, \geqslant_{\mathbb{N}}, >_{\mathbb{N}})$ is a labeling. Next we show the properties demanded in Definition 3.5. For local peaks according to cases (parallel) and (variable-linear) we recall that the steps drawn at opposite sides in the diagram, e.g., the steps labeled with α and γ (β and δ) in Figures 3.2(a) and 3.2(b), are due to applications of the same rule. Hence $\alpha = \gamma$ and $\beta = \delta$ in Figures 3.2(a) and 3.2(b), which shows the result. □

Inspired by [71] we propose a labeling based on relative termination.

Lemma 3.8. *Let $\mathcal{R}$ be a TRS and $\ell_{\mathrm{rt}}(s \to t) = s$. Then $\ell^{\mathcal{S}}_{\mathrm{rt}} = (\ell_{\mathrm{rt}}, \to^*_{\mathcal{R}}, \to^+_{\mathcal{S}/\mathcal{R}})$ is an L-labeling for $\mathcal{R}$, provided $\to_{\mathcal{S}} \subseteq \to_{\mathcal{R}}$ and $\mathcal{S}/\mathcal{R}$ is terminating.*

Proof. Let $\geqslant\ = \ \to_{\mathcal{R}}^{*}$ and $> \ = \ \to_{\mathcal{S}/\mathcal{R}}^{+}$. First we show that $(\ell_{\text{rt}}, \geqslant, >)$ is a labeling. By definition of relative rewriting, $\geqslant$ and $>$ are compatible and $>$ is well-founded by the termination assumption of $\mathcal{S}/\mathcal{R}$. Since rewriting is closed under contexts and substitutions, $\ell_{\text{rt}}^{\mathcal{S}}$ is monotone and stable and hence a labeling. Next we show the properties demanded in Definition 3.5. The assumption $\to_{\mathcal{S}} \subseteq \to_{\mathcal{R}}$ yields $> \ \subseteq \ \geqslant$. Combining $\alpha = s = \beta$, $\gamma = u$, and $\delta = t$ with $s \to_{\mathcal{R}} t$ and $s \to_{\mathcal{R}} u$ yields $\alpha = \beta \geqslant \gamma, \delta$ for local peaks according to (parallel) and (variable-linear) in Figures 3.2(a) and 3.2(b). □

The L-labeling from the previous lemma allows to establish a decrease with respect to some steps of $\mathcal{R}$. The next lemma allows to combine L-labelings. Let $\ell_1 \colon \Lambda_{\mathcal{R}} \to W_1$ and $\ell_2 \colon \Lambda_{\mathcal{R}} \to W_2$. Then $(\ell_1, \geqslant_1, >_1) \times (\ell_2, \geqslant_2, >_2)$ is defined as $(\ell_1 \times \ell_2, \geqslant_{12}, >_{12})$ where $\ell_1 \times \ell_2 \colon \Lambda_{\mathcal{R}} \to W_1 \times W_2$ with $(\ell_1 \times \ell_2)(\Gamma) = (\ell_1(\Gamma), \ell_2(\Gamma))$. Furthermore $(x_1, x_2) \geqslant_{12} (y_1, y_2)$ if and only if $x_1 >_1 y_1$ or $x_1 \geqslant_1 y_1$ and $x_2 \geqslant_2 y_2$ and $(x_1, x_2) >_{12} (y_1, y_2)$ if and only if $x_1 >_1 y_1$ or $x_1 \geqslant_1 y_1$ and $x_2 >_2 y_2$.

Lemma 3.9. *Let ℓ_1 and ℓ_2 be L-labelings. Then $\ell_1 \times \ell_2$ is an L-labeling.*

Proof. First we show that $\ell_1 \times \ell_2$ is monotone and stable whenever ℓ_1 and ℓ_2 are labelings. Indeed if $(\ell_1 \times \ell_2)(\Gamma) \geqslant (\ell_1 \times \ell_2)(\Delta)$ then $\ell_1(\Gamma) > \ell_1(\Delta)$ or $\ell_1(\Gamma) \geqslant \ell_1(\Delta)$ and $\ell_2(\Gamma) \geqslant \ell_2(\Delta)$, which for all contexts C and substitutions σ implies $\ell_1(C[\Gamma\sigma]) > \ell_1(C[\Delta\sigma])$ or $\ell_1(C[\Gamma\sigma]) \geqslant \ell_1(C[\Delta\sigma])$ and $\ell_2(C[\Gamma\sigma]) \geqslant \ell_2(C[\Delta\sigma])$ by stability and monotonicity of ℓ_1 and ℓ_2, which is equivalent to $(\ell_1 \times \ell_2)(C[\Gamma\sigma]) \geqslant (\ell_1 \times \ell_2)(C[\Delta\sigma])$. Showing stability and monotonicity of $>$ is similar. Since the lexicographic product satisfies $>_{12} \ \subseteq \ \geqslant_{12}$ if ℓ_1 and ℓ_2 are labelings we conclude that $\ell_1 \times \ell_2$ is a labeling.

Next we show that $\ell_1 \times \ell_2$ satisfies the requirements of Definition 3.5. If ℓ_1 and ℓ_2 are L-labelings then the diagram of Figure 3.2(b) has the shape as in Figure 3.3(a) and 3.3(b), respectively. It is easy to see that the lexicographic combination is again an L-labeling (cf. Figure 3.3(c)). □

3.3.2. Left-linear TRSs

For left-linear TRSs the notion of an LL-labeling is introduced. The following definition exploits that Figure 3.2(b) is an instance of Figure 3.2(c).

$$\begin{tikzcd}
 & s \arrow[dl, "\alpha_1"'] \arrow[dr, "\beta_1"] & \\
t \arrow[dr, "\leqslant\!/\beta_1"'] & & u \arrow[dl, "=", "\leqslant\!/\alpha_1"'] \\
 & v &
\end{tikzcd}$$

(a) Labeling ℓ_1.

$$\begin{tikzcd}
 & s \arrow[dl, "\alpha_2"'] \arrow[dr, "\beta_2"] & \\
t \arrow[dr, "\leqslant\!/\beta_2"'] & & u \arrow[dl, "=", "\leqslant\!/\alpha_2"'] \\
 & v &
\end{tikzcd}$$

(b) Labeling ℓ_2.

$$\begin{tikzcd}
 & s \arrow[dl, "{(\alpha_1,\alpha_2)}"'] \arrow[dr, "{(\beta_1,\beta_2)}"] & \\
t \arrow[dr, "{\leqslant\!/(\beta_1,\beta_2)}"'] & & u \arrow[dl, "=", "{\leqslant\!/(\alpha_1,\alpha_2)}"'] \\
 & v &
\end{tikzcd}$$

(c) Labeling $\ell_1 \times \ell_2$.

Figure 3.3.: Lexicographic combination of L-labelings.

Definition 3.10. A labeling ℓ for a TRS $\mathcal{R}$ is an *LL-labeling (for $\mathcal{R}$)* if

1. in Figure 3.2(a), $\alpha \geqslant \gamma$ and $\beta \geqslant \delta$,
2. in Figure 3.2(c), $\alpha \geqslant \overline{\gamma}$ and $\beta \geqslant \delta$ for all permutations of the rewrite steps of $u \twoheadrightarrow v$, where $\alpha \geqslant \overline{\gamma}$ means $\alpha \geqslant \gamma_i$ for $1 \leqslant i \leqslant n$, and
3. in Figure 3.2(c), $\alpha > \overline{\gamma}$ for some permutation of the rewrite steps of $u \twoheadrightarrow v$, where $\alpha > \overline{\gamma}$ means $\alpha \geqslant \gamma_1$ and $\alpha > \gamma_i$ for $2 \leqslant i \leqslant n$.

A labeling ℓ is a *weak LL-labeling* if the first two conditions are satisfied.

We strengthened the definition of (weak) LL-labelings from [201]. All labelings proposed in [201] satisfy the stronger conditions. Considering *all* permutations in condition 2 of Definition 3.10 is necessary to ensure that the lexicographic combination of two weak LL-labelings again is a weak LL-labeling (cf. Lemma 3.13). Furthermore, this condition facilitates their use for parallel rewriting (Section 3.4).

Remark 3.1. *The L-labelings presented so far (cf. Lemmata 3.7 and 3.8) are weak LL-labelings.*

The next theorem states that LL-labelings allow to show confluence of left-linear TRSs.

Theorem 3.11. *Let $\mathcal{R}$ be a left-linear TRS. If the critical peaks of $\mathcal{R}$ are LL-decreasing then $\mathcal{R}$ is confluent.*

Proof. By assumption the critical peaks of $\mathcal{R}$ are decreasing for some LL-labeling ℓ. We establish confluence of $\mathcal{R}$ by Theorem 3.3, i.e., we show decreasingness of the ARS $\langle \mathcal{T}(\mathcal{F}, \mathcal{V}), \rightarrow_{\mathcal{R}} \rangle$ by labeling rewrite steps according to ℓ. By definition of an LL-labeling local peaks according to (parallel) and (variable-left-linear) are decreasing. The reasoning for local peaks according to (critical overlap) is the same as in the proof of Theorem 3.6. $\square$

The rule labeling from Lemma 3.7 is a weak LL-labeling but not an LL-labeling since in Figure 3.2(c) we have $\alpha = \gamma_i$ for $1 \leqslant i \leqslant n$ which does not satisfy $\alpha > \overline{\gamma}$ if $n > 1$. (See also [71, Example 9].) We return to this problem and propose two solutions (in Sections 3.3.2 and 3.4) after presenting simpler (weak) LL-labelings based on measuring duplicating steps (Section 3.3.2), the context above the contracted redex (Section 3.3.2), and the contracted redex (Section 3.3.2).

Measuring duplicating steps

The L-labeling from Lemma 3.8 can be adapted to an LL-labeling.

Lemma 3.12. *Let $\mathcal{R}$ be a TRS. Then $\ell_{\mathrm{rt}}^{\mathcal{R}_\mathsf{d}}$ is an LL-labeling, provided $\mathcal{R}_\mathsf{d}/\mathcal{R}_\mathsf{nd}$ is terminating.*

Proof. By Theorem 3.2 the relative TRS $\mathcal{R}_\mathsf{d}/\mathcal{R}_\mathsf{nd}$ is terminating if and only if $\mathcal{R}_\mathsf{d}/\mathcal{R}$ is terminating. Hence $(\ell_{\mathrm{rt}}^{\mathcal{R}_\mathsf{d}}, \geqslant, >)$ is a labeling by Lemma 3.8. Here $\geqslant \, = \, \to_{\mathcal{R}}^{*}$ and $> \, = \, \to_{\mathcal{R}_\mathsf{d}/\mathcal{R}}^{+}$. Since $\ell_{\mathrm{rt}}(s \to t) = s$, we have $\alpha = \beta$ in Figures 3.2(a) and 3.2(c). We have $> \, \subseteq \, \geqslant$. Hence $\alpha \geqslant \gamma$ and $\alpha \geqslant \delta$ in Figure 3.2(a) and, if $l_2 \to r_2$ in the local peak (1) is linear, also in Figure 3.2(c) as $\overline{\gamma}$ is empty or $\overline{\gamma} = \gamma$ in this case. If $l_2 \to r_2$ is not linear then it must be duplicating and hence $\alpha > \gamma_i$ for $1 \leqslant i \leqslant n$. Because $\alpha \geqslant \delta$, $\ell_{\mathrm{rt}}^{\mathcal{R}_\mathsf{d}}$ is an LL-labeling for $\mathcal{R}$. □

To combine the previous lemma with the rule labeling we study how different labelings can be combined.

Lemma 3.13. *Let ℓ_1 be an LL-labeling and let ℓ_2 be a weak LL-labeling. Then $\ell_1 \times \ell_2$ and $\ell_2 \times \ell_1$ are LL-labelings.*

Proof. By the proof of Lemma 3.9 $\ell_1 \times \ell_2$ and $\ell_2 \times \ell_1$ are labelings. The only interesting case of (variable-left-linear) is when $l_2 \to r_2$ in local peak (1) is non-linear, i.e., $\overline{\gamma}$ contains more than one element. First we show that $\ell_1 \times \ell_2$ is an LL-labeling. Here labels according to ℓ_1 are suffixed with the subscript 1 and similarly for ℓ_2. Recall Figure 3.2(c). Let us first deal with Definition 3.10(2). We have $\alpha_1 \geqslant \overline{\gamma}_1$, $\beta_1 \geqslant \delta_1$, $\alpha_2 \geqslant \overline{\gamma}_2$ and $\beta_2 \geqslant \delta_2$, which yields $(\beta_1, \beta_2) \geqslant (\delta_1, \delta_2)$, $(\alpha_1, \alpha_2) \geqslant (\gamma_{1i}, \gamma_{2i})$ for all $1 \leqslant i \leqslant n$, by the definition of the lexicographic product. Next we consider Definition 3.10(3). By assumption we have $\alpha_1 > \overline{\gamma}_1$, and $\alpha_2 \geqslant \overline{\gamma}_2$, which yields the desired $(\alpha_1, \alpha_2) \geqslant (\gamma_{11}, \gamma_{21})$, $(\alpha_1, \alpha_2) > (\gamma_{1i}, \gamma_{2i})$ for $2 \leqslant i \leqslant n$. In the proof for $\ell_2 \times \ell_1$ the assumptions yield $(\beta_2, \beta_1) \geqslant (\delta_2, \delta_1)$ and $(\alpha_2, \alpha_1) \geqslant (\gamma_{2i}, \gamma_{1i})$ for $1 \leqslant i \leqslant n$ for Definition 3.10(2) and additionally $(\alpha_2, \alpha_1) > (\gamma_{2i}, \gamma_{1i})$ for $2 \leqslant i \leqslant n$ for Definition 3.10(3). □

Remark 3.2. *If ℓ_1 and ℓ_2 are weak LL-labelings then so are $\ell_1 \times \ell_2$ and $\ell_2 \times \ell_1$. Furthermore, LL-labelings are also weak LL-labelings by definition. In particular LL-labelings can be composed lexicographically.*

From Theorem 3.11 and Lemmata 3.12 and 3.13 we obtain the following result.

Corollary 3.14. *Let $\mathcal{R}$ be a left-linear TRS. If $\mathcal{R}_{\mathsf{d}}/\mathcal{R}_{\mathsf{nd}}$ is terminating and all critical peaks of $\mathcal{R}$ are weakly LL-decreasing then $\mathcal{R}$ is confluent.*

Proof. By Lemma 3.12 $\ell_{\mathrm{rt}}^{\mathcal{R}_{\mathsf{d}}}$ is an LL-labeling. By assumption the critical peaks of $\mathcal{R}$ are decreasing for some weak LL-labeling ℓ. By Lemma 3.13 also $\ell_{\mathrm{rt}}^{\mathcal{R}_{\mathsf{d}}} \times \ell$ is an LL-labeling. It remains to show decreasingness of the critical peaks for $\ell_{\mathrm{rt}}^{\mathcal{R}_{\mathsf{d}}} \times \ell$. This is obvious since for terms s, t, u with $s \to_{\mathcal{R}} t \to_{\mathcal{R}} u$ we have $\ell_{\mathrm{rt}}^{\mathcal{R}_{\mathsf{d}}}(s \to t) \geqslant \ell_{\mathrm{rt}}^{\mathcal{R}_{\mathsf{d}}}(t \to u)$. Hence decreasingness for ℓ implies decreasingness for $\ell_{\mathrm{rt}}^{\mathcal{R}_{\mathsf{d}}} \times \ell$. Confluence of $\mathcal{R}$ follows from Theorem 3.11. $\square$

We revisit the example from the introduction.

Example 3.15. For the TRS $\mathcal{R}$ from Example 3.1 the polynomial interpretation

$$+_{\mathbb{N}}(x,y) = x + y \quad \mathsf{s}_{\mathbb{N}}(x) = x + 1 \quad \times_{\mathbb{N}}(x,y) = x^2 + xy + y^2 \quad \mathsf{sq}_{\mathbb{N}}(x) = 3x^2 + 1$$

shows termination of $\mathcal{R}_{\mathsf{d}}/\mathcal{R}_{\mathsf{nd}}$. It is easy to check that ℓ_{rl}^{i} with $i(3) = i(6) = 2$, $i(4) = i(10) = 1$, and $i(l \to r) = 0$ for all other rules $l \to r \in \mathcal{R}$ establishes decreasingness of the 34 critical peaks. We consider two selected critical peaks (where the applied rewrite rule is indicated above the arrow in parentheses). The peaks

$$\begin{aligned}
t_1 = x + ((y + z) + w) &\xleftarrow[0]{(1)} x + (y + (z + w)) \xrightarrow[0]{(1)} (x + y) + (z + w) = u_1 \\
t_2 = \mathsf{s}(x) \times \mathsf{s}(x) &\xleftarrow[2]{(3)} \mathsf{sq}(\mathsf{s}(x)) \xrightarrow[1]{(4)} (x \times x) + \mathsf{s}(x + x) = u_2
\end{aligned}$$

can be joined decreasingly as follows:

$$\begin{aligned}
&t_1 \xrightarrow[0]{(2)} x + (y + (z + w)) \xleftarrow[0]{(2)} u_1 \\
&t_2 \xrightarrow[1]{(10)} (x \times \mathsf{s}(x)) + \mathsf{s}(x) \xrightarrow[0]{(9)} (x + (x \times x)) + \mathsf{s}(x) \xrightarrow[0]{(2)} x + ((x \times x) + \mathsf{s}(x)) \\
&\quad \xrightarrow[0]{(8)} x + (\mathsf{s}(x \times x) + x) \xleftarrow[0]{(2)} (x + \mathsf{s}(x \times x)) + x \xleftarrow[0]{(5)} (\mathsf{s}(x \times x) + x) + x \\
&\quad \xleftarrow[0]{(1)} \mathsf{s}(x \times x) + (x + x) \xleftarrow[0]{(8)} u_2
\end{aligned}$$

The next example is concise and constitutes a minimal example to familiarize the reader with Corollary 3.14.

Example 3.16. Consider the TRS $\mathcal{R}$ consisting of the three rules

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

We have $\mathcal{R}_{\mathsf{d}} = \{3\}$ and $\mathcal{R}_{\mathsf{nd}} = \{1,2\}$. Termination of $\mathcal{R}_{\mathsf{d}}/\mathcal{R}_{\mathsf{nd}}$ can be established by LPO with precedence $\mathsf{a} \sim \mathsf{b}$ and $\mathsf{f} > \mathsf{g}$. Taking rule numbers as labels the rule labeling shows the only critical peak $\mathsf{f}(\mathsf{g}(x,\mathsf{b}))\ {}_2{\leftarrow}\ \mathsf{f}(\mathsf{g}(x,\mathsf{a})) \to_3 \mathsf{g}(\mathsf{f}(x),\mathsf{f}(x))$ decreasing, as $\mathsf{f}(\mathsf{g}(x,\mathsf{b})) \to_1 \mathsf{f}(\mathsf{g}(x,\mathsf{a})) \to_3 \mathsf{g}(\mathsf{f}(x),\mathsf{f}(x))$. Hence we obtain the confluence of $\mathcal{R}$ by Corollary 3.14.

Remark 3.3. *Using $\ell^i_{\mathrm{rl}}(\cdot) = 0$ as weak LL-labeling, Corollary 3.14 gives a condition (termination of $\mathcal{R}_{\mathsf{d}}/\mathcal{R}_{\mathsf{nd}}$) such that $t \to^= u$ or $u \to^= t$ for all critical pairs $t \leftarrow\!\rtimes\!\to u$ implies confluence of a left-linear TRS $\mathcal{R}$. This partially answers one question in the RTA list of open problems #13.*[2]

Measuring the context above the contracted redex

In [138, Example 20] van Oostrom suggests to count function symbols above the contracted redex, demands that this measurement decreases for variables that are duplicated, and combines this with the rule labeling. Consequently local peaks according to Figure 3.2(c) are decreasing. Below we exploit this idea but incorporate the following beneficial generalizations. First, we do not restrict to counting function symbols (which has been adopted and extended by Aoto in [4]) but represent the constraints as a relative termination problem. This abstract formulation allows to strictly subsume the criteria from [4, 138] (see Section 3.5) because more advanced techniques than counting symbols can be applied for proving termination. Additionally, our setting also allows to weaken these constraints significantly (cf. Lemma 3.23).

The next example motivates the need for an LL-labeling that does not require termination of $\mathcal{R}_{\mathsf{d}}/\mathcal{R}_{\mathsf{nd}}$.

Example 3.17. Consider the TRS $\mathcal{R}$ consisting of the six rules

$$\begin{array}{rlrlrl} \mathsf{f}(\mathsf{h}(x)) &\to \mathsf{h}(\mathsf{g}(\mathsf{f}(x),x,\mathsf{f}(\mathsf{h}(\mathsf{a})))) & \mathsf{f}(x) &\to \mathsf{a} & \mathsf{a} &\to \mathsf{b} \\ \mathsf{h}(x) &\to \mathsf{c} & \mathsf{b} &\to \bot & \mathsf{c} &\to \bot \end{array}$$

As the duplicating rule admits an infinite sequence, Corollary 3.14 cannot succeed.

[2] `http://www.cs.tau.ac.il/~nachum/rtaloop/problems/13.html`

In the sequel we let $\mathcal{G}$ be the signature consisting of unary function symbols $f_1, \dots, f_n$ for every n-ary function symbol $f \in \mathcal{F}$.

Definition 3.18. Let $x \in \mathcal{V}$. We define a partial mapping $\star$ from terms in the original signature and positions $\mathcal{T}(\mathcal{F}, \mathcal{V}) \times \mathbb{N}_+^*$ to terms in $\mathcal{T}(\mathcal{G}, \mathcal{V})$ as follows:

$$\star(f(t_1, \dots, t_n), p) = \begin{cases} f_i(\star(t_i, q)) & \text{if } p = iq \\ x & \text{if } p = \epsilon \end{cases}$$

For a TRS $\mathcal{R}$ we abbreviate $\mathcal{R}^\star_>/\mathcal{R}^\star_=$ by $\star(\mathcal{R})$. Here, for $\gtrsim \in \{>, =\}$, $\mathcal{R}^\star_\gtrsim$ consists of all rules $\star(l, p) \to \star(r, q)$ such that $l \to r \in \mathcal{R}$, $l|_p = r|_q = y \in \mathcal{V}$, and $|r|_y \gtrsim 1$.

The next example illustrates the transformation $\star(\cdot)$.

Example 3.19. Consider the TRS $\mathcal{R}$ from Example 3.17. The relative TRS $\star(\mathcal{R}) = \mathcal{R}^\star_>/\mathcal{R}^\star_=$ consists of the TRS $\mathcal{R}^\star_>$ with rules

$$\mathsf{f_1}(\mathsf{h_1}(x)) \to \mathsf{h_1}(\mathsf{g_1}(\mathsf{f_1}(x))) \qquad \mathsf{f_1}(\mathsf{h_1}(x)) \to \mathsf{h_1}(\mathsf{g_2}(x))$$

and the TRS $\mathcal{R}^\star_=$ which is empty.

Due to the next lemma a termination proof of $\star(\mathcal{R})$ yields an LL-labeling.

Lemma 3.20. *Let $\mathcal{R}$ be a TRS and $\ell_\star(s \to^\pi t) = \star(s, p_\pi)$. Then $(\ell_\star, \geqslant, >)$ is an LL-labeling, provided $(\geqslant, >)$ is a monotone reduction pair, $\mathcal{R}^\star_> \subseteq >$, and $\mathcal{R}^\star_> \cup \mathcal{R}^\star_= \subseteq \geqslant$.*

Proof. Because $(\geqslant, >)$ is a monotone reduction pair, $(\ell_\star, \geqslant, >)$ is a labeling for $\mathcal{R}$. Note that monotonicity and stability are with respect to the signature $\mathcal{G}$. To see that the constraints of Definition 3.10 are satisfied we argue as follows. For Figure 3.2(a) we have $\alpha = \gamma$ and $\beta = \delta$ because the steps drawn at opposing sides in the diagram take place at the same positions and the function symbols above these positions stay the same. Next we consider Figure 3.2(b), i.e., the right-linear case. Recall the local peak (1). Again we have $\beta = \delta$ because $q < p$. To see $\alpha \geqslant \gamma$ consider the step $s \to^{q, l_2 \to r_2} u$ and let q' be the unique position in $\mathcal{P}\mathsf{os}_\mathcal{V}(l_2)$ such that $qq'r = p$ with $x = l_2|_{q'}$ for some position r. If $|r_2|_x = 0$ then there is no step and we are done. Otherwise let q'' be the position in r_2 with $|r_2|_{q''} = x$. By construction $\mathcal{R}^\star_=$ contains the rule $\star(l_2, q') \to \star(r_2, q'')$. Combining the assumption $\mathcal{R}^\star_= \subseteq \geqslant$ with monotonicity and stability of $\ell_\star$ yields $\star(s, p) \geqslant \star(u, qq''r)$, i.e., $\alpha \geqslant \gamma$. Next we consider Figure 3.2(c) for the duplicating case. Recall the local peak (1). Again we have $\beta = \delta$ because $q < p$. To see $\alpha > \overline{\gamma}$ (for any permutation of the steps) consider the step $s \to^{q, l_2 \to r_2} u$ and let q' be the unique position in $\mathcal{P}\mathsf{os}_\mathcal{V}(l_2)$

such that $qq'r = p$ for some position r. Let $x = l_2|_{q'}$ and $Q = \{q'_1, \ldots, q'_n\}$ with $r_2|_{q'_i} = x$. Then $P = \{qq'_ir \mid q'_i \in Q\}$ is the set of descendants of p. By construction $\mathcal{R}^\star_>$ contains all rules $\star(l_2, q') \to \star(r_2, q'_i)$ for $1 \leqslant i \leqslant n$. Combining the assumption $\mathcal{R}^\star_> \subseteq >$ with monotonicity and stability of $\ell_\star$ yields $\star(s,p) > \star(u, p'_i)$ for $p'_i \in P$. Since $u \twoheadrightarrow\!\!\!\!\!\!\!+\;^P v$ we obtain $\alpha > \gamma_i$ for $1 \leqslant i \leqslant n$ and hence the desired $\alpha > \overline{\gamma}$. □

Remark 3.4. *It is also possible to formulate Lemma 3.20 as a relative termination criterion without the use of a monotone reduction pair. However, the monotone reduction pair may admit more labels to be comparable (in the critical diagrams) because of the inclusions $\mathcal{R}^\star_> \subseteq >$ and $\mathcal{R}^\star_> \cup \mathcal{R}^\star_= \subseteq \geqslant$.*

From Lemma 3.20 we obtain the following corollary.

Corollary 3.21. *Let $\mathcal{R}$ be a left-linear TRS and let ℓ be a weak LL-labeling. Let $\ell_\star\ell$ denote $\ell \times \ell_\star$ or $\ell_\star \times \ell$. Let $(\geqslant, >)$ be a monotone reduction pair showing termination of $\star(\mathcal{R})$. If the critical peaks of $\mathcal{R}$ are decreasing for $\ell_\star\ell$ then $\mathcal{R}$ is confluent.*

Proof. The function $\ell_\star$ is an LL-labeling by Lemma 3.20. Lemma 3.13 yields that $\ell_\star\ell$ is an LL-labeling. By assumption the critical peaks are decreasing for $\ell_\star\ell$ and hence Theorem 3.11 yields the confluence of $\mathcal{R}$. □

The next example illustrates the use of Corollary 3.21.

Example 3.22. We show confluence of the TRS $\mathcal{R}$ from Example 3.17. Termination of $\star(\mathcal{R})$ (cf. Example 3.19) is easily shown, e.g., the polynomial interpretation

$$\mathsf{f_1}_{\mathbb{N}}(x) = 2x \qquad \mathsf{g_1}_{\mathbb{N}}(x) = \mathsf{g_2}_{\mathbb{N}}(x) = x \qquad \mathsf{h_1}_{\mathbb{N}}(x) = x + 1$$

orients both rules in $\mathcal{R}^\star_>$ strictly. The labeling $\ell_\star$ in combination with the rule labeling where $i(\mathsf{f}(\mathsf{h}(x)) \to \mathsf{h}(\mathsf{g}(\mathsf{f}(x), x, \mathsf{f}(\mathsf{h}(\mathsf{a}))))) = 1$ and all other rules receive label 0 shows decreasingness of the three critical peaks (two of which are symmetric). For the moment we label a step $s \to^\pi t$ with the interpretation of $\star(s, p_\pi)$. E.g., a step $\mathsf{f}(\mathsf{h}(\mathsf{b})) \to \mathsf{f}(\mathsf{h}(\bot))$ is labeled $2x + 2$ since $\star(\mathsf{f}(\mathsf{h}(\mathsf{b})), 11) = \mathsf{f_1}(\mathsf{h_1}(x))$ and $[\mathsf{f_1}(\mathsf{h_1}(x))]_{\mathbb{N}} = 2x + 2$. The critical peak $\mathsf{h}(\mathsf{g}(\mathsf{f}(x), x, \mathsf{f}(\mathsf{h}(\mathsf{a})))) \;{}_{x,1}\!\!\leftarrow \mathsf{f}(\mathsf{h}(x)) \to_{x,0} \mathsf{a}$ is closed decreasingly by

$$\mathsf{h}(\mathsf{g}(\mathsf{f}(x), x, \mathsf{f}(\mathsf{h}(\mathsf{a})))) \underset{x,0}{\longrightarrow} \mathsf{c} \underset{x,0}{\longrightarrow} \bot \underset{x,0}{\longleftarrow} \mathsf{b} \underset{x,0}{\longleftarrow} \mathsf{a}$$

and the critical peak $\mathsf{h}(\mathsf{g}(\mathsf{f}(x), x, \mathsf{f}(\mathsf{h}(\mathsf{a})))) \;{}_{x,1}\!\!\leftarrow \mathsf{f}(\mathsf{h}(x)) \to_{2x,0} \mathsf{f}(\mathsf{c})$ is closed decreasingly by

$$\mathsf{h}(\mathsf{g}(\mathsf{f}(x), x, \mathsf{f}(\mathsf{h}(\mathsf{a})))) \underset{x,0}{\longrightarrow} \mathsf{c} \underset{x,0}{\longrightarrow} \bot \underset{x,0}{\longleftarrow} \mathsf{b} \underset{x,0}{\longleftarrow} \mathsf{a} \underset{x,0}{\longleftarrow} \mathsf{f}(\mathsf{c})$$

which allows to prove confluence of $\mathcal{R}$ by Corollary 3.21.

By definition of $\alpha > \overline{\gamma}$ (cf. Definition 3.10) we observe that the definition of $\star(\mathcal{R})$ can be relaxed. If $l_2 \to r_2$ with $l_2|_{q'} = x \in \mathcal{V}$ and $\{q'_1, \ldots, q'_n\}$ are the positions of the variable x in r_2 then it suffices if $n - 1$ instances of $\star(l_2, q') \to \star(r_2, q'_i)$ are put in $\mathcal{R}^{\star}_{>}$ while one $\star(l_2, q') \to \star(r_2, q'_j)$ can be put in $\mathcal{R}^{\star}_{=}$ (since the steps labeled $\overline{\gamma}$ in Figure 3.2(c) are at parallel positions we can choose the first closing step such that $\alpha \geqslant \gamma_1$). This improved version of $\star(\mathcal{R})$ is denoted by $\overset{\star}{\star}(\mathcal{R}) = \mathcal{R}^{\star\star}_{>}/\mathcal{R}^{\star\star}_{=}$. We obtain the following variant of Lemma 3.20.

Lemma 3.23. *Let $\mathcal{R}$ be a TRS. Then $(\ell_\star, \geqslant, >)$ is an LL-labeling, provided $(\geqslant, >)$ is a monotone reduction pair, $\mathcal{R}^{\star\star}_{>} \subseteq >$, and $\mathcal{R}^{\star\star}_{>} \cup \mathcal{R}^{\star\star}_{=} \subseteq \geqslant$.* ☐

Obviously any $\overset{\star}{\star}(\mathcal{R})$ inherits termination from $\star(\mathcal{R})$. The next example shows that the reverse statement does not hold. In Section 3.6 we show how the intrinsic indeterminism of $\overset{\star}{\star}(\mathcal{R})$ is eliminated in the implementation.

Example 3.24. Consider the TRS $\mathcal{R}$ from Example 3.1. The TRS $\mathcal{R}^{\star}_{>}$ consists of the rules

$$\begin{array}{lll}
\mathsf{sq}_1(x) \to \times_1(x) & \mathsf{sq}_1(\mathsf{s}_1(x)) \to +_1(\times_1(x)) & \times_1(x) \to +_1(x) \\
\mathsf{sq}_1(x) \to \times_2(x) & \mathsf{sq}_1(\mathsf{s}_1(x)) \to +_1(\times_2(x)) & \dagger\colon \times_1(x) \to +_2(\times_1(x)) \\
 & \mathsf{sq}_1(\mathsf{s}_1(x)) \to +_2(\mathsf{s}_1(+_1(x))) & \dagger\colon \times_2(y) \to +_1(\times_2(y)) \\
 & \mathsf{sq}_1(\mathsf{s}_1(x)) \to +_2(\mathsf{s}_1(+_2(x))) & \times_2(y) \to +_2(y)
\end{array}$$

while $\mathcal{R}^{\star}_{=}$ consists of the rules

$$\begin{array}{rrr}
+_1(x) \to +_1(+_1(x)) & +_1(x) \to +_2(x) & +_1(x) \to +_1(\mathsf{s}_1(x)) \\
+_2(+_1(y)) \to +_1(+_2(y)) & +_2(y) \to +_1(y) & +_2(\mathsf{s}_1(y)) \to +_2(y) \\
+_2(+_2(z)) \to +_2(z) & \times_1(x) \to \times_2(x) & \times_2(\mathsf{s}_1(y)) \to +_2(\times_2(y)) \\
+_1(+_1(x)) \to +_1(x) & \times_2(y) \to \times_1(y) & \times_1(\mathsf{s}_1(x)) \to +_1(\times_1(x)) \\
+_1(+_2(y)) \to +_2(+_1(y)) & +_1(\mathsf{s}_1(x)) \to +_1(x) & \\
+_2(z) \to +_2(+_2(z)) & +_2(y) \to +_2(\mathsf{s}_1(y)) &
\end{array}$$

Let $\mathcal{R}^{\star}_{\dagger}$ denote the rules in $\mathcal{R}^{\star}_{>}$ marked with $\dagger$. Termination of $\star(\mathcal{R})$ cannot be established (because $\mathcal{R}^{\star}_{\dagger}$ is non-terminating) but we stress that moving these rules into $\mathcal{R}^{\star}_{=}$ yields a valid $\overset{\star}{\star}(\mathcal{R})$ which can be proved terminating by the polynomial interpretation with

$$\mathsf{sq}_{1\mathbb{N}}(x) = x + 2 \qquad\qquad \times_{1\mathbb{N}}(x) = \times_{2\mathbb{N}}(x) = x + 1$$

that interprets the remaining function symbols by the identity function. We remark that Corollary 3.21 with the labeling from Lemma 3.23 establishes confluence of $\mathcal{R}$.

Since all reductions in the 34 joining sequences have only $+$ above the redex and $+_{1\mathbb{N}}(x) = +_{2\mathbb{N}}(x) = x$, the $\ell_\star$ labeling attaches x to any of these steps. The rule labeling that assigns $i(3) = i(6) = 2$, $i(4) = i(10) = 1$, and 0 to all other rules shows the 34 critical peaks decreasing.

Measuring the contracted redex

Instead of the labeling $\ell_\star$, which is based on the context above the contracted redex, one can also use the contracted redex itself for labeling.

Lemma 3.25. *Let $\mathcal{R}$ be a TRS and $\ell_\triangle(s \to^\pi t) = s|_{p_\pi}$. Then $(\ell_\triangle, \geqslant, >)$ is a weak LL-labeling, provided $(\geqslant, >)$ is a monotone reduction pair with $\mathcal{R} \subseteq \geqslant$.*

Proof. Because $(\geqslant, >)$ is a monotone reduction pair, $(\ell_\triangle, \geqslant, >)$ is a labeling for $\mathcal{R}$. To see that the constraints of Definition 3.10 are satisfied we argue as follows. For Figure 3.2(a) we have $\alpha = \gamma$ and $\beta = \delta$. For Figure 3.2(c) we get $\alpha = \gamma_1 = \cdots = \gamma_n$ (since the same redex is contracted) and $\beta \geqslant \delta$ by the assumption $\mathcal{R} \subseteq \geqslant$ and monotonicity and stability of $\geqslant$. $\square$

The following definition collects the constraints, such that variable overlaps can be made decreasing.

Definition 3.26. For a TRS $\mathcal{R}$ let $\mathcal{R}^\triangle = \{l \to x \mid l \to r \in \mathcal{R} \text{ and } |r|_x > 1\}$.

Due to the next result a termination proof of $\mathcal{R}^\triangle/\mathcal{R}$ enables a weak LL-labeling to establish confluence.

Corollary 3.27. *Let $\mathcal{R}$ be a left-linear TRS and let ℓ be a weak LL-labeling. Let $(\geqslant, >)$ be a simple monotone reduction pair showing termination of $\mathcal{R}^\triangle/\mathcal{R}$. If the critical peaks of $\mathcal{R}$ are decreasing for $\ell_\triangle \times \ell$ then $\mathcal{R}$ is confluent.*

Proof. Note that $\ell_\triangle \times \ell$ is a weak LL-labeling (cf. Remark 3.2), which shows the peaks in Figure 3.2(a) and Figure 3.2(b) decreasing. For the duplicating case of Figure 3.2(c) we inspect the labels with regard to $\ell_\triangle$. Consider the local peak (1). Clearly, $\beta = l_2\sigma$ and $\alpha = l_1\sigma$. Since $\gamma_i = \alpha$, we want to establish $\beta > \alpha$. To this end let $q' \in \mathcal{P}\mathsf{os}_\mathcal{V}(l_2)$ such that $qq'r = p$ and $x = l_2|_{q'}$. Note that $l_2 \to x \in \mathcal{R}^\triangle$ because we are in the duplicating case. Hence the relative termination assumption gives $l_2 > x$, and $l_2\sigma > x\sigma$ is obtained by stability. Now as $x\sigma|_r = l_1\sigma$ the desired $\beta > \alpha$ follows from simplicity of the reduction pair since $l_2\sigma > x\sigma \geqslant l_1\sigma$. Combining $\ell_\triangle$ lexicographically with a weak LL-labeling ℓ into $\ell_\triangle \times \ell$ maintains decreasingness. $\square$

Remark 3.5. *Note that the labeling $\ell_\triangle \times \ell$ from Corollary 3.27 is not an LL-labeling. The point is that there are multiple ways of ensuring decreasingness of Figure 3.2(c). For LL-labelings, we use $\alpha > \overline{\gamma}$, while in Corollary 3.27, $\beta > \gamma_i$ for $1 \leqslant i \leqslant n$ does the job. This is also the reason why $\ell \times \ell_\triangle$ cannot be used in Corollary 3.27. Consider the TRS with the rules $1 : \mathsf{f}(x) \to \mathsf{g}(x, x)$ and $2 : \mathsf{a} \to \mathsf{b}$. Let ℓ_{rl} be the rule labeling attaching the rule numbers as labels. Then the variable overlap is not decreasing for $\ell_{\mathrm{rl}} \times \ell_\triangle$.*

We demonstrate Corollary 3.27 on the TRS from Example 3.16.

Example 3.28. For the TRS from Example 3.16 the polynomial interpretation

$$\mathsf{g}_{\mathbb{N}}(x, y) = 2x + 2y + 1 \qquad \mathsf{a}_{\mathbb{N}} = \mathsf{b}_{\mathbb{N}} = 0 \qquad \mathsf{f}_{\mathbb{N}}(x) = x^2$$

establishes relative termination of $\{\mathsf{f}(\mathsf{g}(x, \mathsf{a})) \to x\}/\mathcal{R}$ and shows the critical peak decreasing when labeling steps with the pair obtained by the interpretation of the redex and the rule labeling, i.e., $\mathsf{f}(\mathsf{g}(x, \mathsf{b}))\ {}_{0,2}{\leftarrow}\ \mathsf{f}(\mathsf{g}(x, \mathsf{a})) \to_{(2x+1)^2,3} \mathsf{g}(\mathsf{f}(x), \mathsf{f}(x))$ for the peak and $\mathsf{f}(\mathsf{g}(x, \mathsf{b})) \to_{0,1} \mathsf{f}(\mathsf{g}(x, \mathsf{a})) \to_{(2x+1)^2,3} \mathsf{g}(\mathsf{f}(x), \mathsf{f}(x))$ for the join.

Exploiting Persistence

In this section we show how to exploit persistence of confluence [5, 47] to enhance the applicability of L-labelings to certain duplicating left-linear TRSs. Compared to Sections 3.3.2–3.3.2, where variable overlaps were closed decreasingly by a relative termination criterion, here persistence arguments are employed to avoid reasoning about variable overlaps at duplicating variable positions at all. To this end we recall order-sorted TRSs.

Definition 3.29. Let S be a set of sorts equipped with a partial order $\leq$. A signature $\mathcal{F}$ and a set of variables $\mathcal{V}$ are S-sorted if every n-ary function symbol $f \in \mathcal{F}$ is equipped with a sort declaration $\alpha_1 \times \cdots \times \alpha_n \to \alpha$ where $\alpha_1, \ldots, \alpha_n, \alpha \in S$ and every variable $x \in \mathcal{V}$ has exactly one sort $\alpha \in S$. We write $S(f) = \alpha$, $S(f, i) = \alpha_i$ for $1 \leqslant i \leqslant n$, and $S(x) = \alpha$, respectively. We let $\mathcal{V}_\alpha = \{x \in \mathcal{V} \mid S(x) = \alpha\}$ and require that $\mathcal{V}_\alpha$ is infinite for all $\alpha \in S$. The set of S-sorted terms, $\mathcal{T}_S(\mathcal{F}, \mathcal{V})$, is the union of the sets $\mathcal{T}_\alpha(\mathcal{F}, \mathcal{V})$ for $\alpha \in S$ that are inductively defined as follows: $\mathcal{V}_\alpha \subseteq \mathcal{T}_\alpha(\mathcal{F}, \mathcal{V})$ and $f(t_1, \ldots, t_n) \in \mathcal{T}_\alpha(\mathcal{F}, \mathcal{V})$ whenever $f \in \mathcal{F}$ has sort declaration $\alpha_1 \times \cdots \times \alpha_n \to \alpha$ and $t_i \in \mathcal{T}_{\leq \alpha_i}(\mathcal{F}, \mathcal{V})$ for all $1 \leqslant i \leqslant n$. Here $\mathcal{T}_{\leq \alpha}(\mathcal{F}, \mathcal{V})$ is the union of all $\mathcal{T}_\beta(\mathcal{F}, \mathcal{V})$ for $\beta \leq \alpha$.

The notion of S-sorted terms properly extends many-sorted terms. Indeed, if we let $\leq$ be the identity relation then $\mathcal{T}_{\leq \alpha}(\mathcal{F}, \mathcal{V}) = \mathcal{T}_\alpha(\mathcal{F}, \mathcal{V})$, which means that the i-th argument of f in an S-sorted term must have sort $S(f, i)$.

Definition 3.30. We extend $S(\cdot)$ and $S(\cdot,\cdot)$ to S-sorted terms t and non-root positions of t. If $t = f(t_1,\dots,t_n)$ then $S(t) = S(f)$, $S(t,i) = S(f,i)$, and $S(t,ip) = S(t_i,p)$ for $p \neq \epsilon$. If $t = x \in \mathcal{V}$ then $S(t) = S(x)$.

Example 3.31. Let $S = \{0,1,2\}$ with $0 \leq 1$ and consider the sort declarations $\mathsf{f} : 1 \to 2$ and $x : 0$. Then $t = \mathsf{f}(x) \in \mathcal{T}_S(\{\mathsf{f}\},\{x\})$, $S(t) = 2$, $S(t,1) = 1$, and $S(t|_1) = 0 \leq 1$.

One easily observes that $S(t,p)$ defines the maximal sort induced by the context $t[\square]_p$: a term $t[u]_p$ is S-sorted if and only if $u \in \mathcal{T}_{\leq S(t,p)}(\mathcal{F},\mathcal{V})$. Consequently, we have $S(t|_p) \leq S(t,p)$ for all non-root positions p of t.

We are particularly interested in the case where rewriting restricted to S-sorted terms coincides with ordinary rewriting with initial terms restricted to S-sorted ones. This property is captured by S-compatible TRSs.

Definition 3.32. A TRS $\mathcal{R}$ is *S-compatible* if for every rule $l \to r \in \mathcal{R}$ there exists a sort $\alpha \in S$ such that $l \in \mathcal{T}_\alpha(\mathcal{F},\mathcal{V})$ and $r \in \mathcal{T}_{\leq\alpha}(\mathcal{F},\mathcal{V})$, and $S(l,p) = S(l|_p)$ for all $p \in \mathcal{P}\mathsf{os}_\mathcal{V}(l)$.

The following lemma is well-known (e.g. [188]) and easy to prove.

Lemma 3.33. *If $\mathcal{R}$ is S-compatible then $\mathcal{T}_S(\mathcal{F},\mathcal{V})$ and $\mathcal{T}_{\leq\alpha}(\mathcal{F},\mathcal{V})$ for every $\alpha \in S$ are closed under rewriting by $\mathcal{R}$.* $\square$

The following result is a special case of [47, Theorem 6.2].

Theorem 3.34. *An S-compatible left-linear TRS $\mathcal{R}$ is confluent on $\mathcal{T}(\mathcal{F},\mathcal{V})$ if and only if it is confluent on $\mathcal{T}_S(\mathcal{F},\mathcal{V})$.* $\square$

Example 3.35. Consider the duplicating TRS $\mathcal{R}$ with rules

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

Recall that L-labelings (in particular, rule labelings) that are not LL-labelings are not applicable to non-linear TRSs because the variable overlap diagram (Figure 3.2(c)) is not decreasing. Let $S = \{0,1\}$ with the following sort declarations:

$$x : 0 \qquad \mathsf{a} : 0 \qquad \mathsf{b} : 0 \qquad \mathsf{f} : 0 \to 1 \qquad \mathsf{g} : 1 \times 1 \to 1$$

The TRS $\mathcal{R}$ is S-compatible and hence we may restrict rewriting to S-sorted terms without affecting confluence by Theorem 3.34. This has the beneficial effect that variable overlaps are ruled out. To see how, note that no subterms of sort 1 can appear inside terms of sort 0. Consider the left-hand side $\mathsf{f}(x)$ of $\mathcal{R}$. We have $S(\mathsf{f}(x),1) = 0$, so that any term substituted for x must have sort 0. Further note that both left-hand sides have sort 1. Consequently, no rule application may be nested below $\mathsf{f}(x) \to \mathsf{g}(\mathsf{f}(x), \mathsf{f}(x))$ and hence variable overlaps are ruled out. Therefore, we may use L-labelings to show confluence of $\mathcal{R}$ even though $\mathcal{R}$ is not linear, and in fact the rule labeling which takes the rule numbers as labels allows us to join the sole (modulo symmetry) critical peak $t = \mathsf{f}(\mathsf{b}) \;{}_1\!\leftarrow \mathsf{f}(\mathsf{a}) \to_2 \mathsf{g}(\mathsf{f}(\mathsf{a}), \mathsf{f}(\mathsf{a})) = u$ decreasingly: $t \to_2 \mathsf{g}(\mathsf{f}(\mathsf{b}), \mathsf{f}(\mathsf{b})) \;{}_1\!\leftarrow \mathsf{g}(\mathsf{f}(\mathsf{b}), \mathsf{f}(\mathsf{a})) \;{}_1\!\leftarrow u$.

Formally, we define $\mathcal{T}_{\trianglelefteq\alpha}(\mathcal{F}, \mathcal{V}) = \{t \mid t \trianglelefteq t' \text{ for some } t' \in \mathcal{T}_{\leq\alpha}(\mathcal{F}, \mathcal{V})\}$, to capture which terms may occur as subterms of terms of sort α or below.

Theorem 3.36. *Let $\mathcal{R}$ be a left-linear S-compatible TRS such that the variable $l|_p$ occurs at most once in r whenever $l \to r \in \mathcal{R}$ and $l' \to r' \in \mathcal{R}$ with $l' \in \mathcal{T}_{\trianglelefteq S(l,p)}(\mathcal{F}, \mathcal{V})$ for some $p \in \mathcal{P}\mathsf{os}_\mathcal{V}(l)$. Then $\mathcal{R}$ is confluent if all its critical peaks are L-decreasing.*

Proof. By Theorem 3.34 we may restrict rewriting to S-sorted terms. The proof follows that of Theorem 3.6, except in the analysis of local peaks, where right-linearity of $\mathcal{R}$ is used, which is not among our assumptions. Instead, we argue as follows: Since $\mathcal{R}$ is left-linear, any local peak has the shape (parallel), (critical overlap), or (variable-left-linear). In the latter case, the step $s \to^{q, l' \to r'} t$ is nested below $s \to^{p, l \to r} u$, and it is easy to see that this implies $l' \in \mathcal{T}_{\trianglelefteq S(l,q')}(\mathcal{F}, \mathcal{V})$ for some variable position q' of l such that $pq' \leqslant q$. Consequently the variable $x = l|_{q'}$ occurs at most once in r by assumption, and the parallel step (which contains one rewrite step for every occurrence of x in r) is empty or a single step, resulting in a decreasing diagram. □

As a refinement of Theorem 3.36, instead of ruling out duplicating (variable-left-linear) overlaps completely, we can also add additional constraints on the labeling for the remaining variable overlaps.

Definition 3.37. Let ℓ be a weak LL-labeling for an S-compatible TRS $\mathcal{R}$. We call ℓ *persistent* if whenever rules $l \to r, l' \to r' \in \mathcal{R}$ satisfy $l' \in \mathcal{T}_{\trianglelefteq S(l,p)}(\mathcal{F}, \mathcal{V})$ for some $p \in \mathcal{P}\mathsf{os}_\mathcal{V}(l)$, either $|r|_{l|_p} \leqslant 1$ or $\beta > \overline{\gamma}$ in Figure 3.2(c) for all resulting variable overlaps with $l' \to r'$ below $l \to r$. We call $\mathcal{R}$ persistent LL-decreasing if there is a persistent, weak LL-labeling ℓ such that all critical peaks of $\mathcal{R}$ are decreasing with respect to ℓ.

Theorem 3.38. *Let $\mathcal{R}$ be a left-linear TRS. If the critical peaks of $\mathcal{R}$ are persistent LL-decreasing then $\mathcal{R}$ is confluent.*

Proof. The proof follows along the lines of the proof of Theorem 3.36. In the case of a duplicating variable-left-linear overlap, the additional constraints ensure that the resulting diagram is decreasing. □

Example 3.39. Suppose we extend the TRS from Example 3.35 with the rule $\mathsf{a} \to \mathsf{b}$, using the same sorts:

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

Theorem 3.36 is no longer applicable, because rule 3 may be nested below rule 1, which is duplicating. However, by the preceding remark, any rule labeling with $\ell^i_{\mathrm{rl}}(1) > \ell^i_{\mathrm{rl}}(3)$ will make the corresponding variable overlaps decreasing.

Remark 3.6. *Note that Theorem 3.38 does not subsume Theorem 3.36, because the former demands a weak LL-labeling whereas the latter requires only an L-labeling. If we were to restrict the L-labeling and weak LL-labeling conditions to those variable overlaps that are consistent with the sort declarations, then Theorem 3.38 would subsume Theorem 3.36. We chose not to do so because all our labelings are weak LL-labelings.*

The following example shows that considering order-sorted instead of many-sorted signatures is beneficial.

Example 3.40. Consider the duplicating TRS $\mathcal{R}$ given by the rules

$$1\colon\ \mathsf{h}(\mathsf{a}, \mathsf{a}) \to \mathsf{f}(\mathsf{a}) \qquad 2\colon\ \mathsf{f}(\mathsf{a}) \to \mathsf{a} \qquad 3\colon\ \mathsf{f}(x) \to \mathsf{h}(x, x)$$

Furthermore, let $\mathcal{S} = \{0, 1\}$ with $1 > 0$ and take the sort declarations

$$\mathsf{h} : 0 \times 0 \to 1 \qquad \mathsf{f} : 0 \to 1 \qquad \mathsf{a} : 0$$

Considering only $\mathcal{S}$-sorted terms, no rule can be nested below the duplicating rule $\mathsf{f}(x) \to \mathsf{h}(x, x)$. Basically, there is one critical peak, $\mathsf{h}(\mathsf{a}, \mathsf{a}) \;{}^3\!\!\leftarrow \mathsf{f}(\mathsf{a}) \to^2 \mathsf{a}$, which is decreasingly joinable as $\mathsf{h}(\mathsf{a}, \mathsf{a}) \to^1 \mathsf{f}(\mathsf{a}) \to^2 \mathsf{a}$ by the rule labeling (using rule numbers as labels), and confluence follows by Theorem 3.36. Due to the rule $\mathsf{f}(\mathsf{a}) \to \mathsf{a}$, any many-sorted sort declaration for $\mathcal{R}$ must assign the same sorts to a and the argument and result types of f. Therefore, $\mathsf{f}(x) \to \mathsf{h}(x, x)$ may be nested below itself, and Theorems 3.36 and 3.38 would fail in connection with the rule labeling.

3.4. Labelings for Parallel Rewriting

In this section, rather than labeling individual rewrite steps, we will label parallel rewrite steps instead. This is inspired by the parallel moves lemma, which says that any peak $t \mathrel{+\!\!\!\!\leftarrow} s \mathrel{+\!\!\!\!\rightarrow} u$ of two non-overlapping parallel rewrite steps can be joined in a diamond as $t \twoheadrightarrow\!\!\!\!\!\!+\ \cdot \twoheadleftarrow\!\!\!\!\!\!+\ u$, and diamonds are comparatively easy to label decreasingly, as we saw in Section 3.3.1.

The main problem is to label parallel steps such that variable overlaps are decreasing. Since $\{\alpha\} \not\geqslant_{\mathsf{mul}} \{\alpha, \ldots, \alpha\}$, the multiset of the single steps' labels does not work. Hence we use sets to label parallel steps which we denote by capital Greek letters. Sets of labels are ordered by the Hoare preorder of $(\geqslant, >)$, which we denote by $(\geqslant_H, >_H)$ and is defined by

$$\begin{aligned} \Gamma >_H \Delta &\quad\iff\quad \Gamma \neq \varnothing \wedge \forall \beta \in \Delta\, \exists\, \alpha \in \Gamma\, (\alpha > \beta) \\ \Gamma \geqslant_H \Delta &\quad\iff\quad \forall \beta \in \Delta\, \exists\, \alpha \in \Gamma\, (\alpha \geqslant \beta) \end{aligned}$$

For readability we drop the subscript H when attaching labels to rewrite steps as in $\twoheadrightarrow\!\!\!\!\!\!+_{\ \vee\Gamma}$.

Example 3.41. Let $\geqslant$ denote the natural order on $\mathbb{N}$. Then $\{1\} \geqslant_H \{0, 1\}$ and $\{1\} \geqslant_H \{1, 1, 1\} = \{1\}$ but $\{5, 4\} \not\geqslant_H \{5, 3\}$.

The following lemma states obvious properties of Hoare preorders which we implicitly use in the sequel.

Lemma 3.42. *Let $(\geqslant_H, >_H)$ be a Hoare preorder.*

1. *If $(\geqslant, >)$ is a monotone reduction pair then $(\geqslant_H, >_H)$ is a monotone reduction pair.*
2. *If $\Gamma \supseteq \Gamma'$ then $\Gamma \geqslant_H \Gamma'$.*
3. *If $\Gamma >_H \Gamma'$ and $\Delta >_H \Delta'$ then $\Gamma \cup \Delta >_H \Gamma' \cup \Delta'$.*
4. *If $\Gamma \geqslant_H \Gamma'$ and $\Delta \geqslant_H \Delta'$ then $\Gamma \cup \Delta \geqslant_H \Gamma' \cup \Delta'$.* □

As we have seen in Section 3.3.2, constructing LL-labelings is quite a bit harder than constructing L-labelings, because of the duplicated steps in the (variable-left-linear) case (Figure 3.2(c)). Here, we use weak LL-labelings for labeling single and parallel rewrite steps. Throughout this section we assume a given left-linear TRS $\mathcal{R}$, and a weak LL-labeling ℓ with corresponding labeling function for parallel steps $\ell^{\|}$, as introduced in the following definition.

Definition 3.43. We lift a weak LL-labeling ℓ to parallel steps $t \twoheadrightarrow\!\!\!\!\!\!+^{\ P}\ t'$ as follows. For each $\pi \in P$, we have a rewrite step $t \to^{\pi} t^{\pi}$. We label a parallel step $t \twoheadrightarrow\!\!\!\!\!\!+^{\ P}\ t'$ by $\ell^{\|}(t \twoheadrightarrow\!\!\!\!\!\!+^{\ P}\ t') = \{\ell(t \to^{\pi} t^{\pi}) \mid \pi \in P\}$.

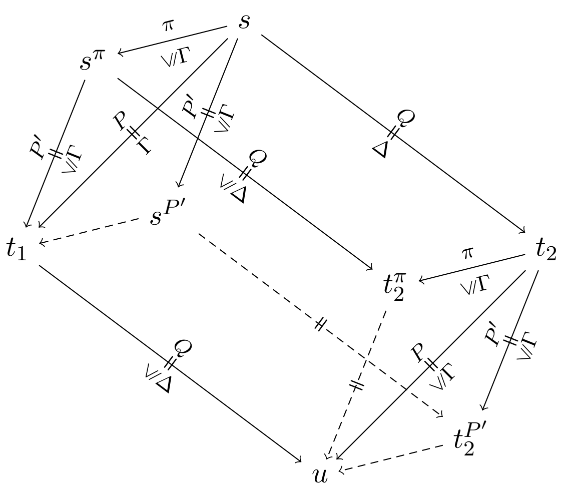


Figure 3.4.: Weak LL-labeling applied to parallel steps.

A parallel rewrite step is labeled by the set of the labels of the single steps making up the parallel step, written $t \mathrel{+\!\!\!\twoheadrightarrow}_\Gamma^P t'$.

The next example shows that the labels change when decomposing a parallel step into a sequence of single steps, i.e., the label of the parallel step may be different from the union of labels of the single steps. However, the proof of Lemma 3.45 reveals that for weak LL-labelings the labels never increase when sequencing a parallel step.

Example 3.44. Consider the rule $\mathsf{a} \to \mathsf{b}$ and the extension of the source labeling $\ell(s \to t) = s$ to parallel steps. Then $\mathsf{f(a,a)} \mathrel{+\!\!\!\twoheadrightarrow}_{\{\mathsf{f(a,a)}\}} \mathsf{f(b,b)}$ but $\mathsf{f(a,a)} \mathrel{+\!\!\!\twoheadrightarrow}_{\{\mathsf{f(a,a)}\}} \mathsf{f(b,a)} \mathrel{+\!\!\!\twoheadrightarrow}_{\{\mathsf{f(b,a)}\}} \mathsf{f(b,b)}$. Clearly $\{\mathsf{f(a,a)}\} \neq \{\mathsf{f(a,a)}, \mathsf{f(b,a)}\}$. This effect is intrinsic to labelings that take the context of the rewrite step into account. On the other hand, the rule labeling gives $\mathsf{f(a,a)} \mathrel{+\!\!\!\twoheadrightarrow}_{\{1\}} \mathsf{f(b,b)}$ and $\mathsf{f(a,a)} \mathrel{+\!\!\!\twoheadrightarrow}_{\{1\}} \mathsf{f(b,a)} \mathrel{+\!\!\!\twoheadrightarrow}_{\{1\}} \mathsf{f(b,b)}$ with $\{1\} = \{1,1\}$, because the labels are independent of the context.

The next lemma is the key to show that even for parallel rewriting overlaps due to Figure 3.2(a) (parallel) and Figure 3.2(c) (variable-left-linear) are decreasing.

Lemma 3.45.

1. *Let $t_1 \;{}_\Gamma^P\!\!\twoheadleftarrow\!\!\!+\; s \mathrel{+\!\!\!\twoheadrightarrow}_\Delta^Q t_2$ with $P \parallel Q$. Then there is a term u such that $s \mathrel{+\!\!\!\twoheadrightarrow}_{\Gamma\cup\Delta}^{P\cup Q} u$ and $t_1 \mathrel{+\!\!\!\twoheadrightarrow}_{\Delta'}^Q u \;{}_{\Gamma'}^P\!\!\twoheadleftarrow\!\!\!+\; t_2$, where $\Gamma \geqslant_H \Gamma'$ and $\Delta \geqslant_H \Delta'$.*

2. *Let $s \mathrel{+\!\!\!\twoheadrightarrow} s'$ and $\sigma(x) \mathrel{+\!\!\!\twoheadrightarrow} \sigma'(x)$ for all $x \in \mathcal{V}$, so that there are parallel rewrite steps $s\sigma' \;{}_\Gamma^P\!\!\twoheadleftarrow\!\!\!+\; s\sigma \mathrel{+\!\!\!\twoheadrightarrow}_\Delta^Q s'\sigma$. Then $s\sigma' \mathrel{+\!\!\!\twoheadrightarrow}_{\Delta'}^Q s'\sigma' \;{}_{\Gamma'}\!\!\twoheadleftarrow\!\!\!+\; s'\sigma$ and $\Gamma \geqslant_H \Gamma'$, $\Delta \geqslant_H \Delta'$. Furthermore, if $\sigma(x) = \sigma'(x)$ for all $x \in \mathcal{V}\mathsf{ar}(s'|_Q)$ then $s\sigma \mathrel{+\!\!\!\twoheadrightarrow}_\Sigma s'\sigma'$ for some $\Sigma \subseteq \Gamma \cup \Delta$.*

Proof. 1. First note that since $P \parallel Q$, a term u with $s \mathrel{+\!\!\!\twoheadrightarrow}^{P\cup Q} u$ exists. We have

$$\begin{aligned}\ell^{\parallel}(s \xrightarrow{P\cup Q}\!\!\!\!\!\!\!\!\!+\;\; u) &= \{\ell(s \xrightarrow{\pi} s^{\pi}) \mid \pi \in P \cup Q\}\\ &= \{\ell(s \xrightarrow{\pi} s^{\pi}) \mid \pi \in P\} \cup \{\ell(s \xrightarrow{\pi} s^{\pi}) \mid \pi \in Q\}\\ &= \ell^{\parallel}(s \overset{P}{\mathrel{+\!\!\!\twoheadrightarrow}} t_1) \cup \ell^{\parallel}(s \overset{Q}{\mathrel{+\!\!\!\twoheadrightarrow}} t_2) = \Gamma \cup \Delta\end{aligned}$$

by definition. To establish $t_1 \mathrel{+\!\!\!\twoheadrightarrow}^{Q}_{\Delta'} u \;{}^{P}_{\Gamma'}\!\mathrel{\twoheadleftarrow\!\!\!+}\, t_2$, we use induction on $|P| + |Q|$. We consider several base cases. If $|P| = 0$ or $|Q| = 0$ then the result follows by definition of parallel rewriting. If $|P| = |Q| = 1$ the result follows from the fact that ℓ is a weak LL-labeling, Definition 3.10(1) (Figure 3.2(a)). For the induction step, assume without loss of generality that $|P| > 1$ and let $P = \{\pi\} \uplus P'$. The proof is illustrated in Figure 3.4. The parallel P-step can be decomposed into a π-step and a P'-step. Since $\{\pi\}, P' \subseteq P$, the labels are less than or equal to Γ. Then we apply the induction hypothesis to the peaks

i. $s^{P'} \;{}^{P'}_{\vee\!\!/\Gamma}\!\mathrel{\twoheadleftarrow\!\!\!+}\, s \mathrel{+\!\!\!\twoheadrightarrow}^{\{\pi\}}_{\vee\!\!/\Gamma} s^{\pi}$ yielding $s^{\pi} \mathrel{+\!\!\!\twoheadrightarrow}^{P'}_{\vee\!\!/\Gamma} t_1$,

ii. $s^{\pi} \;{}^{\{\pi\}}_{\vee\!\!/\Gamma}\!\mathrel{\twoheadleftarrow\!\!\!+}\, s \mathrel{+\!\!\!\twoheadrightarrow}^{Q}_{\Delta} t_2$ yielding $t_2 \mathrel{+\!\!\!\twoheadrightarrow}^{\{\pi\}}_{\vee\!\!/\Gamma} t_2^{\pi}$ and $s^{\pi} \mathrel{+\!\!\!\twoheadrightarrow}^{Q}_{\vee\!\!/\Delta} t_2^{\pi}$,

iii. $s^{P'} \;{}^{P'}_{\vee\!\!/\Gamma}\!\mathrel{\twoheadleftarrow\!\!\!+}\, s \mathrel{+\!\!\!\twoheadrightarrow}^{Q}_{\Delta} t_2$ yielding $t_2 \mathrel{+\!\!\!\twoheadrightarrow}^{P'}_{\vee\!\!/\Gamma} t_2^{P'}$, which we merge with $t_2 \mathrel{+\!\!\!\twoheadrightarrow}^{\{\pi\}}_{\vee\!\!/\Gamma} t_2^{\pi}$ to obtain $t_2 \mathrel{+\!\!\!\twoheadrightarrow}^{P}_{\vee\!\!/\Gamma} u$, noting that the union of two sets from $\vee\!\!/\Gamma$ is again in $\vee\!\!/\Gamma$, and finally

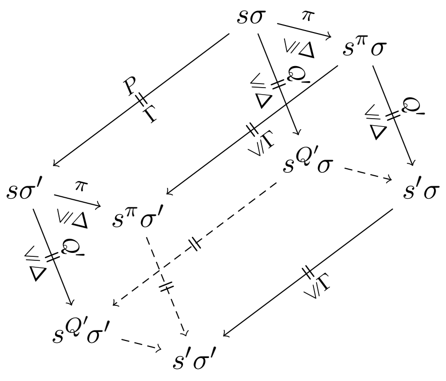
iv. $t_1 \;{}^{P'}_{\vee\!\!/\Gamma}\!\mathrel{\twoheadleftarrow\!\!\!+}\, s^{\pi} \mathrel{+\!\!\!\twoheadrightarrow}^{Q}_{\vee\!\!/\Delta} t_2^{\pi}$ yielding $t_1 \mathrel{+\!\!\!\twoheadrightarrow}^{Q}_{\vee\!\!/\Delta} u$.

2. The existence of parallel rewrite steps $s\sigma' \mathrel{+\!\!\!\twoheadrightarrow} s'\sigma'$ and $s'\sigma \mathrel{+\!\!\!\twoheadrightarrow} s'\sigma'$ follows easily from the definition of parallel steps. We establish $\Gamma \geqslant_H \Gamma'$ and $\Delta \geqslant_H \Delta'$ by induction on $|Q|$. The reasoning for the induction step ($|Q| > 1$) is very similar to the induction step in item 1, cf. Figure 3.5(a): Taking $Q = \{\pi\} \uplus Q'$, we split $s\sigma \mathrel{+\!\!\!\twoheadrightarrow}^{Q}_{\Delta} s'\sigma$ into $s\sigma \mathrel{+\!\!\!\twoheadrightarrow}^{\{\pi\}}_{\vee\!\!/\Delta} s^{\pi}\sigma$ and $s\sigma \mathrel{+\!\!\!\twoheadrightarrow}^{Q'}_{\vee\!\!/\Delta} s^{Q'}\sigma$. We apply the induction hypothesis to the peaks

i. $s\sigma' \;{}^{P}_{\Gamma}\!\mathrel{\twoheadleftarrow\!\!\!+}\, s\sigma \mathrel{+\!\!\!\twoheadrightarrow}^{\{\pi\}}_{\vee\!\!/\Delta} s^{\pi}\sigma$ yielding $s\sigma' \mathrel{+\!\!\!\twoheadrightarrow}^{\{\pi\}}_{\vee\!\!/\Delta} s^{\pi}\sigma'$ and $s^{\pi}\sigma \mathrel{+\!\!\!\twoheadrightarrow}_{\vee\!\!/\Gamma} s^{\pi}\sigma'$,

ii. $s\sigma' \;{}^{P}_{\Gamma}\!\mathrel{\twoheadleftarrow\!\!\!+}\, s\sigma \mathrel{+\!\!\!\twoheadrightarrow}^{Q'}_{\vee\!\!/\Delta} s^{Q'}\sigma$ yielding $s\sigma' \mathrel{+\!\!\!\twoheadrightarrow}^{Q'}_{\vee\!\!/\Delta} s^{Q'}\sigma'$, which can be merged with $s\sigma' \mathrel{+\!\!\!\twoheadrightarrow}^{\{\pi\}}_{\vee\!\!/\Delta} s^{\pi}\sigma'$ to obtain $s\sigma' \mathrel{+\!\!\!\twoheadrightarrow}^{Q}_{\vee\!\!/\Delta} s'\sigma'$, and finally

iii. $s^{\pi}\sigma' \;{}_{\vee\!\!/\Gamma}\!\mathrel{\twoheadleftarrow\!\!\!+}\, s^{\pi}\sigma \mathrel{+\!\!\!\twoheadrightarrow}^{Q'}_{\vee\!\!/\Delta} s'\sigma$ yielding $s'\sigma \mathrel{+\!\!\!\twoheadrightarrow}_{\vee\!\!/\Gamma} s'\sigma'$, where $s^{\pi}\sigma \mathrel{+\!\!\!\twoheadrightarrow}^{Q'}_{\vee\!\!/\Delta} s'\sigma$ is obtained from part 1 of this lemma applied to $s^{Q'}\sigma \;{}^{Q'}_{\vee\!\!/\Delta}\!\mathrel{\twoheadleftarrow\!\!\!+}\, s\sigma \mathrel{+\!\!\!\twoheadrightarrow}^{\{\pi\}}_{\vee\!\!/\Delta} s^{\pi}\sigma$.


(a) Split base step.

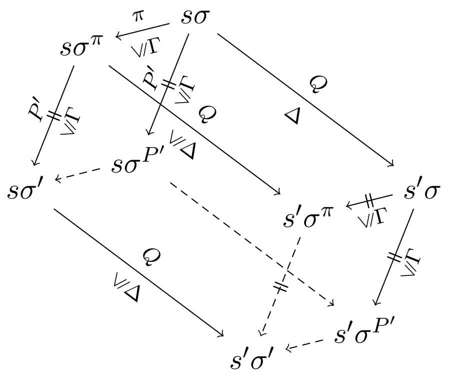

(b) Split substitution.

Figure 3.5.: Weak LL-labeling applied to nested parallel steps.

This concludes the induction step. If $|Q| = 0$, there is nothing to show, so only the base case $|Q| = 1$ remains. Note that because $\mathcal{R}$ is left-linear, we may assume without loss of generality that s is linear. Therefore, every rewrite step of $s\sigma \mathrel{+\!\!\!\!\twoheadrightarrow}^{P} s\sigma'$ can be performed by modifying σ. For $P' \subseteq P$, we write $\sigma^{P'}$ for the substitution τ that satisfies $s\sigma \mathrel{+\!\!\!\!\twoheadrightarrow}^{P'} s\tau$, and proceed by induction on $|P|$. For the induction step ($|P| > 1$), the argument is again almost the same as before, cf. Figure 3.5(b). Let $P = \{\pi\} \uplus P'$. We split $s\sigma \mathrel{+\!\!\!\!\twoheadrightarrow}_{\Gamma}^{P} s\sigma'$ into $s\sigma \mathrel{+\!\!\!\!\twoheadrightarrow}_{\curlyvee\!\Gamma}^{\{\pi\}} s\sigma^{\pi}$ and $s\sigma \mathrel{+\!\!\!\!\twoheadrightarrow}_{\curlyvee\!\Gamma}^{P'} s\sigma^{P'}$. Next we apply the induction hypothesis to the peaks

i. $s\sigma^{\pi} \;{}_{\curlyvee\!\Gamma}^{\{\pi\}}\!\!\twoheadleftarrow\!\!+ \; s\sigma \rightarrow_{\Delta}^{Q} s'\sigma$ yielding $s\sigma^{\pi} \rightarrow_{\curlyvee\!\Delta}^{Q} s'\sigma^{\pi}$ and $s'\sigma \mathrel{+\!\!\!\!\twoheadrightarrow}_{\curlyvee\!\Gamma} s'\sigma^{\pi}$,

ii. $s\sigma^{P'} \;{}_{\curlyvee\!\Gamma}^{P'}\!\!\twoheadleftarrow\!\!+ \; s\sigma \rightarrow_{\Delta}^{Q} s'\sigma$ yielding $s'\sigma \mathrel{+\!\!\!\!\twoheadrightarrow}_{\curlyvee\!\Gamma} s'\sigma^{P'}$, which can be merged with $s'\sigma \mathrel{+\!\!\!\!\twoheadrightarrow}_{\curlyvee\!\Gamma} s'\sigma^{\pi}$ to obtain $s'\sigma \mathrel{+\!\!\!\!\twoheadrightarrow}_{\curlyvee\!\Gamma} s'\sigma'$, and finally

iii. $s\sigma' \;{}_{\curlyvee\!\Gamma}^{P'}\!\!\twoheadleftarrow\!\!+ \; s\sigma^{\pi} \rightarrow_{\curlyvee\!\Delta}^{Q} s'\sigma^{\pi}$ yielding $s\sigma' \rightarrow_{\curlyvee\!\Delta}^{Q} s'\sigma'$, where $s\sigma^{\pi} \mathrel{+\!\!\!\!\twoheadrightarrow}_{\curlyvee\!\Gamma}^{P'} s\sigma'$ is obtained from part 1 of this lemma applied to $s\sigma^{\pi} \;{}_{\curlyvee\!\Gamma}^{\{\pi\}}\!\!\twoheadleftarrow\!\!+ \; s\sigma \mathrel{+\!\!\!\!\twoheadrightarrow}_{\curlyvee\!\Gamma}^{P'} s\sigma^{P'}$.

This concludes the induction step. If $|P| = 0$ then there is nothing to show. Finally, if $|P| = |Q| = 1$, then we are left with a parallel or variable overlap, and we conclude by Definition 3.10(1) or 3.10(2), respectively. This concludes the proof that $\Gamma \geqslant_H \Gamma'$ and $\Delta \geqslant_H \Delta'$. Now if $\sigma(x) = \sigma'(x)$ for all $x \in \mathcal{V}\mathsf{ar}(s'|_Q)$, then $s'\sigma \mathrel{+\!\!\!\!\twoheadrightarrow}^{P'} s'\sigma'$ satisfies $P' \parallel Q$. Performing the same rewrite steps on $s\sigma$, we obtain a parallel rewrite step $s\sigma \mathrel{+\!\!\!\!\twoheadrightarrow}^{P'} s''$ with $P' \subseteq P$ and therefore $\Gamma'' = \ell^{\parallel}(s\sigma \mathrel{+\!\!\!\!\twoheadrightarrow}^{P'} s'') \subseteq \ell^{\parallel}(s\sigma \mathrel{+\!\!\!\!\twoheadrightarrow}^{P} s\sigma') = \Gamma$. Finally, using the first part of this lemma, we can combine the two parallel steps from $s\sigma$ into a single one, $s\sigma \mathrel{+\!\!\!\!\twoheadrightarrow}_{\Gamma''\cup\Delta}^{P'\cup Q} s'\sigma'$ with $\Sigma = \Gamma'' \cup \Delta \subseteq \Gamma \cup \Delta$ as claimed. $\square$

Only Definition 3.10(1) was used in the proof of Lemma 3.45(1). This fact can be exploited for an alternative characterization of weak LL-labelings.

Corollary 3.46. *Let ℓ be a labeling. Then ℓ is a weak LL-labeling if and only if*

1. *in Figure 3.2(a), $\alpha \geqslant \gamma$ and $\beta \geqslant \delta$, and*
2. *in Figure 3.2(c), $\beta \geqslant \delta$ and $\{\alpha\} \geqslant_H \ell^{\|}(u \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow} v)$.*

Proof. Assume that ℓ is a weak LL-labeling. The first condition of this lemma is identical to Definition 3.10(1). For the second condition, $\beta \geqslant \delta$ follows from Definition 3.10(2). To establish $\{\alpha\} \geqslant_H \ell^{\|}(u \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}^P v)$, we need to show that $\alpha \geqslant \ell(u \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}^\pi u^\pi)$ for all $\pi \in P$. For each π, we can arrange that $\ell(u \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}^\pi u^\pi) = \gamma_1$ by choosing $u \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}^\pi u^\pi$ as the first step in the permutation of $u \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow} v$, and then $\alpha \geqslant \gamma_1$ follows from Definition 3.10(2), establishing the claim.

Next assume that ℓ satisfies the conditions of this lemma. Then the condition of Definition 3.10(1) holds. To show the conditions of Definition 3.10(2), note that $\beta \geqslant \delta$ holds by assumption. Consider the parallel rewrite step $u \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}^P v$ and a permutation $\pi_1, \ldots, \pi_n$ of P. We can decompose $u \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}^P v$ into a sequence $u = u_0 \to_{\gamma_1}^{\pi_1} u_1 \to_{\gamma_2}^{\pi_2} \cdots \to_{\gamma_n}^{\pi_n} u_n = v$. By Lemma 3.45(1) applied to the peaks

$$\begin{tikzcd}\cdot & u \arrow[l, "\{\pi_i\}"', "\vee\!\!\!/\{\alpha\}"] \arrow[r, "\{\pi_1,\ldots,\pi_{i-1}\}", "\vee\!\!\!/\{\alpha\}"'] & u_{i-1}\end{tikzcd}$$

we obtain $u_{i-1} \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}_{\vee\!\!\!/\{\alpha\}}^{\{\pi_i\}} u_i$, i.e., $\{\alpha\} \geqslant_H \{\gamma_i\}$, which is equivalent to $\alpha \geqslant \gamma_i$. Hence $\alpha \geqslant \overline{\gamma}$. ☐

The following lemma is used to reduce the number of parallel peaks that have to be considered in the proof of Theorem 3.50.

Lemma 3.47. *Let $s \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}_{\vee\Gamma}^* \cdot \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}_{\vee\!\!\!/\Delta} \cdot \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}_{\vee\Gamma\Delta}^* t$ and $s \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}_{\vee\Gamma}^* \cdot \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}_{\vee\!\!\!/\Delta} \cdot \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}_{\vee\Gamma\Delta}^* u$ be two rewrite sequences such that all rewrite steps in the sequence to t are at or below a position p and the rewrite steps in the sequence to u are parallel to p. Then the two rewrite sequences can be merged into $s \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}_{\vee\Gamma}^* \cdot \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}_{\vee\!\!\!/\Delta} \cdot \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}_{\vee\Gamma\Delta}^* u[t|_p]_p$.*

Proof. Let $s \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}_{\vee\Gamma}^* t_1 \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}_{\vee\!\!\!/\Delta} t_2 \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}_{\vee\Gamma\Delta}^* t$ and $s \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}_{\vee\Gamma}^* u_1 \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}_{\vee\!\!\!/\Delta} u_2 \mathrel{-\!\!\shortparallel\!\!\twoheadrightarrow}_{\vee\Gamma\Delta}^* u$. Using Lemma 3.45(1) repeatedly, we can derive a sequence

$$\begin{tikzcd}s \arrow[r, "*", "\vee\Gamma"'] & u_1[t_1|_p]_p \arrow[r, "\vee\!\!\!/\Delta"'] & u_2[t_2|_p]_p \arrow[r, "*", "\vee\Gamma\Delta"'] & u[t|_p]_p\end{tikzcd}$$

which establishes the claim. ☐

In order to perform a critical pair analysis for parallel rewrite steps, we need parallel critical pairs [64, 181].

Definition 3.48. Let $l \to r$ be a rule in a TRS $\mathcal{R}$ and P be a non-empty set of pairwise parallel redex patterns such that every $\pi \in P$ critically overlaps with l. By choosing variants of rules from $\mathcal{R}$ appropriately, we may assume that the sets $\mathcal{V}\mathsf{ar}(l_\pi)$ for $\pi \in P$ and $\mathcal{V}\mathsf{ar}(l)$ are pairwise disjoint. Assume that the unification problem $\{l|_{p_\pi} \approx l_\pi \mid \pi \in P\}$ has a solution and let σ be a most general unifier. Then there is a unique term l_P such that $l\sigma \mathrel{-\!\!\!\|\!\!\!\to}^P l_P$. We call $l_P \mathrel{\leftarrow\!\!\!\|\!\!\!-}\!\rtimes\!\!\rightarrow r\sigma$ a *parallel critical pair*, and $l_P \mathrel{\leftarrow\!\!\!\|\!\!\!-} l\sigma \to r\sigma$ a *parallel critical peak.*

Note that every standard critical pair also is a parallel critical pair. The following lemma states how critical pair analysis for a peak consisting of a parallel and a root rewrite step is done. It is a straightforward extension of [64, Lemma 4.7].

Lemma 3.49. *Let $\mathcal{R}$ be a left-linear TRS and $t \; {}^P\!\!\mathrel{\leftarrow\!\!\!\|\!\!\!-} s \to^\pi u$ with $p_\pi = \epsilon$. Then either $P \perp \pi$ or there are substitutions $\sigma \mathrel{-\!\!\!\|\!\!\!\to} \sigma'$ and a parallel critical pair $t' \mathrel{\leftarrow\!\!\!\|\!\!\!-}\!\rtimes\!\!\rightarrow u'$ such that $t = t'\sigma' \; {}^{P \setminus P'}\!\!\mathrel{\leftarrow\!\!\!\|\!\!\!-} t'\sigma \; {}^{P'}\!\!\mathrel{\leftarrow\!\!\!\|\!\!\!-} s \to u'\sigma = u$ with $P' \subseteq P$.* $\square$

Note that left-linearity is essential for the substitutions σ and σ' to exist in Lemma 3.49. We are now ready for the main theorem of this section.

Theorem 3.50. *A left-linear TRS $\mathcal{R}$ is confluent if all its parallel critical peaks $t \; {}^{P}_{\Gamma}\!\!\mathrel{\leftarrow\!\!\!\|\!\!\!-} s \to_\Delta u$ can be joined decreasingly as*

$$t \xrightarrow[\vee\Gamma]{*} \cdot \underset{\vee\!\!/\Delta}{\mathrel{-\!\!\!\|\!\!\!\to}} \cdot \xrightarrow[\vee\Gamma\Delta]{*} \cdot \xleftarrow[\vee\Gamma\Delta]{*} v \underset{\vee\!\!/\Gamma}{\overset{Q}{\mathrel{\leftarrow\!\!\!\|\!\!\!-}}} \cdot \xleftarrow[\vee\Delta]{*} u$$

such that $\mathcal{V}\mathsf{ar}(v|_Q) \subseteq \mathcal{V}\mathsf{ar}(s|_P)$.

Proof. We show that $\mathrel{-\!\!\!\|\!\!\!\to}$ is decreasing, which implies confluence of $\mathcal{R}$. Consider $t \; {}^{P}_{\Gamma}\!\!\mathrel{\leftarrow\!\!\!\|\!\!\!-} s \mathrel{-\!\!\!\|\!\!\!\to}^Q_\Delta u$. It suffices to show that

$$t \xrightarrow[\vee\Gamma]{*} \cdot \underset{\vee\!\!/\Delta}{\mathrel{-\!\!\!\|\!\!\!\to}} \cdot \xrightarrow[\vee\Gamma\Delta]{*} \cdot \xleftarrow[\vee\Gamma\Delta]{*} \cdot \underset{\vee\!\!/\Gamma}{\mathrel{\leftarrow\!\!\!\|\!\!\!-}} \cdot \xleftarrow[\vee\Delta]{*} u \tag{2}$$

Below we show that (2) holds whenever $P = \{\pi\}$ or $Q = \{\pi\}$ with $p_\pi = \epsilon$. Then for all $p \in \min\{p_\pi \mid \pi \in P \cup Q\}$, $t \; {}^{P}_{\Gamma}\!\!\mathrel{\leftarrow\!\!\!\|\!\!\!-} s \mathrel{-\!\!\!\|\!\!\!\to}^Q_\Delta u$ induces a peak $t|_p \; {}^{P'}_{\Gamma_0}\!\!\mathrel{\leftarrow\!\!\!\|\!\!\!-} s|_p \mathrel{-\!\!\!\|\!\!\!\to}^{Q'}_{\Delta_0} u|_p$, where $P' = \{\pi\}$ or $Q' = \{\pi\}$ for some π with $p_\pi = \epsilon$. So for each p, we obtain a joining sequence for $t|_p$ and $u|_p$ of shape (2). By the monotonicity of labelings, this results in joining sequences

$$s[t|_p]_p \xrightarrow[\vee\Gamma]{*} \cdot \underset{\vee\!\!/\Delta}{\mathrel{-\!\!\!\|\!\!\!\to}} \cdot \xrightarrow[\vee\Gamma\Delta]{*} \cdot \xleftarrow[\vee\Gamma\Delta]{*} \cdot \underset{\vee\!\!/\Gamma}{\mathrel{\leftarrow\!\!\!\|\!\!\!-}} \cdot \xleftarrow[\vee\Delta]{*} s[u|_p]_p$$

which are mutually parallel since the positions $p \in \min(P \cup Q)$ are mutually parallel. By repeated application of Lemma 3.47 those sequences can be combined into a single sequence of the same shape.

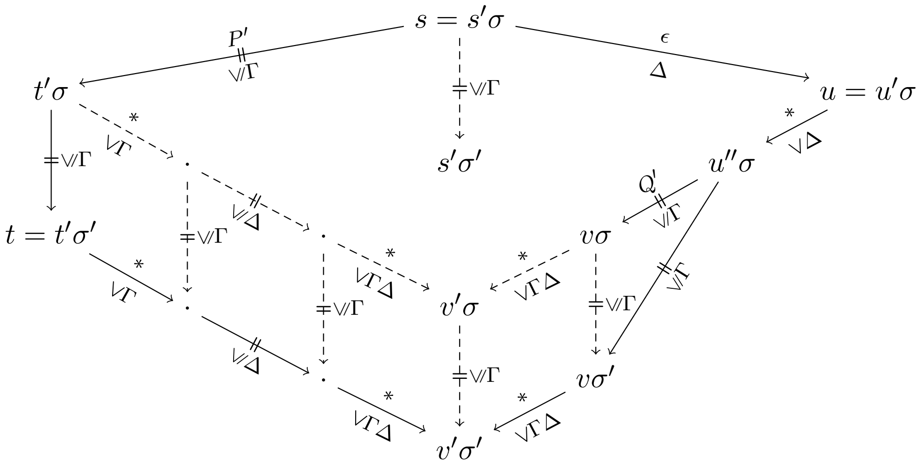


Figure 3.6.: Part of the proof of Theorem 3.50.

In order to show (2) for $P = \{\pi\}$ or $Q = \{\pi\}$ with $p_\pi = \epsilon$, assume without loss of generality that $Q = \{\pi\}$. If $P \perp \pi$ then $s = l_\pi\sigma$ and, because l_π is linear, there is a substitution σ' with $t = l_\pi\sigma'$ and $\sigma(x) \mathrel{+\!\!\!\!\!\!\rightarrow} \sigma'(x)$ for all variables $x \in \mathcal{V}$. We conclude by Lemma 3.45(2). Otherwise P and π overlap, and by Lemma 3.49, there are a parallel critical peak $t' \; {}^{P'}\!\!\mathrel{\leftarrow\!\!\!\!\!\!+} s' \to u'$ and substitutions σ, σ' such that $\sigma \mathrel{+\!\!\!\!\!\!\rightarrow} \sigma'$ and $t = t'\sigma' \; {}^{P\setminus P'}_{\not\vee\Gamma}\!\!\mathrel{\leftarrow\!\!\!\!\!\!+} t'\sigma \; {}^{P'}_{\not\vee\Gamma}\!\!\mathrel{\leftarrow\!\!\!\!\!\!+} s'\sigma = s \to^{\epsilon}_{\Delta} u'\sigma = u$ with $P' \subseteq P$. This case is illustrated in Figure 3.6. By assumption there are u'', v and v' with $\mathcal{V}\mathsf{ar}(v|_{Q'}) \subseteq \mathcal{V}\mathsf{ar}(s|_{P'})$ such that we can join t' and u' decreasingly, and consequently, using the stability of labelings we obtain

$$t'\sigma \xrightarrow[\vee\Gamma]{*} \cdot \xrightarrow[\not\vee\Delta]{+\!\!+} \cdot \xrightarrow[\vee\Gamma\Delta]{*} v'\sigma \xleftarrow[\vee\Gamma\Delta]{*} v\sigma \overset{Q'}{\underset{\not\vee\Gamma}{\leftarrow\!\!\!\!\!\!+\!\!+}} u''\sigma \xleftarrow[\vee\Delta]{*} u'\sigma = u$$

Furthermore, making repeated use of Lemma 3.45(2),

$$t = t'\sigma' \xrightarrow[\vee\Gamma]{*} \cdot \xrightarrow[\not\vee\Delta]{+\!\!+} \cdot \xrightarrow[\vee\Gamma\Delta]{*} v'\sigma' \xleftarrow[\vee\Gamma\Delta]{*} v\sigma' \underset{\not\vee\Gamma}{\leftarrow\!\!\!\!\!\!+\!\!+} v\sigma$$

Notably, the step $v\sigma \mathrel{+\!\!\!\!\!\!\rightarrow}_{\not\vee\Gamma} v\sigma'$ is obtained from $s'\sigma \mathrel{+\!\!\!\!\!\!\rightarrow}_{\not\vee\Gamma} s'\sigma'$ by passing through the rewrite sequence $s'\sigma \to u'\sigma \to^* u''\sigma \mathrel{+\!\!\!\!\!\!\rightarrow} v\sigma$. We have $\sigma(x) = \sigma'(x)$ for $x \in \mathcal{V}\mathsf{ar}(s|_{P'})$ for otherwise $s \mathrel{+\!\!\!\!\!\!\rightarrow}_{\Gamma} t$ would not be a parallel step. Together with the assumption $\mathcal{V}\mathsf{ar}(v|_{Q'}) \subseteq \mathcal{V}\mathsf{ar}(s|_{P'})$, the parallel steps $u''\sigma \mathrel{+\!\!\!\!\!\!\rightarrow}_{\not\vee\Gamma} v\sigma$ and $v\sigma \mathrel{+\!\!\!\!\!\!\rightarrow}_{\not\vee\Gamma} v\sigma'$ can be combined into a single $\mathrel{+\!\!\!\!\!\!\rightarrow}_{\not\vee\Gamma}$ step by Lemma 3.45(2). Thus we can join t and u decreasingly with common reduct $v'\sigma'$, completing the proof. □

To conclude the section we demonstrate Theorem 3.50 on two examples. Both are based on rule labeling.

Example 3.51. Consider the TRS $\mathcal{R}$ consisting of the following five rules with labels $2 > 1 > 0$:

$$\mathsf{a} \xrightarrow[1]{} \mathsf{b} \qquad \mathsf{b} \xrightarrow[0]{} \mathsf{a} \qquad \mathsf{f}(\mathsf{a},\mathsf{a}) \xrightarrow[1]{} \mathsf{c} \qquad \mathsf{f}(\mathsf{b},\mathsf{b}) \xrightarrow[2]{} \mathsf{c} \qquad \mathsf{h}(x) \xrightarrow[0]{} \mathsf{h}(\mathsf{f}(x,x))$$

There are six parallel critical peaks that can all be joined decreasingly as required by Theorem 3.50:

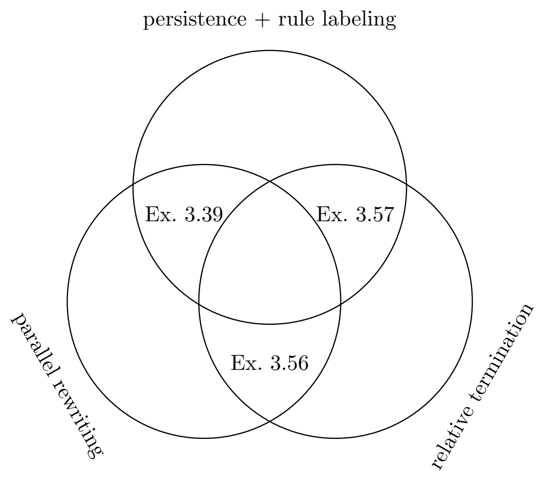
$$\begin{array}{rl}
\mathsf{f}(\mathsf{b},\mathsf{a}) \underset{\{1\}}{\leftarrow\!\shortparallel\!-} \mathsf{f}(\mathsf{a},\mathsf{a}) \xrightarrow[\{1\}]{} \mathsf{c}: & \mathsf{f}(\mathsf{b},\mathsf{a}) \xrightarrow[\{0\}]{} \mathsf{f}(\mathsf{a},\mathsf{a}) \xrightarrow[\{1\}]{} \mathsf{c} \\
\mathsf{f}(\mathsf{a},\mathsf{b}) \underset{\{1\}}{\leftarrow\!\shortparallel\!-} \mathsf{f}(\mathsf{a},\mathsf{a}) \xrightarrow[\{1\}]{} \mathsf{c}: & \mathsf{f}(\mathsf{a},\mathsf{b}) \xrightarrow[\{0\}]{} \mathsf{f}(\mathsf{a},\mathsf{a}) \xrightarrow[\{1\}]{} \mathsf{c} \\
\mathsf{f}(\mathsf{b},\mathsf{b}) \underset{\{1\}}{\leftarrow\!\shortparallel\!-} \mathsf{f}(\mathsf{a},\mathsf{a}) \xrightarrow[\{1\}]{} \mathsf{c}: & \mathsf{f}(\mathsf{b},\mathsf{b}) \underset{\{0\}}{-\!\shortparallel\!\rightarrow} \mathsf{f}(\mathsf{a},\mathsf{a}) \xrightarrow[\{1\}]{} \mathsf{c} \\
\mathsf{f}(\mathsf{a},\mathsf{b}) \underset{\{0\}}{\leftarrow\!\shortparallel\!-} \mathsf{f}(\mathsf{b},\mathsf{b}) \xrightarrow[\{2\}]{} \mathsf{c}: & \mathsf{f}(\mathsf{a},\mathsf{b}) \xrightarrow[\{0\}]{} \mathsf{f}(\mathsf{a},\mathsf{a}) \xrightarrow[\{1\}]{} \mathsf{c} \\
\mathsf{f}(\mathsf{b},\mathsf{a}) \underset{\{0\}}{\leftarrow\!\shortparallel\!-} \mathsf{f}(\mathsf{b},\mathsf{b}) \xrightarrow[\{2\}]{} \mathsf{c}: & \mathsf{f}(\mathsf{b},\mathsf{a}) \xrightarrow[\{0\}]{} \mathsf{f}(\mathsf{a},\mathsf{a}) \xrightarrow[\{1\}]{} \mathsf{c} \\
\mathsf{f}(\mathsf{a},\mathsf{a}) \underset{\{0\}}{\leftarrow\!\shortparallel\!-} \mathsf{f}(\mathsf{b},\mathsf{b}) \xrightarrow[\{2\}]{} \mathsf{c}: & \mathsf{f}(\mathsf{a},\mathsf{a}) \xrightarrow[\{1\}]{} \mathsf{c}
\end{array}$$

Therefore, $\mathcal{R}$ is confluent.

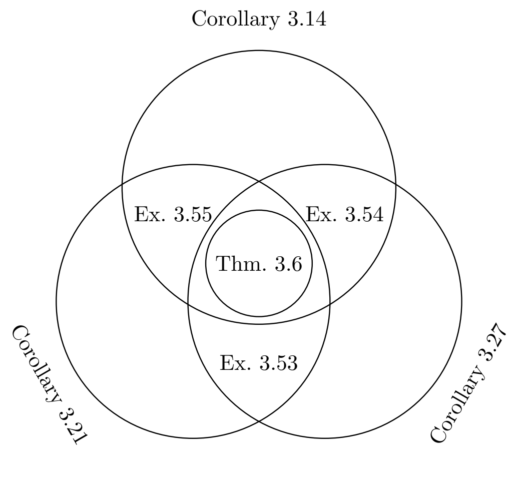
Example 3.52. Let $\mathcal{R}$ be the TRS (Cops #62) consisting of the (labeled) rules

$$\begin{array}{rlrlrl}
x - 0 & \xrightarrow[0]{} x & 0 - x & \xrightarrow[0]{} 0 & \mathsf{s}(x) - \mathsf{s}(y) & \xrightarrow[0]{} x - y \\
0 < \mathsf{s}(x) & \xrightarrow[0]{} \mathsf{true} & x < 0 & \xrightarrow[0]{} \mathsf{false} & \mathsf{s}(x) < \mathsf{s}(y) & \xrightarrow[0]{} x < y \\
\mathsf{gcd}(x,0) & \xrightarrow[0]{} x & \mathsf{gcd}(0,x) & \xrightarrow[0]{} x & \mathsf{gcd}(x,y) & \xrightarrow[1]{} \mathsf{gcd}(y,\mathsf{mod}(x,y)) \\
\mathsf{if}(\mathsf{true},x,y) & \xrightarrow[0]{} x & \mathsf{if}(\mathsf{false},x,y) & \xrightarrow[0]{} y & & \\
\mathsf{mod}(x,0) & \xrightarrow[0]{} x & \mathsf{mod}(0,x) & \xrightarrow[0]{} 0 & & \\
\mathsf{mod}(x,\mathsf{s}(y)) & \multicolumn{5}{l}{\xrightarrow[1]{} \mathsf{if}(x < \mathsf{s}(y), x, \mathsf{mod}(x - \mathsf{s}(y), \mathsf{s}(y)))}
\end{array}$$

There are 12 critical pairs, 6 of which are trivial. One easily verifies that the remaining 6 pairs can be joined decreasingly, using the order $1 > 0$. Hence the confluence of $\mathcal{R}$ follows from Theorem 3.50. Even though $\mathcal{R}$ lacks proper parallel critical pairs, none of the other results in this paper applies. Note that the preconditions for Corollaries 3.14, 3.21, and 3.27 are not satisfied as $\mathcal{R}_{\mathsf{d}}/\mathcal{R}_{\mathsf{nd}}$, $\star(\mathcal{R})$, and $\mathcal{R}^{\triangle}$ are non-terminating (due to the rules with label 1). Finally, persistence cannot rule out variable overlaps (of the duplicating mod rule below the variable x) and hence Theorems 3.36 and 3.38 based on the rule labeling fail.



(a) All results.



(b) Relative termination results.

Figure 3.7.: Interrelationships.

3.5. Assessment

In this section we relate the results from this article to each other (Section 3.5.1) and to the recent literature [4, 71] (Section 3.5.2).

3.5.1. Interrelationships

The main results for left-linear systems presented in this article can be divided into three classes. Those that require relative termination as a precondition (Corollaries 3.14, 3.21, and 3.27), those exploiting persistence (Theorems 3.36 and 3.38), and those considering parallel rewriting (Theorem 3.50). Figure 3.7(a) demonstrates that these three classes are incomparable. The same holds when focusing on the results relying on relative termination, cf. Figure 3.7(b). Note that the regions where only one class is applicable can be populated with examples using Toyama's celebrated modularity result [180], e.g., the disjoint union (after renaming function symbols) of the TRSs in Examples 3.56 and 3.57 can only be handled by the approach based on relative termination. We discuss the interrelationships in more detail below.

First we observe that Corollaries 3.14, 3.21, and 3.27 subsume Theorem 3.6 since the preconditions of the corollaries evaporate for linear systems. The inclusion is strict since Theorem 3.6 cannot deal with the rule $\mathsf{f}(x) \to \mathsf{g}(x,x)$, while all the corollaries can. Furthermore, Theorem 3.6 is subsumed by Theorem 3.36, which, if restricted to weak LL-labelings, is subsumed by Theorem 3.38.

The following three examples show that Corollaries 3.14, 3.21, and 3.27 are pairwise incomparable in power (for an overview see Figure 3.7(b)).

Example 3.53. Consider the TRS $\mathcal{R}$ consisting of the following rules

$$\begin{array}{rcr} \mathsf{f}(\mathsf{h}(x)) \to \mathsf{k}(\mathsf{g}(\mathsf{f}(x), x, \mathsf{f}(\mathsf{h}(\mathsf{a})))) & \mathsf{f}(x) \to \mathsf{a} & \mathsf{a} \to \mathsf{b} \\ \mathsf{k}(x) \to \mathsf{c} & \mathsf{b} \to \bot & \mathsf{c} \to \bot \end{array}$$

This TRS has one critical peak (modulo symmetry). Corollary 3.14 does not apply since $\mathcal{R}_{\mathsf{d}}/\mathcal{R}_{\mathsf{nd}}$ is non-terminating, For Corollary 3.21 observe that $\star(\mathcal{R})$ is terminating using the interpretation $\mathsf{h}_{1\mathbb{N}}(x) = x + 1$ and the identify function for all other function symbols. To show decreasingness we use the labeling $\ell_\star \times \ell^i_{\mathrm{rl}}$ with $i(\mathsf{f}(x) \to \mathsf{a}) = 1$ and all other rules receive label 0. The critical peak $t = \mathsf{a} \ {}_{x,1}\!\leftarrow \mathsf{f}(\mathsf{h}(x)) \to_{x,0} \mathsf{k}(\mathsf{g}(\mathsf{f}(x), x, \mathsf{f}(\mathsf{h}(\mathsf{a})))) = u$ is closed decreasingly by the join $t \to_{x,0} \mathsf{b} \to_{x,0} \bot \ {}_{x,0}\!\leftarrow \mathsf{c} \ {}_{x,0}\!\leftarrow u$. Corollary 3.27 also applies since the polynomial interpretation with $\mathsf{h}_{\mathbb{N}}(x) = 3x + 1$ and interpreting all other function symbols by the sum of its arguments establishes termination of $\mathcal{R}^{\triangle}/\mathcal{R}$. The critical peak $t = \mathsf{a} \ {}_{3x+1}\!\leftarrow \mathsf{f}(\mathsf{h}(x)) \to_{3x+1} \mathsf{k}(\mathsf{g}(\mathsf{f}(x), x, \mathsf{f}(\mathsf{h}(\mathsf{a})))) = u$ can be closed decreasingly by $t \to_0 \mathsf{b} \to_0 \bot \ {}_0\!\leftarrow \mathsf{c} \ {}_{2x+1}\!\leftarrow u$, when taking the identity for ℓ in Corollary 3.27.

Example 3.54. It is easy to adapt the TRS from Example 3.16 such that $\star(\mathcal{R})$ becomes non-terminating. Consider the TRS $\mathcal{R}$

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

for which termination of $\mathcal{R}_{\mathsf{d}}/\mathcal{R}_{\mathsf{nd}}$ is proved by LPO with precedence $\mathsf{f} > \mathsf{g}$ and $\mathsf{a} \sim \mathsf{b} > \mathsf{c}$. Corollary 3.14 applies since the rule labeling establishes decreasingness of the critical peak $t = \mathsf{f}(\mathsf{g}(x, \mathsf{b})) \ {}_2\!\leftarrow \mathsf{f}(\mathsf{g}(x, \mathsf{a})) \to_3 \mathsf{g}(\mathsf{f}(x), \mathsf{f}(\mathsf{g}(x, \mathsf{c}))) = u$ by the join $t \to_1 \mathsf{f}(\mathsf{g}(x, \mathsf{a})) \to_3 u$. Note that $\mathsf{f}_1(\mathsf{g}_1(x)) \to \mathsf{g}_2(\mathsf{f}_1(\mathsf{g}_1(x))) \in \mathcal{R}^\star_>$ is non-terminating and hence Corollary 3.21 does not apply.[3] For Corollary 3.27 the (above) termination proof establishes termination of $\mathcal{R}^{\triangle}/\mathcal{R}$ and $\ell_\triangle$ in combination with the rule labeling (taking rule numbers as labels) labels the critical peak $t = \mathsf{f}(\mathsf{g}(x, \mathsf{b})) \ {}_{\mathsf{a},2}\!\leftarrow \mathsf{f}(\mathsf{g}(x, \mathsf{a})) \to_{\mathsf{f}(\mathsf{g}(x,\mathsf{a})),3} \mathsf{g}(\mathsf{f}(x), \mathsf{f}(\mathsf{g}(x, \mathsf{c}))) = u$ decreasingly since $t \to_{\mathsf{b},1} \mathsf{f}(\mathsf{g}(x, \mathsf{a})) \to_{\mathsf{f}(\mathsf{g}(x,\mathsf{a})),3} u$.

[3]We remark that it is easy to extend this example such that also $\overset{\star}{\star}(\mathcal{R})$ is non-terminating; just consider the rule $\mathsf{f}(\mathsf{g}(x, \mathsf{a})) \to \mathsf{g}(\mathsf{f}(x), \mathsf{g}(\mathsf{f}(\mathsf{g}(x, \mathsf{c}), \mathsf{f}(\mathsf{g}(x, \mathsf{c})))))$.

Example 3.55. Consider the TRS consisting of the rules

$$\mathsf{a}(\mathsf{a}(\mathsf{c})) \to \mathsf{a}(\mathsf{b}(\mathsf{a}(\mathsf{c}))) \qquad\qquad \mathsf{b}(x) \to \mathsf{h}(x,x)$$

The TRS $\mathcal{R}$ has no critical peaks and is terminating by the following matrix interpretation over $\mathbb{N}^2$:

$$\mathsf{a}_{\mathbb{N}^2}(\vec{x}) = \begin{pmatrix}1 & 1\\ 1 & 2\end{pmatrix}\vec{x} + \begin{pmatrix}0\\3\end{pmatrix} \qquad \mathsf{h}_{\mathbb{N}^2}(\vec{x},\vec{y}) = \begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix}\vec{x} + \begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix}\vec{y}$$
$$\mathsf{b}_{\mathbb{N}^2}(\vec{x}) = \begin{pmatrix}2 & 0\\ 0 & 0\end{pmatrix}\vec{x} + \begin{pmatrix}2\\0\end{pmatrix} \qquad \mathsf{c}_{\mathbb{N}^2} = \begin{pmatrix}0\\0\end{pmatrix}$$

Hence also $\mathcal{R}_{\mathsf{d}}/\mathcal{R}_{\mathsf{nd}}$ is terminating, and by Corollary 3.14 the TRS $\mathcal{R}$ is confluent. Corollary 3.21 also applies since $\star(\mathcal{R})$ is terminating. The derivation $\mathsf{a}(\mathsf{a}(\mathsf{c})) \to \mathsf{a}(\mathsf{b}(\mathsf{a}(\mathsf{c}))) \to_{\mathcal{R}^{\triangle}} \mathsf{a}(\mathsf{a}(\mathsf{c})) \to \cdots$ shows that $\mathcal{R}^{\triangle}/\mathcal{R}$ is non-terminating, so Corollary 3.27 does not apply.

Note that any simple monotone reduction pair showing termination of $\mathcal{R}_{\mathsf{d}}/\mathcal{R}_{\mathsf{nd}}$ will also establish termination of $\mathcal{R}^{\triangle}/\mathcal{R}$, because if $l \to x \in \mathcal{R}^{\triangle}$ then there is a rule $l \to r \in \mathcal{R}_{\mathsf{d}}$ that duplicates x, whence $l > r \geqslant x$. Hence it is no surprise that Example 3.55 used a matrix interpretation of dimension 2.

Furthermore, the results on relative termination are incomparable with those on persistence and those based on parallel rewriting. To this end observe that the first rule of Example 3.39 violates all preconditions of Corollaries 3.14, 3.21, and 3.27 but Theorems 3.38 and 3.50 apply. Note that Theorem 3.38 based on arbitrary weak LL-labelings subsumes Corollaries 3.14 and 3.21, since they produce LL-labelings which may be used to close problematic variable peaks decreasingly even without persistence. However, if restricted to the rule labeling the following TRS cannot be handled using persistence while each of the Corollaries 3.14, 3.21, and 3.27 as well as Theorem 3.50 succeeds.

Example 3.56. Consider the TRS consisting of the rules

$$1\colon\ \mathsf{f}(x,y,\mathsf{a}) \to \mathsf{f}(x,x,\mathsf{b}) \qquad\qquad 2\colon\ \mathsf{f}(\mathsf{f}(x,y,\mathsf{b}),z,\mathsf{c}) \to x$$

which is orthogonal. Since a most general sort assignment cannot exclude variable overlaps of the first rule with itself, Theorem 3.38 can only succeed when used in combination with an LL-labeling. Note that all preconditions for Corollaries 3.14, 3.21, and 3.27 are satisfied and due to the lack of critical overlaps they are decreasing. For the same reason Theorem 3.50 applies.

The final example shows that Theorem 3.50 does not subsume the plain version for linear TRSs (because of the variable condition).

Example 3.57. Consider the linear TRS consisting of the single rule

$$(x + y) + z \to (z + y) + x$$

Note that all steps are labeled the same, because they use the same rule. There is only one (parallel) critical peak, $((z{+}y){+}x){+}u \leftarrow ((x{+}y){+}z){+}u \to (u{+}z){+}(x{+}y)$, which may be joined as $((z + y) + x) + u \to ((x + y) + z) + u \leftarrow (u + z) + (x + y)$. Confluence of $\mathcal{R}$ can be established by Theorem 3.6 using the rule labeling from Lemma 3.7. On the other hand, trying to use Theorem 3.50 fails for this joining sequence, because $\mathcal{V}\mathsf{ar}(((z + y) + x) + u) \nsubseteq \mathcal{V}\mathsf{ar}((z + y) + x)$. All other ways of joining the critical peak fail to be decreasing because they require more than one parallel rewrite step from $((z{+}y){+}x){+}u$ or $(u{+}z){+}(x{+}y)$, e.g. $((z{+}y){+}x){+}u \to ((x + y) + z) + y \to (y + z) + (x + y)$.

3.5.2. Related work

In this section we relate our results to [4, 71]. To compare our setting with the main result from [71] we define the *critical pair steps* $\mathsf{CPS}(\mathcal{R}) = \{s \to t, s \to u \mid t \leftarrow s \to u \text{ is a critical peak of } \mathcal{R}\}$. Furthermore let $\mathsf{CPS}'(\mathcal{R})$ be the critical pair steps which do not give rise to trivial critical pairs.

Theorem 3.58 ([71, Theorem 3]). *A left-linear locally confluent TRS $\mathcal{R}$ is confluent if $\mathsf{CPS}'(\mathcal{R})/\mathcal{R}$ is terminating.*

Using the weak LL-labeling $\ell_{\mathrm{rt}}^{\mathsf{PCPS}'(\mathcal{R})}$, from Theorem 3.50 we obtain the following corollary. Here $\mathsf{PCPS}'(\mathcal{R})$ are the parallel critical pair steps which do not give rise to trivial parallel critical pairs.

Corollary 3.59. *A left-linear TRS $\mathcal{R}$ whose parallel critical pairs are joinable is confluent if $\mathsf{PCPS}'(\mathcal{R})/\mathcal{R}$ is terminating.*

Proof. We need to show that the relative termination assumption eliminates the variable condition in Theorem 3.50. If $\mathsf{PCPS}'(\mathcal{R})/\mathcal{R}$ is terminating then for any (non-trivial) parallel critical peak $t \;{}_{\Gamma}^{P}\!\nleftarrow\!\!\!\shortparallel\; s \to_{\Delta} u$ we obtain $t \to_{\vee\Gamma}^{*} \cdot {}_{\vee\Delta}^{*}\!\!\leftarrow u$, hence Q can be chosen to be empty and $\varnothing = \mathcal{V}\mathsf{ar}(v|_{\varnothing}) \subseteq \mathcal{V}\mathsf{ar}(s|_P)$ trivially holds.[4] $\square$

We stress that despite the fact that the preconditions in Corollary 3.59 require more (implementation) effort to check than those in Theorem 3.58, in theory Corollary 3.59 subsumes Theorem 3.58. To this end observe that termination

[4]The condition that $\{s \to t \mid u \leftarrow s \to t \text{ is a critical pair}\}/\mathcal{R}$ is terminating also eliminates the variable condition.

of $\mathsf{PCPS}'(\mathcal{R})/\mathcal{R}$ is equivalent to termination of $\mathsf{CPS}'(\mathcal{R})/\mathcal{R}$. Furthermore joinability of the parallel critical pairs is a necessary condition for confluence just as local confluence is.

Due to the flexibility of the $\ell^{\mathcal{S}}_{\mathrm{rt}}$ labeling we can choose $\mathcal{S}$ to be (a subset of) the *critical diagram steps* $\mathsf{CDS}(\mathcal{R}) = \{s \to t_i, s \to u_j \mid \ t_0 \leftarrow s \to u_0$ is a critical peak in $\mathcal{R}$, $t_0 \to^* t_n = u_m \ {}^*\!\leftarrow u_0$, $0 \leqslant i \leqslant n$, and $0 \leqslant j \leqslant m\}$. Using $\mathsf{CDS}(\mathcal{R})$ allows to detect a possible decrease also somewhere in the joining part of the diagrams.[5] This incorporates (and generalizes) the idea of critical valleys [141]. However, we remark that our setting does not (yet) follow another recent trend, i.e., to drop development closed critical pairs (see [69, 141]). We leave this for future work.

Next we show that Corollary 3.21 generalizes the results from [4, Sections 5 and 6]. It is not difficult to see that the encoding presented in [4, Theorem 5.4] can be mimicked by Corollary 3.21 where linear polynomial interpretations over $\mathbb{N}$ of the shape as in (1)

$$(1) \quad f_{i\mathbb{N}}(x) = x + c_f \qquad\qquad (2) \quad f_{i\mathbb{N}}(x) = x + c_{f_i}$$

are used to prove termination of $\star(\mathcal{R})$ and $\ell_\star \times \ell_{\mathrm{rl}}$ is employed to show LL-decreasingness of the critical peaks. In contrast to [4, Theorem 5.4], which explicitly encodes these constraints in a single formula of linear arithmetic, our abstract formulation has the following advantages. First, we do not restrict to weight functions but allow powerful machinery for proving relative termination and second our approach allows to combine arbitrarily many labelings lexicographically (cf. Lemma 3.13). Furthermore we stress that our abstract treatment of $\star(\mathcal{R})$ allows to implement Corollary 3.21 based on $\overset{\star}{\star}(\mathcal{R})$ (cf. Section 3.6) which admits further gains in power (cf. Example 3.1 as well as Section 3.7).

The idea of the extension presented in [4, Example 6.1] amounts to using $\ell_{\mathrm{rl}} \times \ell_\star$ instead of $\ell_\star \times \ell_{\mathrm{rl}}$, which is an application of Lemma 3.13 in our setting. Finally, the extension discussed in [4, Example 6.3] suggests to use linear polynomial interpretations over $\mathbb{N}$ of the shape as in (2) to prove termination of $\star(\mathcal{R})$. Note that these interpretations are still weight functions. This explains why the approach from [4] fails to establish confluence of the TRSs in Examples 3.16 and 3.17 since a weight function cannot show termination of the rules $\mathsf{f}_1(\mathsf{g}_1(x)) \to \mathsf{g}_1(\mathsf{f}_1(x))$ and $\mathsf{f}_1(\mathsf{h}_1(x)) \to \mathsf{h}_1(\mathsf{g}_1(\mathsf{f}_1(x)))$, respectively.

Note that both recent approaches [4, 71] based on decreasing diagrams fail to prove the TRS $\mathcal{R}$ from Example 3.1 confluent. The former can, e.g., not cope with the non-terminating rule $\times_1(x) \to +_0(\times_1(x))$ in $\mathcal{R}^\star_>$ (cf. Example 3.24) while

[5] In [201] we employed the strictly weaker system where all steps of the join (e.g., $t_i \to t_{i+1}$) are used whereas here we use $s \to t_{i+1}$.

overlaps with the non-terminating rule $x + y \to y + x \in \mathcal{R}$ prevent the latter approach from succeeding. In contrast, Examples 3.15 and 3.24 give two confluence proofs based on our setting.

3.6. Implementation

In this section we sketch how the results from this article can be implemented.

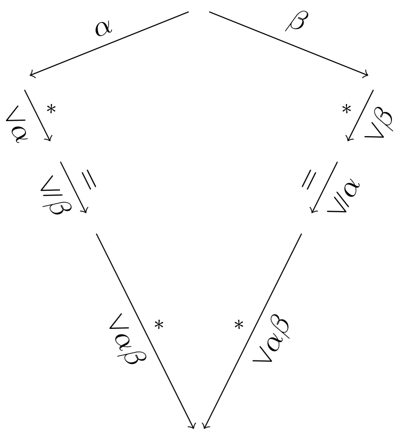
Before decreasingness of critical peaks can be investigated, the critical pairs must be shown to be convergent. For a critical pair $t \leftarrow\!\rtimes\!\rightarrow u$ in our implementation we consider all joining sequences such that $t \to^{\leqslant n} \cdot {}^{\leqslant n}\!\leftarrow u$ and there is no smaller n that admits a common reduct. While in theory longer joining sequences might be easier to label decreasingly, preliminary experiments revealed that the effort due to the consideration of additional diagrams decreased performance.

To exploit the possibility for incremental confluence proofs by lexicographically combining labels (cf. Lemmata 3.9 and 3.13) our implementation considers lists of labels. The search for relative termination proofs (and thus the labelings) is implemented by encoding the constraints in non-linear (integer) arithmetic. Below we describe how we combine existing labels (some partial progress) with the search for a new labeling to show the critical peaks decreasing. Note that labelings use different domains (natural numbers, terms), and, even worse, different orders (matrix interpretations, LPO, etc.). The crucial observation for incremental labeling is that neither the actual labels nor the precise order on the labels have to be recorded but only how the labels in the join relate to the labels from the peak. We use the following encoding. Let the local peak have labels $t\ {}_{\alpha}\!\leftarrow s \to_{\beta} u$. Then a step $v \to_{\gamma} w$ is labeled by the pair $(\circ_{\alpha}, \circ_{\beta})$ where $\circ_{\alpha}$ and $\circ_{\beta}$ indicates if $\alpha \circ_{\alpha} \gamma$ and $\beta \circ_{\beta} \gamma$, respectively. Here $\{\circ_{\alpha}, \circ_{\beta}\} \subseteq \{>, \geqslant, ?\}$ and ? means that the labels are incomparable, e.g., $\mathsf{f}(x) ? \mathsf{g}(y)$ in LPO or $2x + 1 ? x + 2$ for (matrix) interpretations.[6] Decreasingness as depicted in Figure 3.8(a) can then be captured by the conditions shown in Figure 3.8(b), where $\circ$ can be replaced by any symbol.

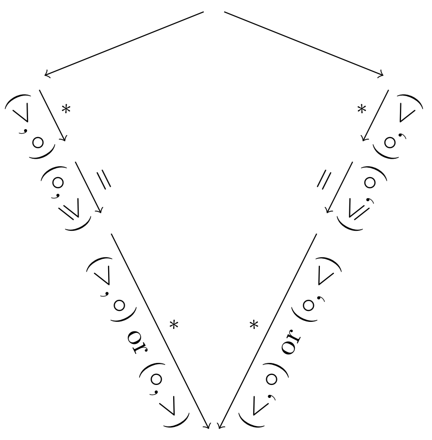
It is straightforward to implement Corollary 3.14. After establishing termination of $\mathcal{R}_{\mathsf{d}}/\mathcal{R}_{\mathsf{nd}}$ (e.g., by an external termination prover) any weak LL-labeling can be tried to show the critical peaks decreasing. In [4, 71] it is shown how the rule labeling can be implemented by encoding the constraints in linear arithmetic. Note that when using weak LL-labelings the implementation does not have to test condition 2 in Definition 3.10 since this property is intrinsic to weak LL-labelings.

We sketch how to implement the labeling $\ell^{\mathcal{S}}_{\mathrm{rt}}$ from Lemma 3.8 as a relative termination problem. First we fix a suitable set $\mathcal{S}$, i.e., the critical diagram steps

[6]Our previous implementation (reported in [201]) had a bug, as it did not track incomparable labels properly.



(a) Decreasingness.



(b) Encoding of decreasingness.

Figure 3.8.: Encoding the order on the labels.

(see Section 3.5). Facing the relative termination problem $\mathcal{S}/\mathcal{R}$ we try to simplify it according to Theorem 3.2 into some $\mathcal{S}'/\mathcal{R}'$. Note that it is not necessary to finish the proof. By Theorem 3.2 the relative TRS $(\mathcal{S} \setminus \mathcal{S}')/\mathcal{R}$ is terminating and hence by Lemma 3.8 $\ell_{\text{rt}}^{\mathcal{S}\setminus\mathcal{S}'}$ is an L-labeling. Let $\geqslant \; = \; \to_{\mathcal{R}}^{*}$ and $> \; = \; \to_{(\mathcal{S}\setminus\mathcal{S}')/\mathcal{R}}^{+}$. Since $\geqslant$ and $>$ can never increase by rewriting, it suffices to exploit the first decrease with respect to $>$. Consider a rewrite sequence $v_1 \to_{\mathcal{R}} v_2 \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} v_l$. Take the smallest k such that $v_1 \to v_{k+1} \in \mathcal{S}$ but $v_1 \to v_{k+1} \notin \mathcal{S}'$. Then $v_i \to_{(\geqslant,\geqslant)} v_{i+1}$ for $1 \leqslant i \leqslant k$ and $v_i \to_{(>,>)} v_{i+1}$ for $k < i < l$. If no such k exists set $v_i \to_{(\geqslant,\geqslant)} v_{i+1}$ for $1 \leqslant i < l$. We demonstrate the above idea on an example.

Example 3.60. Consider the following TRS $\mathcal{R}$ from [11]:

$$\mathsf{I}(x) \to \mathsf{I}(\mathsf{J}(x)) \qquad \mathsf{J}(x) \to \mathsf{J}(\mathsf{K}(\mathsf{J}(x))) \qquad \mathsf{H}(\mathsf{I}(x)) \to \mathsf{K}(\mathsf{J}(x)) \qquad \mathsf{J}(x) \to \mathsf{K}(\mathsf{J}(x))$$

We show how the critical peak $\mathsf{H}(\mathsf{I}(\mathsf{J}(x))) \leftarrow \mathsf{H}(\mathsf{I}(x)) \to \mathsf{K}(\mathsf{J}(x))$ can be closed decreasingly $\mathsf{H}(\mathsf{I}(\mathsf{J}(x))) \to_{(\geqslant,\geqslant)} \mathsf{K}(\mathsf{J}(\mathsf{J}(x))) \to_{(>,>)} \mathsf{K}(\mathsf{J}(\mathsf{K}(\mathsf{J}(x)))) \;_{(\leqslant,\leqslant)}\!\leftarrow \mathsf{K}(\mathsf{J}(x))$ by $\ell_{\text{rt}}^{\mathcal{S}}$. Let $\mathcal{S}$ be the TRS consisting of the critical diagram steps from the above diagram, i.e.,

$$\begin{array}{ll} \mathsf{H}(\mathsf{I}(x)) \to \mathsf{H}(\mathsf{I}(\mathsf{J}(x))) & \qquad \mathsf{H}(\mathsf{I}(x)) \to \mathsf{K}(\mathsf{J}(\mathsf{J}(x))) \\ \mathsf{H}(\mathsf{I}(x)) \to \mathsf{K}(\mathsf{J}(x)) & \qquad \mathsf{H}(\mathsf{I}(x)) \to \mathsf{K}(\mathsf{J}(\mathsf{K}(\mathsf{J}(x)))) \end{array}$$

The interpretation $\mathsf{H}_{\mathbb{N}}(x) = \mathsf{J}_{\mathbb{N}}(x) = \mathsf{K}_{\mathbb{N}}(x) = x$ and $\mathsf{I}_{\mathbb{N}}(x) = x + 1$ allows to "simplify" termination of the problem $\mathcal{S}/\mathcal{R}$ according to Theorem 3.2. Since the rules that reduce the number of $\mathsf{I}'s$ are dropped from $\mathcal{S}$ (and $\mathcal{R}$), those rules admit a decrease in the labeling.

The abstraction works similarly for the labelings $\ell_\star$ and $\ell_\triangle$ from Lemmata 3.20 and 3.25, respectively.

Finally, we explain why $\overset{\star}{\star}(\mathcal{R})$ need not be computed explicitly to implement Corollary 3.21 with the labeling from Lemma 3.23. The idea is to start with $\star(\mathcal{R})$ and incrementally prove termination of $\mathcal{R}^\star_>/\mathcal{R}^\star_=$ until some $\mathcal{S}_1/\mathcal{S}_2$ is reached. If all left-hand sides in $\mathcal{S}_1$ are distinct then they must have been derived from different combinations (l, x) with $l \to r \in \mathcal{R}$ and $x \in \mathcal{V}\mathsf{ar}(l)$.[7] Hence they are exactly those rules which should be placed in $\mathcal{R}^\star_=$. We show the idea by means of an example.

Example 3.61. We revisit Example 3.1 and try to prove termination of $\star(\mathcal{R})$. By an application of Theorem 3.2 with the interpretation given in Example 3.24 the problem is termination equivalent to $\mathcal{R}_\dagger/\mathcal{R}^\star_=$. By another application of Theorem 3.2 the same proof can be used to show termination of $(\mathcal{R}^\star_> \setminus \mathcal{R}^\star_\dagger)/(\mathcal{R}^\star_= \cup \mathcal{R}^\star_\dagger)$ which is a suitable candidate for $\overset{\star}{\star}(\mathcal{R})$ since the rules in $\mathcal{R}^\star_\dagger$ have different left-hand sides.

We have also implemented Theorems 3.36 and 3.38. The requirements of Theorem 3.36 can be checked effectively by characterizing $t \in \mathcal{T}_{\trianglelefteq\alpha}(\mathcal{F}, \mathcal{V})$ as follows:

Remark 3.7. *The condition $t \in \mathcal{T}_{\trianglelefteq\alpha}(\mathcal{F}, \mathcal{V})$ holds if and only if t is S-sorted and $S(t)\ (\leq \cup \lhd_1)^*\ \alpha$, where the relation $\lhd_1$ on sorts relates argument types to result types: $S(f, i) \lhd_1 S(f)$ for all function symbols $f \in \mathcal{F}$ of arity n and $1 \leqslant i \leqslant n$.*

We only implemented the simplest case of Theorem 3.38, where ℓ is a rule labeling. First, using Remark 3.7, we determine for which rules $l \to r \in \mathcal{R}$, $l' \to r' \in \mathcal{R}$, it is possible to nest $l' \to r'$ below a duplicating variable of $l \to r$. We add constraints $i(l \to r) > i(l' \to r')$ to our constraint satisfaction problem for the rule labeling. The hard work is done by an SMT solver.

To postpone the expensive computation (and labeling) of parallel critical pairs as long as possible we implemented Theorem 3.50 according the following lazy approach. We first find ordinary weak LL-labelings for the critical diagrams, as described earlier in this section. Only if confluence cannot be established by considering this weak LL-labeling for (non-parallel) critical peaks, we generate parallel critical peaks together with joining sequences. Finally, we check whether the weak LL-labeling joins all resulting diagrams (critical and parallel critical) decreasing as per Theorem 3.50. This check is also responsible for combining single steps into a parallel one for the joining sequence. We confess that this implementation for Theorem 3.50 is somewhat opportunistic but allows to reuse partial progress (the weak LL-labeling) while postponing parallel critical pairs as long as possible.

[7] When computing $\star(\mathcal{R})$ the implementation renames variables such that (ℓ, x) uniquely identifies a rule $\ell \to r$.

method	pre	CR(ℓ_{rl})	CR(ℓ_{rt})	CR
Theorem 3.6	69	42	36	44
Theorem 3.36	92	46	40	48
Theorem 3.38	92	53	–	–
Corollary 3.14	65	47	40	49
Corollary 3.21$\star$	66	48	41	50
Corollary 3.21⁑	69	51	43	53
Corollary 3.27	65	47	41	49
Theorem 3.50	92	55	55	57

Table 3.1.: Experimental results for 92 left-linear TRSs.

3.7. Experiments

The results from the article have been implemented and form the core of the confluence prover CSI [200]. For experiments[8] using version 0.4 of the tool we considered the current 276 TRSs in Cops. In the experiments we focus on the 149 systems which have been referenced from the confluence literature. From these systems 92 are left-linear. Our experiments have been performed on a notebook equipped with an Intel® quad core processor i7-2640M running at a clock rate of 2.8 GHz and 4 GB of main memory. For 3 systems not even local confluence could be established within 60 seconds. All other tests finished within this time limit.

Table 3.1 shows an evaluation of the results from this article. The first column indicates which criterion has been used to investigate confluence. A $\star$ means that the corresponding corollary is implemented using $\star(\mathcal{R})$ whereas ⁑ refers to ⁑$(\mathcal{R})$. The column labeled pre shows for how many systems the precondition of the respective criterion is satisfied, e.g., for Theorem 3.6 the precondition is linearity while for Corollary 3.14 the precondition is termination of $\mathcal{R}_{\mathsf{d}}/\mathcal{R}_{\mathsf{nd}}$. The columns labeled CR(ℓ) give the number of systems for which confluence could be established using labeling ℓ. (For Corollary 3.21 implicitly $\ell_\star$ is also employed. Similarly Corollary 3.27 employs $\ell_\triangle$.) The column labeled CR corresponds to the full power of each result, i.e., when the lexicographic combination of all labelings is used.

From the table we draw the following conclusions. On this test bed the labeling function ℓ_{rl} can handle more systems than ℓ_{rt} when considering single steps but for parallel rewriting both labelings succeed on equally many systems. Still, in both settings most power is obtained when using all labelings. In practice the study of parallel rewriting (Theorem 3.50) is beneficial. This suggests that the preconditions to obtain weak LL-labelings are severe.

[8]Details available from `http://cl-informatik.uibk.ac.at/software/csi/labeling2`.

tool	CR	not CR
ACP	63	22
CSI	67	20
saigawa	53	12
$\sum$	68	22

Table 3.2.: Comparison with other tools on 92 left-linear TRSs.

For reference in Table 3.2 we compare the power of the confluence provers participating in the Confluence Competition (CoCo),[9] i.e., ACP [11], CSI [200], and saigawa [71, 97].

- ACP is a powerful confluence prover which implements numerous confluence criteria from the literature. Its distinctive feature is the strong support for problems with AC semantics [6].
- CSI gains most of its power from the labeling framework presented here. In addition it implements development closed critical pairs [143] and persistence [47]. Recently, the techniques introduced in [6] and [97] have also been integrated.
- saigawa also heavily exploits relative termination, remarkably also to analyze confluence of non-left-linear systems [97].

From Tables 3.1 and 3.2 we conclude that our framework admits a state-of-the-art confluence prover for left-linear systems.

3.8. Conclusion

In this article we studied how the decreasing diagrams technique can be automated. We presented conditions (subsuming recent related results) that ensure confluence of a left-linear TRS whenever its critical peaks are decreasing. The labelings we proposed can be combined lexicographically which allows incremental proofs of confluence and has a modular flavor in the following sense: Whenever a new labeling function is invented, the whole framework gains power. We discussed several situations (Examples 3.1, 3.16, 3.17, 3.54) where traditional confluence techniques fail but our approach easily establishes confluence.

[9] `http://coco.nue.riec.tohoku.ac.jp`

We have also considered parallel rewriting resulting in a significantly more powerful approach. We leave the study of $\circ\!\!\!\rightarrow$ and the integration of development closed critical pairs as in [69, 141] as future work.

Recently confluence by decreasing diagrams (for abstract rewrite systems) has been formalized in the theorem prover Isabelle/HOL [197, 199]. Since the generated (incremental) labeling proofs are often impossible to check for humans it seems a natural point for future work to also formalize the labeling framework to enable automatic certification of confluence proofs. Since our setting is based on a single method (decreasing diagrams) while still powerful it offers itself as a perfect candidate for future certification efforts.

Acknowledgments

We thank the anonymous reviewers for providing many helpful and detailed comments.

4. Confluence by Decreasing Diagrams – Formalized

Publication Details

Harald Zankl, Confluence by Decreasing Diagrams – Formalized, In Proceedings of the 24th International Conference on Rewriting Techniques and Applications (RTA 2013), Leibniz International Proceedings in Informatics 21, pp. 352–367, Schloss Dagstuhl, 2013.
doi: `10.4230/LIPIcs.RTA.2013.352`

Abstract

This paper presents a formalization of decreasing diagrams in the theorem prover Isabelle. It discusses mechanical proofs showing that any locally decreasing abstract rewrite system is confluent. The valley and the conversion version of decreasing diagrams are considered.

4.1. Introduction

Formalizing confluence criteria has a long history in λ-calculus. Huet [85] proved a stronger variant of the parallel moves lemma in Coq. Isabelle/HOL was used in [132] to prove the Church-Rosser property of β, η, and $\beta\eta$. For β-reduction the standard Tait/Martin-Löf proof as well as Takahashi's proof [171] were formalized. The first mechanically verified proof of the Church-Rosser property of β-reduction was done using the Boyer-Moore theorem prover [160]. The formalization in Twelf [149] was used to formalize the confluence proof of a specific higher-order rewrite system in [168].

Newman's lemma (for abstract rewrite systems) and Knuth and Bendix' critical pair theorem (for first-order rewrite systems) have been proved in [153] using ACL. An alternative proof of the latter in PVS, following the higher-order structure of Huet's proof, is presented in [51]. PVS is also used in the formalization of the lemmas of Newman and Yokouchi in [50]. Knuth and Bendix' criterion has also been formalized in Coq [37] and Isabelle/HOL [173].

Decreasing diagrams [139] are a complete characterization of confluence for abstract rewrite systems whose convertibility classes are countable. As a criterion for abstract rewrite systems, they can easily be applied for first- and higher-order rewriting, including term rewriting and the λ-calculus. Furthermore, decreasing diagrams yield constructive proofs of confluence [144] (in the sense that the joining sequences can be computed based on the divergence). We are not aware of a (complete) formalization of decreasing diagrams in any theorem prover (see remarks in Section 4.6).

In this paper we discuss a formalization of decreasing diagrams in the theorem prover Isabelle/HOL. (In the sequel we just call it Isabelle.) We closely follow the proofs in [138, 139]. For alternative proofs see [22, 99] or [45, 90, 142] where proof orders play an essential role. The main contributions of this paper are (two) mechanical proofs of Theorem 4.1 in Isabelle.

Theorem 4.1 ([138, 139]). *Every locally decreasing abstract rewrite system is confluent.* □

As a consequence all definitions (lemmata) in this paper have been formalized (proved) in Isabelle. The definitions from the paper are (modulo notation) identical to the ones used in Isabelle. Our formalization (Decreasing_Diagrams.thy, available from [198]) consists of approximately 1600 lines of Isabelle code in the Isar style and contains 31 definitions and 122 lemmata. The valley version [139] amounts to ca. 1000 lines, 22 definitions, and 97 lemmata while the conversion version [138] has additional 600 lines of Isabelle comprising 9 definitions and 25 lemmata. Our formalization imports the theory Multiset.thy from the Isabelle library and Abstract_Rewriting.thy [163] from the Archive of Formal Proofs. We used Isabelle 2012 and the Archive of Formal Proofs from July 30, 2012.

The remainder of this paper is organized as follows. In the next section we recall helpful preliminaries for our formalization of [139], which is described in Section 4.3. The conversion version of decreasing diagrams [138] is the topic of Section 4.4. In Section 4.5 we highlight changes to (and omissions in) the proofs from [138, 139] before we conclude in Section 4.6.

4.2. Preliminaries

We assume familiarity with rewriting [172] and decreasing diagrams [139]. Basic knowledge of Isabelle [133] is not essential but may be helpful.

meaning	set	multiset	sequence/list	[139]
empty	$\{\}$	$\{\#\}$	$[\,]$	$\emptyset/\epsilon$
singleton	$\{\alpha\}$	$\{\#\alpha\#\}$	$[\alpha]$	$\{\alpha\}/[\alpha]/\alpha$
membership	$\alpha \in S$	$\alpha \in\# M$	–	$\in$
union/concatenation	$S \cup T$	$M + N$	$\sigma @ \tau$	$\uplus/\sigma\tau$
intersection	$S \cap T$	$M \#\cap N$	–	$\cap$
difference	$S - T$	$M - N$	–	$-$
sub(multi)set	$S \subseteq T$	$M \leq N$	–	$\subseteq$

Table 4.1.: Predefined Isabelle operators.

Given a relation $\to$ we write $\leftarrow$ for its inverse, $\twoheadrightarrow$ for its transitive closure, and $\to^{=}$ (in pictures also $\overset{=}{\to}$) for its reflexive closure. We write $\leftrightarrow$ for $\to$ or $\leftarrow$ and denote sets by S, T, U, multisets by M, N, I, J, K, Q, single labels by α, β, γ, and lists of labels by σ, τ, υ, κ, μ, and ρ (possibly primed or indexed).

Table 4.1 gives an overview of several predefined operators in Isabelle for sets, multisets, and lists (sequences) where we also incorporated the notation from [139] in the rightmost column. In the paper we will use the Isabelle notation, but drop the @ for concatenating sequences and write α instead of $[\alpha]$. In addition to the operators provided by Isabelle, we need the difference (intersection) of a multiset with a set. Here $M -s\ S$ ($M \cap s\ S$) removes (keeps) all occurrences of elements in M that are in S. Sometimes it will be necessary to convert e.g. a multiset to a set (or a list). In the paper we leave these conversions implicit, since no confusion can arise. We establish the following useful equivalences:

Lemma 4.2 (parts of [139, Lemma A.3])**.**

1. $(M + N) -s\ S = (M -s\ S) + (N -s\ S)$
2. $(M -s\ S) -s\ T = M -s\ (S \cup T)$
3. $M = (M \cap s\ S) + (M -s\ S)$
4. $(M -s\ T) \cap s\ S = (M \cap s\ S) -s\ T$

Proof. By unfolding the definitions of multiset and the operators. □

4.3. Formalization of Decreasing Diagrams

We assume familiarity with the original proof of decreasing diagrams in [139], upon which our formalization in this section is based. Nevertheless we will recall the important definitions and lemmata. However, we only give proofs if our proof deviates from the original argument. In addition we state (sometimes small) key results, since an effective collection of lemmata is crucial for completely formal proofs.

The remainder of this section is organized as follows: Section 4.3.1 describes our results on multisets. Section 4.3.2 is dedicated to decreasingness (of sequences of labels) and Section 4.3.3 is concerned with an alternative formulation of local decreasingness. Afterwards, Section 4.3.4 lifts decreasingness (from labels) to diagrams. Well-foundedness of the measure (on peaks) is proved in Section 4.3.5, where we also establish the main result.

4.3.1. Multisets

In the sequel we assume $\prec$ to be a transitive and irreflexive binary relation.

Definition 4.3 ([139, Definition 2.5]).

1. The set $\curlyvee\alpha$ is the strict order ideal generated by (or *down-set* of) α, defined by $\curlyvee\alpha = \{\beta \mid \beta \prec \alpha\}$. This is extended to sets $\curlyvee S = \bigcup_{\alpha \in S} \curlyvee\alpha$. We define $\curlyvee M$ and $\curlyvee\sigma$ to be the down-set generated by the set of elements in M and σ, respectively.

2. The *(standard) multiset extension* (denoted by $\prec_{\mathsf{mul}}$) of $\prec$ is defined by

$$M \prec_{\mathsf{mul}} N \text{ if } \exists\ I\ J\ K.\ M = I + K,\ N = I + J,\ K \subseteq \curlyvee J, \text{ and } J \neq \{\#\}$$

   The relation $\preccurlyeq_{\mathsf{mul}}$ is obtained by removing the last condition ($J \neq \{\#\}$). Note that $\preccurlyeq_{\mathsf{mul}}$ is the reflexive closure of $\prec_{\mathsf{mul}}$ (cf. Lemma 4.39 in Section 4.5).

The following result is not mentioned in [139]—while [140, Proposition 1.4.8(3)] shows a more general result—but turned out handy for our formalization.

Lemma 4.4. $\curlyvee(\curlyvee S) \subseteq \curlyvee S$

Proof. Assume $x \in \curlyvee(\curlyvee S)$. By Definition 4.3 there must be a $y \in \curlyvee S$ with $x \prec y$. From $y \in \curlyvee S$ we obtain a $z \in S$ with $y \prec z$. Then $x \prec z$ by transitivity of $\prec$ and hence $x \in \curlyvee S$. □

The multiset extension inherits some properties of the base relation, which we will implicitly use in the sequel.

Lemma 4.5. *Let $\prec$ be a transitive and well-founded relation. Then $\prec_{\mathsf{mul}}$ is transitive and well-founded, and $\preccurlyeq_{\mathsf{mul}}$ is reflexive and transitive.*

Proof. By Lemmata 4.38 and 4.39 in combination with existing results in Multiset.thy. □

We can now establish the following properties.

Lemma 4.6 ([139, Lemma 2.6])**.**

1. $\curlyvee(S \cup T) = \curlyvee S \cup \curlyvee T$ *and* $\curlyvee(\sigma\tau) = \curlyvee\sigma \cup \curlyvee\tau$ *and* $\curlyvee(M -s\ S) \supseteq \curlyvee M -s\ \curlyvee S$
2. $M \leq N \Rightarrow M \preccurlyeq_{\mathsf{mul}} N \Rightarrow \curlyvee M \subseteq \curlyvee N$
3. $M \preccurlyeq_{\mathsf{mul}} N \Rightarrow \exists\ I\ J\ K.\ M = I + K \wedge N = I + J \wedge K \subseteq \curlyvee J \wedge J \#\!\cap K = \{\#\}$
4. $N \neq \{\#\} \wedge M \subseteq \curlyvee N \Rightarrow M \prec_{\mathsf{mul}} N$
5. $M \preccurlyeq_{\mathsf{mul}} N \Rightarrow M -s\ \curlyvee S \preccurlyeq_{\mathsf{mul}} N -s\ \curlyvee S$
6. $M \preccurlyeq_{\mathsf{mul}} N \Leftrightarrow Q + M \preccurlyeq_{\mathsf{mul}} Q + N$
7. $Q \subseteq \curlyvee N - \curlyvee M \wedge M \preccurlyeq_{\mathsf{mul}} N \Rightarrow Q + M \preccurlyeq_{\mathsf{mul}} N$
8. $S \subseteq T \Rightarrow M -s\ T \preccurlyeq_{\mathsf{mul}} M -s\ S$
9. $M \prec_{\mathsf{mul}} N \Rightarrow Q + M \prec_{\mathsf{mul}} Q + N$

Note that statements (5) and (6) slightly differ from [139, Lemma 2.6](5,6), but are easier to apply. The (easy) statements of (8) and (9) are not mentioned in [139], which we required for [139, Lemmata 3.5 and 3.6].

4.3.2. Decreasingness

We define the *lexicographic maximum* measure, which maps lists to multisets, inductively.

Definition 4.7 ([139, Definition 3.2])**.**

- $|[\,]| = \{\#\}$
- $|\alpha\sigma| = \{\#\alpha\#\} + (|\sigma| -s\ \curlyvee\alpha)$

The next lemma establishes properties of the lexicographic maximum measure.

Lemma 4.8 ([139, Lemma 3.2])**.**

1. $\curlyvee|\sigma| = \curlyvee\sigma$
2. $\curlyvee|\sigma\tau| = |\sigma| + (|\tau| -s\ \curlyvee\sigma)$

$$\begin{tikzcd}
{} \arrow[r, two heads, "\tau"] \arrow[d, two heads, "\sigma"'] \arrow[dr, phantom, "\mathrm{D}"] & {} \arrow[d, two heads, "\sigma'"] \\
{} \arrow[r, two heads, "\tau'"'] & {}
\end{tikzcd}$$

(a) Decreasing diagram.

$$\begin{tikzcd}
{} \arrow[r, "\beta"] \arrow[d, "\alpha"'] \arrow[dr, phantom, "\mathrm{LD}"] & {} \arrow[d, two heads, "\sigma'"] \\
{} \arrow[r, two heads, "\tau'"'] & {}
\end{tikzcd}$$

(b) Locally decreasing diagram.

Figure 4.1.: Diagrams.

Proof.

1. By induction on σ. The base case is trivial. Using Lemma 4.6(1) the inductive step amounts to $\curlyvee\alpha \cup \curlyvee(|\sigma| -s \curlyvee\alpha) = \curlyvee\alpha \cup \curlyvee\sigma$. The inclusion from left to right follows from the induction hypothesis. For the inclusion from right to left we proceed by case analysis. If $x \in \curlyvee\alpha$ then the result immediately follows. If $x \notin \curlyvee\alpha$ then $x \in \curlyvee\sigma$ and from the induction hypothesis $x \in \curlyvee|\sigma|$. Furthermore $x \notin \curlyvee\alpha$ using Lemma 4.4 also yields $x \notin \curlyvee(\curlyvee\alpha)$. Hence $x \in \curlyvee|\sigma| -s \curlyvee(\curlyvee\alpha)$ and from Lemma 4.6(1) we obtain $x \in \curlyvee(|\sigma| -s \curlyvee\alpha)$, from which the result follows.

2. By induction on σ, see [139]. □

Decreasingness is defined on quadruples (of sequences of labels).

Definition 4.9 ([139, Definition 3.3] for labels)**.** The tuple of labels $(\tau, \sigma, \sigma', \tau')$ is *decreasing* (D) if $|\sigma\tau'| \preccurlyeq_{\mathsf{mul}} |\tau| + |\sigma|$ and $|\tau\sigma'| \preccurlyeq_{\mathsf{mul}} |\tau| + |\sigma|$. For a visualization see Figure 4.1a.[1]

We write D into a diagram to indicate that its labels are decreasing. Decreasingness can also be stated differently.

Lemma 4.10 ([139, Definition 3.3])**.** *The following two statements are equivalent:*

1. $|\sigma\tau'| \preccurlyeq_{\mathsf{mul}} |\tau| + |\sigma|$ *and* $|\tau\sigma'| \preccurlyeq_{\mathsf{mul}} |\tau| + |\sigma|$

2. $|\tau'| -s \curlyvee\sigma \preccurlyeq_{\mathsf{mul}} |\tau|$ *and* $|\sigma'| -s \curlyvee\tau \preccurlyeq_{\mathsf{mul}} |\sigma|$

[1]Although the results in Sections 4.3.2 and 4.3.3 are on labels only for visualization we already use diagrams.

$$\begin{tikzcd}
{} \arrow[r, two heads, "\tau"] \arrow[d, two heads, "\sigma"'] \arrow[dr, phantom, "\mathrm{D}"] & {} \arrow[r, two heads, "\upsilon"] \arrow[d, two heads, "\sigma'"] \arrow[dr, phantom, "\mathrm{D}"] & {} \arrow[d, two heads, "\sigma''"] & \Rightarrow & {} \arrow[r, two heads, "\tau\upsilon"] \arrow[d, two heads, "\sigma"'] \arrow[dr, phantom, "\mathrm{D}"] & {} \arrow[d, two heads, "\sigma''"] \\
{} \arrow[r, two heads, "\tau'"'] & {} \arrow[r, two heads, "\upsilon'"'] & {} & & {} \arrow[r, two heads, "\tau'\upsilon'"'] & {}
\end{tikzcd}$$

(a) Lemma 4.11.

$$\begin{tikzcd}
{} \arrow[r, two heads, "\tau"] \arrow[d, two heads, "\sigma"'] \arrow[dr, phantom, "\mathrm{D}"] & {} \arrow[r, two heads, "\upsilon"] \arrow[d, two heads, "\sigma'"] & {} \\
{} \arrow[r, two heads, "\tau'"'] & {} &
\end{tikzcd}$$

(b) Lemma 4.12.

Figure 4.2.: Pasting preserves decreasingness and is hypothesis decreasing.

Proof. By Lemma 4.8(2) and Lemma 4.6(6). □

We have followed the (involved) proofs in [139] that pasting preserves decreasingness (Lemma 4.11) and that pasting is hypothesis decreasing (Lemma 4.12) without big changes.

Lemma 4.11 ([139, Lemma 3.5] for labels)**.** *If $(\tau, \sigma, \sigma', \tau')$ and $(\upsilon, \sigma', \sigma'', \upsilon')$ are decreasing, then $(\tau\upsilon, \sigma, \sigma'', \tau'\upsilon')$ is decreasing (see Figure 4.2a).*

Proof. As in [139] but we show $(|\upsilon'| -s \curlyvee\sigma\tau') -s \curlyvee\tau \preccurlyeq_{\mathsf{mul}} (|\upsilon'| -s \curlyvee\sigma') -s \curlyvee\tau$ (instead of $\subseteq$) where we needed Lemma 4.6(8) (in the last sequence in [139, Proof of Lemma 3.5]). □

Lemma 4.12 ([139, Lemma 3.6] for labels)**.** *If τ is non-empty and we have that $(\tau, \sigma, \sigma', \tau')$ is decreasing (see Figure 4.2b) then $|\sigma'| + |\upsilon| \prec_{mul} |\sigma| + |\tau\upsilon|$.*

Proof. As in [139] using Lemma 4.6(9) in the second step. □

4.3.3. Local Decreasingness

Labels $(\beta, \alpha, \sigma', \tau')$ are *locally decreasing* (LD) if they are decreasing and both α and β consist of exactly one label (see Figure 4.1b). Now, LD can also be formulated differently:

Lemma 4.13 ([139, Prop. 3.4])**.** *The form of locally decreasing labels is specified in Figure 4.3a.*

To show Lemma 4.13 we give names to the joining sequences as in Figure 4.3b. Then the condition of Figure 4.3a can be expressed as:[2]

$$\begin{aligned} \mathrm{LD}' := {} & \sigma_1 \subseteq \curlyvee\beta \wedge \mathsf{length}\ \sigma_2 \leq 1 \wedge \sigma_2 \subseteq \{\alpha\} \wedge \sigma_3 \subseteq \curlyvee\alpha\beta \wedge \\ & \tau_1 \subseteq \curlyvee\alpha \wedge \mathsf{length}\ \tau_2 \leq 1 \wedge \tau_2 \subseteq \{\beta\} \wedge \tau_3 \subseteq \curlyvee\alpha\beta \end{aligned}$$

[2]Here length computes the length of a list.

$$\begin{tikzcd}
\bullet \arrow[rrr, "\beta"] \arrow[ddd, "\alpha"'] & & & \bullet \arrow[d, twoheadrightarrow, "\Upsilon\beta"] \\
& \mathrm{LD} & & \bullet \arrow[d, "\alpha", "="'] \\
& & & \bullet \arrow[d, twoheadrightarrow, "\Upsilon\alpha\beta"] \\
\bullet \arrow[r, twoheadrightarrow, "\Upsilon\alpha"'] & \bullet \arrow[r, "\beta"', "="] & \bullet \arrow[r, twoheadrightarrow, "\Upsilon\alpha\beta"'] & \bullet
\end{tikzcd}$$

(a) Alternative formulation of local decreasingness.

$$\begin{tikzcd}
\bullet \arrow[rrr, "\beta"] \arrow[ddd, "\alpha"'] & & & \bullet \arrow[d, twoheadrightarrow, "\sigma_1"] \\
& & & \bullet \arrow[d, "\sigma_2", "="'] \\
& & & \bullet \arrow[d, twoheadrightarrow, "\sigma_3"] \\
\bullet \arrow[r, twoheadrightarrow, "\tau_1"'] & \bullet \arrow[r, "\tau_2"', "="] & \bullet \arrow[r, twoheadrightarrow, "\tau_3"'] & \bullet
\end{tikzcd}$$

(b) Giving names to the joining sequences.

Figure 4.3.: Local diagrams.

Local decreasingness of the labels in the diagram of Figure 4.3a yields (using Lemma 4.10)

$$\mathrm{LD} := |\sigma'| -s \Upsilon\beta \preccurlyeq_{\mathsf{mul}} |\alpha| \wedge |\tau'| -s \Upsilon\alpha \preccurlyeq_{\mathsf{mul}} |\beta|$$

Hence Lemma 4.13 states that LD$'$ if and only if LD. This means that

(i) if a local diagram satisfies the conditions in Figure 4.3a, i.e. LD$'$, then it is decreasing and

(ii) local decreasingness implies that the joining sequences τ' and σ' in Figure 4.1b can be decomposed into $\tau_1\tau_2\tau_3$ and $\sigma_1\sigma_2\sigma_3$ such that the properties of the local diagram in Figure 4.3a, i.e. LD$'$, are satisfied.

Lemma 4.15 will be the key result for (i), but first we establish a useful lemma.

Lemma 4.14. $|\sigma| \leq \sigma$

Proof. By induction on σ. The base case is trivial. The step case amounts to

$$|\alpha\sigma| \;=\; \{\#\alpha\#\} + (|\sigma| -s \Upsilon\alpha) \;\leq\; \{\#\alpha\#\} + (\sigma -s \Upsilon\alpha) \;\leq\; \alpha\sigma$$

using Definition 4.7 in the first step and the induction hypothesis in the second step. □

In the sequel we will view $|\sigma|$ and σ as sets and use $|\sigma| \subseteq \sigma$. Now we can prove the following key result to establish (i).

Lemma 4.15. $\sigma_1 \subseteq \Upsilon\beta \wedge \textsf{\textit{length}}\ \sigma_2 \leq 1 \wedge \sigma_2 \subseteq \{\alpha\} \wedge \sigma_3 \subseteq \Upsilon\alpha\beta \Rightarrow |\sigma_1\sigma_2\sigma_3| -s \Upsilon\beta \preccurlyeq_{\mathsf{mul}} |\alpha|$

Proof. We show the property ($\star$), i.e., $(|\sigma_1| -s \curlyvee\beta) + ((|\sigma_2| -s \curlyvee\sigma_1) -s \curlyvee\beta) + (((|\sigma_3| -s \curlyvee\sigma_2) -s \curlyvee\sigma_1) -s \curlyvee\beta) \preccurlyeq_{\mathsf{mul}} \{\#\alpha\#\}$ which is equivalent to the conclusion by Lemmata 4.8(2), 4.2(1) and Definition 4.7. The hypothesis contains $\sigma_1 \subseteq \curlyvee\beta$, which together with Lemma 4.14 yields $|\sigma_1| \subseteq \curlyvee\beta$ and hence

$$|\sigma_1| -s \curlyvee\beta = \{\#\} \tag{1}$$

Similarly from $\sigma_3 \subseteq \curlyvee\alpha\beta$ we get $|\sigma_3| -s (\curlyvee\alpha \cup \curlyvee\beta) = \{\#\}$ and hence

$$|\sigma_3| -s (\curlyvee\sigma_2 \cup \curlyvee\sigma_1 \cup \curlyvee\alpha \cup \curlyvee\beta) = \{\#\} \tag{3}$$

Using $\mathsf{length}\ \sigma_2 \leq 1 \wedge \sigma_2 \subseteq \{\alpha\}$ from the hypothesis we consider two cases for σ_2.

- If $\sigma_2 = [\,]$ then

  $$(|\sigma_2| -s \curlyvee\sigma_1) -s \curlyvee\beta = \{\#\} \tag{2}$$

  and from (3) we have

  $$((|\sigma_3| -s \curlyvee\sigma_2) -s \curlyvee\sigma_1) -s \curlyvee\beta \preccurlyeq_{\mathsf{mul}} \{\#\alpha\#\} \tag{3'}$$

  using Lemma 4.2(2). Then ($\star$) follows immediately from (1), (2), and (3').

- If $\sigma_2 = [\alpha]$ then we get (2')

  $$\begin{aligned}(|\sigma_2| -s \curlyvee\sigma_1) -s \curlyvee\beta &= |\sigma_2| -s (\curlyvee\sigma_1 \cup \curlyvee\beta) && \text{Lemma 4.2(2)}\\ &= \{\#\alpha\#\} -s (\curlyvee\sigma_1 \cup \curlyvee\beta) && \sigma_2 = [\alpha] \text{ with Definition 4.7}\\ &\preccurlyeq_{\mathsf{mul}} \{\#\alpha\#\} && \text{Lemma 4.6(8)}\end{aligned}$$

  and (because $\curlyvee\sigma_2 = \curlyvee\alpha$), similar as in the other case from (3) we get

  $$((|\sigma_3| -s \curlyvee\sigma_2) -s \curlyvee\sigma_1) -s \curlyvee\beta = \{\#\} \tag{3''}$$

  From (1), (2'), and (3'') we conclude ($\star$). □

Next we prepare for the key lemma to establish (ii), i.e., Lemma 4.17, after establishing useful intermediate results.

Lemma 4.16.

1. $\alpha \in\# |\sigma| \Rightarrow \exists\sigma_1\sigma_3.\ \sigma = \sigma_1\alpha\sigma_3 \wedge \alpha \notin \curlyvee\sigma_1$
2. $|\sigma| \subseteq \curlyvee S \Rightarrow \sigma \subseteq \curlyvee S$
3. $S \subseteq \curlyvee T \Rightarrow \curlyvee S \subseteq \curlyvee T$

Proof.

1. By induction on σ. The base case is trivial. In the step case we can assume that $\alpha \in\# |\beta\sigma|$. We proceed by case analysis.
   - If $\alpha = \beta$ then we are done with $\sigma_1 = [\,]$ and $\sigma_3 = \sigma$.
   - In the other case we have $\alpha \in\# |\sigma|$ and $\alpha \notin \curlyvee\beta$ from Definition 4.7. The induction hypothesis yields σ_1' and σ_3' with $\sigma = \sigma_1'\alpha\sigma_3'$ such that $\alpha \notin \curlyvee\sigma_1'$. Because $\alpha \notin \curlyvee\beta$ we can conclude with $\sigma_1 = \beta\sigma_1'$ and $\sigma_3 = \sigma_3'$ using Lemma 4.6(1).
2. Assume $\alpha \in \sigma$. If $\alpha \in\# |\sigma|$ then we are done by the hypothesis. In the other case there must be a $\beta \in |\sigma|$ (easy induction on σ) with $\alpha \prec \beta$. From the hypothesis we get that $\beta \in \curlyvee S$ and by transitivity also $\alpha \in \curlyvee S$, which finishes the proof.
3. By monotonicity of $\curlyvee$ ([140, Proposition 1.4.8(2)]) the assumption yields $\curlyvee S \subseteq \curlyvee(\curlyvee T)$. Lemma 4.4 finishes the proof. □

Note that Lemma 4.16(2) can be seen as an inverse of Lemma 4.14. With Lemma 4.16 we can now prove the following key result to establish (ii):

Lemma 4.17. $|\sigma'| -s\ \curlyvee\beta \preccurlyeq_{\mathsf{mul}} \{\#\alpha\#\} \Rightarrow \exists\sigma_1\sigma_2\sigma_3.\ \sigma' = \sigma_1\sigma_2\sigma_3 \wedge \sigma_1 \subseteq \curlyvee\beta \wedge$ *length* $\sigma_2 \leq 1 \wedge \sigma_2 \subseteq \{\alpha\} \wedge \sigma_3 \subseteq \curlyvee\alpha\beta$

Proof. To show the result we perform a case analysis.

- If $\alpha \in\# |\sigma'| -s\ \curlyvee\beta$ then Lemma 4.16(1) yields σ_1 and σ_3 with $\sigma' = \sigma_1\alpha\sigma_3$ and $\alpha \notin \curlyvee\sigma_1$. Hence from the hypothesis and Lemma 4.8(2) we get

  $$(|\sigma_1| -s\ \curlyvee\beta) + \{\#\alpha\#\} + (((|\sigma_3| -s\ \curlyvee\alpha) -s\ \curlyvee\sigma_1) -s\ \curlyvee\beta) \preccurlyeq_{\mathsf{mul}} \{\#\alpha\#\}$$

  and since $\alpha \notin \curlyvee\sigma_1$ and $\alpha \notin \curlyvee\beta$ it follows that

  $$|\sigma_1| -s\ \curlyvee\beta = \{\#\} \text{ and } ((|\sigma_3| -s\ \curlyvee\alpha) -s\ \curlyvee\sigma_1) -s\ \curlyvee\beta = \{\#\}$$

  Now, Lemma 4.2(2) yields

  $$|\sigma_1| \subseteq \curlyvee\beta \text{ and } |\sigma_3| \subseteq \curlyvee\alpha \cup \curlyvee\sigma_1 \cup \curlyvee\beta$$

  and from Lemma 4.16(2) we get

  $$\sigma_1 \subseteq \curlyvee\beta \text{ and } \sigma_3 \subseteq \curlyvee\alpha \cup \curlyvee\sigma_1 \cup \curlyvee\beta$$

  The latter simplifies to $\sigma_3 \subseteq \curlyvee\alpha\beta$ using $\curlyvee\sigma_1 \subseteq \curlyvee\beta$ (from Lemma 4.16(3)) and Lemma 4.6(1). Hence in this case the result follows with $\sigma_2 = [\alpha]$.

- If $\alpha \notin\# |\sigma'| -s \Upsilon\beta$

$$\begin{aligned} &\Rightarrow |\sigma'| -s \Upsilon\beta \subseteq \Upsilon\alpha && \text{hypothesis} \\ &\Rightarrow |\sigma'| \subseteq \Upsilon\alpha\beta && \text{Lemma 4.6(1)} \\ &\Rightarrow \sigma' \subseteq \Upsilon\alpha\beta && \text{Lemma 4.16(2)} \end{aligned}$$

In this case the result follows with empty σ_1, empty σ_2, and $\sigma' = \sigma_3$. □

Now Lemma 4.13 follows from Lemmata 4.15 and Lemma 4.17, which establish LD′ ⇒ LD and LD ⇒ LD′, respectively.

4.3.4. Labeled Rewriting

So far we have only considered sequences of labels. However, for the main result (Section 4.3.5) we need labeled rewriting. Hence this section sketches how we formalized *labeled* (abstract) rewriting before lifting the results from Section 4.3.2 from labels to labeled rewriting (a step which is left implicit in [139]). In the theory Abstract_Rewriting.thy an *abstract rewrite system* (ARS) is a set of pairs of objects of the same type, i.e., a binary relation. Confluence is also defined in Abstract_Rewriting.thy, but the theory does not provide support for *labeled* abstract rewrite systems. In the sequel we write $\mathcal{A}$ ($\mathcal{B}$) for (labeled) ARSs. A *labeled ARS* $\mathcal{B}$ is a ternary relation. We call $(a, \alpha, b) \in \mathcal{B}$ a *(labeled rewrite) step* and write $a \xrightarrow{\alpha} b$. Next we define *(labeled rewrite) sequences* inductively, i.e., for each object a there is the empty sequence $a \overset{[]}{\twoheadrightarrow} a$ and if $a \xrightarrow{\alpha} b$ is a step and $b \overset{\sigma}{\twoheadrightarrow} c$ is a sequence then $a \overset{\alpha\sigma}{\twoheadrightarrow} c$ is a sequence.

Example 4.18. Let $\mathcal{B}$ be the labeled ARS $\{(a, \alpha, b), (b, \beta, c)\}$. Then $a \xrightarrow{\alpha} b \xrightarrow{\beta} c$ (or $a \overset{\alpha\beta}{\twoheadrightarrow} c$) is a sequence in $\mathcal{B}$. The empty sequence $a \overset{[]}{\twoheadrightarrow} a$ we also write as a.

We prove useful properties for sequences, i.e., that chopping off a segment of a sequence again yields a sequence and that two sequences can be concatenated (provided the last element of the first sequence coincides with the first element of the second sequence).

Lemma 4.19. *Let $a_1 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{n-1}} a_n$ and $b_1 \xrightarrow{\beta_1} \cdots \xrightarrow{\beta_{m-1}} b_m$ be sequences.*

1. *Then $a_1 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{i-1}} a_i$ and $a_i \xrightarrow{\alpha_i} \cdots \xrightarrow{\alpha_{n-1}} a_n$ are sequences for any $1 \leqslant i \leqslant n$.*
2. *If $a_n = b_1$ then $a_1 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{n-1}} a_n = b_1 \xrightarrow{\beta_1} \cdots \xrightarrow{\beta_{m-1}} b_m$ is a sequence.*

Proof. By induction on $a_1 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{n-1}} a_n$. □

As a next step we introduce diagrams.

Definition 4.20. A *diagram* is a quadruple of sequences $(\overset{\tau}{\twoheadrightarrow}, \overset{\sigma}{\twoheadrightarrow}, \overset{\sigma'}{\twoheadrightarrow}, \overset{\tau'}{\twoheadrightarrow})$ such that the start and endpoints of the sequences satisfy the picture in Figure 4.1a. A diagram is called *decreasing* if its labels are.

We lift Lemma 4.11 from labels to diagrams.

Lemma 4.21 ([139, Lemma 3.5] for decreasing diagrams)**.** *Pasting two decreasing diagrams yields a decreasing diagram. For a picture see Figure 4.2a.*

Proof. With the help of Lemma 4.19(2) we show that pasting two diagrams again yields a diagram. That pasting preserves decreasingness follows from Lemma 4.11. □

4.3.5. Main Result

We establish that if all local peaks of a labeled ARS $\mathcal{B}$ are decreasing then all peaks of $\mathcal{B}$ are decreasing, following the structure of the proof of [139, Theorem 3.7]. (Changes are discussed in Section 4.5). Note that only here we need that $\prec$ is well-founded, from which irreflexivity immediately follows (to satisfy our global assumption from Section 4.2). First we introduce (local) peaks.

Definition 4.22. A *peak* $(\overset{\tau}{\twoheadrightarrow}, \overset{\sigma}{\twoheadrightarrow})$ is a pair of labeled rewrite sequences which originate from the same object. A *local peak* is a peak where the sequences consist of a single step.

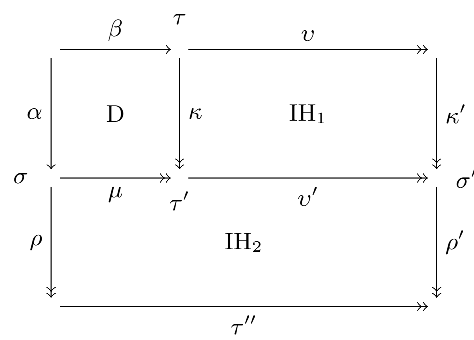
To prove the main result we introduce a measure on *peaks* (actually on pairs of sequences).

Definition 4.23. Let $|(\overset{\tau}{\twoheadrightarrow}, \overset{\sigma}{\twoheadrightarrow})| := |\tau| + |\sigma|$. Then we can lift $\prec$ as a relation on labels to a relation on pairs of sequences $\prec_{\mathsf{peak}}$, i.e., we set $(\overset{\tau}{\twoheadrightarrow}, \overset{\sigma}{\twoheadrightarrow}) \prec_{\mathsf{peak}} (\overset{\tau'}{\twoheadrightarrow}, \overset{\sigma'}{\twoheadrightarrow})$ whenever $|(\overset{\tau}{\twoheadrightarrow}, \overset{\sigma}{\twoheadrightarrow})| \prec_{\mathsf{mul}} |(\overset{\tau'}{\twoheadrightarrow}, \overset{\sigma'}{\twoheadrightarrow})|$.

For proofs of induction we establish that $\prec_{\mathsf{peak}}$ is well-founded.

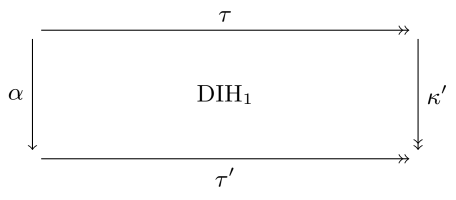
Lemma 4.24. *Let $\prec$ be well-founded. Then $\prec_{\mathsf{peak}}$ is well-founded.*

Proof. From [42] we get that $\prec_{\mathsf{mul}}$ is well-founded (this proof is contained in Multiset.thy). We proceed by contraposition. Assume the measure on peaks is not well-founded. Then we obtain an infinite sequence $\cdots \prec_{\mathsf{peak}} (\tau_2, \sigma_2) \prec_{\mathsf{peak}} (\tau_1, \sigma_1)$ which entails an infinite sequence on multisets $\cdots \prec_{\mathsf{mul}} |\tau_2| + |\sigma_2| \prec_{\mathsf{mul}} |\tau_1| + |\sigma_1|$ showing the result. □



$$\begin{tikzcd}
{} \arrow[r, "\beta"] \arrow[r, phantom, "\tau", near end, shift left=3] \arrow[d, "\alpha"'] \arrow[dr, phantom, "\mathrm{D}"] & {} \arrow[r, two heads, "\upsilon"] \arrow[d, two heads, "\kappa"] \arrow[dr, phantom, "\mathrm{IH}_1"] & {} \arrow[d, two heads, "\kappa'"] \\
\sigma \arrow[r, two heads, "\mu"'] \arrow[d, two heads, "\rho"'] \arrow[drr, phantom, "\mathrm{IH}_2"] & \tau' \arrow[r, two heads, "\upsilon'"'] & \sigma' \arrow[d, two heads, "\rho'"] \\
{} \arrow[rr, two heads, "\tau''"'] & & {}
\end{tikzcd}$$

(a) Local decreasingness implies decreasingness.



$$\begin{tikzcd}
{} \arrow[r, two heads, "\tau"] \arrow[d, "\alpha"'] \arrow[dr, phantom, "\mathrm{DIH}_1"] & {} \arrow[d, two heads, "\kappa'"] \\
{} \arrow[r, two heads, "\tau'"'] & {}
\end{tikzcd}$$

(b) Pasting D and IH_1 into DIH_1.

Figure 4.4.: Lemma 4.26.

Definition 4.25. A peak $(\xrightarrow{\tau}\!\!\!\rightarrow, \xrightarrow{\sigma}\!\!\!\rightarrow)$ in a labeled ARS is *decreasing* if it can be completed into a decreasing diagram, i.e., there are $\xrightarrow{\sigma'}\!\!\!\rightarrow$ and $\xrightarrow{\tau'}\!\!\!\rightarrow$ such that the conditions of Figure 4.1a are satisfied. A peak is *locally decreasing*, if it is decreasing and a local peak.

Lemma 4.26 (similar to [139, Theorem 3.7])**.** *Let $\mathcal{B}$ be a labeled ARS and $\prec$ be a transitive and well-founded relation on the labels. If all local peaks of $\mathcal{B}$ are decreasing, then all peaks of $\mathcal{B}$ are decreasing.*

Proof. To show that all peaks are decreasing we fix a peak $(\xrightarrow{\tau}\!\!\!\rightarrow, \xrightarrow{\sigma}\!\!\!\rightarrow)$ and show that this peak can be completed into a decreasing diagram. The proof is by well-founded induction on $\prec_{\mathsf{peak}}$ and there only is the step case. The interesting situation is when neither τ nor σ are empty, i.e., (using Lemma 4.19(1) we obtain) $\xrightarrow{\tau}\!\!\!\rightarrow = \xrightarrow{\beta} \cdot \xrightarrow{\upsilon}\!\!\!\rightarrow$ and $\xrightarrow{\sigma}\!\!\!\rightarrow = \xrightarrow{\alpha} \cdot \xrightarrow{\rho}\!\!\!\rightarrow$ (see Figure 4.4a). Hence $(\xrightarrow{\beta}, \xrightarrow{\alpha})$ is a local peak and from the assumption we obtain a decreasing diagram with joining sequences $\xrightarrow{\kappa}\!\!\!\rightarrow$ and $\xrightarrow{\mu}\!\!\!\rightarrow$. We obtain that $(\xrightarrow{\upsilon}\!\!\!\rightarrow, \xrightarrow{\kappa}\!\!\!\rightarrow)$ is a peak and want to show that the measure of this peak is smaller than that of $(\xrightarrow{\tau}\!\!\!\rightarrow, \xrightarrow{\sigma}\!\!\!\rightarrow)$ (to apply the induction hypothesis). Since β is not empty with Lemma 4.12 we establish that $|(\xrightarrow{\upsilon}\!\!\!\rightarrow, \xrightarrow{\kappa}\!\!\!\rightarrow)|$ is smaller than $|(\xrightarrow{\tau}\!\!\!\rightarrow, \xrightarrow{\alpha})|$ and from $|\alpha| \preccurlyeq_{\mathsf{mul}} |\sigma|$[3] we obtain the desired result. Now, the induction hypothesis yields that IH_1 is a decreasing diagram. Concatenating (using Lemma 4.19(2)) $\xrightarrow{\mu}\!\!\!\rightarrow$ and $\xrightarrow{\upsilon'}\!\!\!\rightarrow$ into a sequence $\xrightarrow{\tau'}\!\!\!\rightarrow$, using Lemma 4.21 we can paste the diagrams D and IH_1 into a decreasing diagram (DIH_1, see Figure 4.4b). The peak $(\xrightarrow{\tau'}\!\!\!\rightarrow, \xrightarrow{\rho}\!\!\!\rightarrow)$ is

[3]This step is not mentioned in [139, 140] but hinted at in [138].

smaller than the peak $(\overset{\tau}{\twoheadrightarrow}, \overset{\sigma}{\twoheadrightarrow})$ by a mirrored version of Lemma 4.12 and hence the induction hypothesis yields the decreasing diagram IH_2. Finally, a mirrored version of Lemma 4.21 pastes DIH_1 and IH_2 into a decreasing diagram. $\square$

We define local decreasingness for ARSs.

Definition 4.27 ([139, Definition 3.8])**.** An ARS $\mathcal{A}$ is *locally decreasing* if there exists a transitive and well-founded relation $\prec$ on the labels such that all local peaks are decreasing for (a labeled version of) $\mathcal{A}$.

Finally we arrive at the main result for soundness:

Corollary 4.28 ([139, Corollary 3.9])**.** *A locally decreasing ARS is confluent.*

Proof. From local decreasingness we get a transitive and well-founded relation $\prec$ such that all local peaks are decreasing in a labeled version of the ARS. Lemma 4.26 yields that all peaks are decreasing. The result follows by dropping labels from the labeled rewrite sequences. $\square$

4.4. Formalization of the Conversion Version

In this section we give a formal proof for the main result underlying that local decreasingness with respect to conversions (see [138]) implies confluence. To this end we formally introduce (labeled) conversions, similarly to labeled rewrite sequences. For each object a there is the empty conversion $a \overset{[]}{\twoheadleftarrow\!\!\twoheadrightarrow} a$ (also just written a) and if $a \overset{\alpha}{\to} b$ ($a \overset{\alpha}{\leftarrow} b$) is a labeled rewrite step and $b \overset{\sigma}{\twoheadleftarrow\!\!\twoheadrightarrow} c$ is a conversion then $a \overset{\alpha}{\to} b \overset{\sigma}{\twoheadleftarrow\!\!\twoheadrightarrow} c$ ($a \overset{\alpha}{\leftarrow} b \overset{\sigma}{\twoheadleftarrow\!\!\twoheadrightarrow} c$) is a conversion (often written $a \overset{\alpha\sigma}{\twoheadleftarrow\!\!\twoheadrightarrow} c$). For conversions we prove similar properties as for sequences (see Lemma 4.19). In addition we establish that mirroring a conversion again yields a conversion (with the same set of labels) and that every sequence is a conversion.

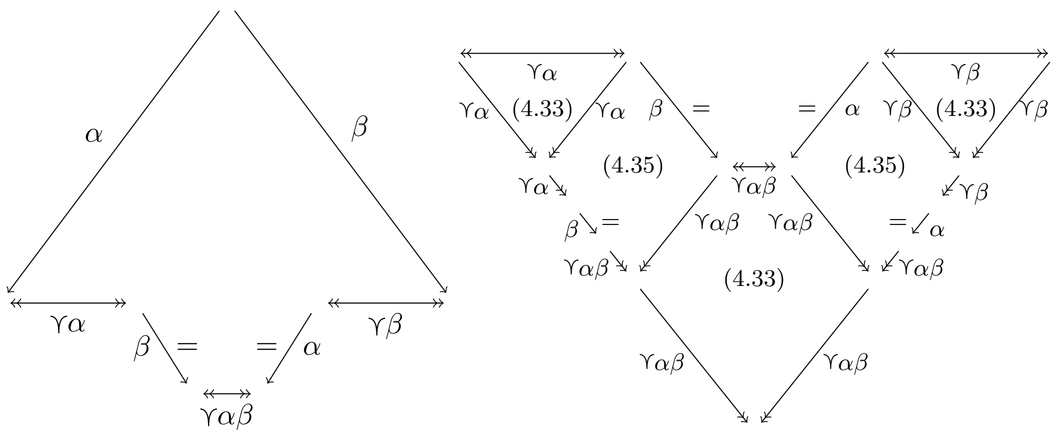
Lemma 4.29. *Let $a_1 \overset{\alpha_1}{\leftrightarrow} \cdots \overset{\alpha_{n-1}}{\leftrightarrow} a_n$ and $b_1 \overset{\beta_1}{\leftrightarrow} \cdots \overset{\beta_{m-1}}{\leftrightarrow} b_m$ be conversions.*

1. *Then $a_1 \overset{\alpha_1}{\leftrightarrow} \cdots \overset{\alpha_{i-1}}{\leftrightarrow} a_i$ and $a_i \overset{\alpha_i}{\leftrightarrow} \cdots \overset{\alpha_{n-1}}{\leftrightarrow} a_n$ are conversions for any $1 \leqslant i \leqslant n$.*

2. *If $a_n = b_1$ then $a_1 \overset{\alpha_1}{\leftrightarrow} \cdots \overset{\alpha_{n-1}}{\leftrightarrow} a_n = b_1 \overset{\beta_1}{\leftrightarrow} \cdots \overset{\beta_{m-1}}{\leftrightarrow} b_m$ is a conversion.*

3. *Then $a_n \overset{\alpha_{n-1}}{\leftrightarrow} \cdots \overset{\alpha_1}{\leftrightarrow} a_1$ is a conversion and $\{\alpha_1, \ldots, \alpha_n\} = \{\alpha_n, \ldots, \alpha_1\}$.*

4. *If $c_1 \overset{\gamma_1}{\to} \cdots \overset{\gamma_{n-1}}{\to} c_n$ is a sequence then $c_1 \overset{\gamma_1}{\leftrightarrow} \cdots \overset{\gamma_{n-1}}{\leftrightarrow} c_n$ is a conversion.*



(a) Local decreasingness wrt. conversions. (b) Closing the conversion into a valley.

Figure 4.5.: Conversion version of decreasing diagrams.

Proof. Items (1–3) are proved by induction on the first conversion, item (4) is proved by induction on the sequence. ☐

We will also use the following easy lemma being a direct consequence of Definition 4.3.

Lemma 4.30. *If $M \preccurlyeq_{\mathsf{mul}} N$ and $N \subseteq \curlyvee S$ then $M \subseteq \curlyvee S$.* ☐

The following result (stated as observation in [138]) follows from Lemma 4.30.

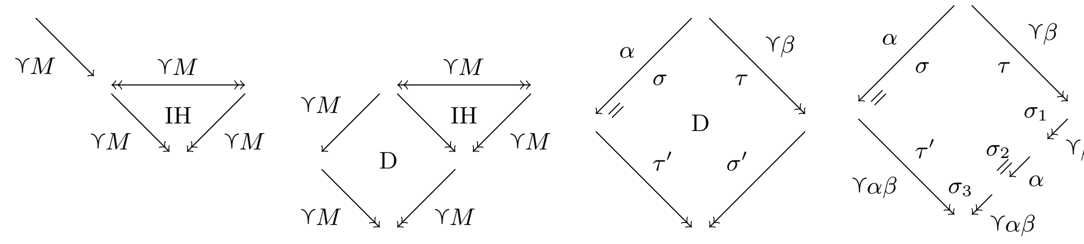
Lemma 4.31. *If $(\xrightarrow{\tau}\!\!\!\!\rightarrow, \xrightarrow{\sigma}\!\!\!\!\rightarrow, \xrightarrow{\sigma'}\!\!\!\!\rightarrow, \xrightarrow{\tau'}\!\!\!\!\rightarrow)$ is a decreasing diagram and $|(\xrightarrow{\tau}\!\!\!\!\rightarrow, \xrightarrow{\sigma}\!\!\!\!\rightarrow)| \subseteq \curlyvee M$ then also $|(\xrightarrow{\sigma'}\!\!\!\!\rightarrow, \xrightarrow{\tau'}\!\!\!\!\rightarrow)| \subseteq \curlyvee M$.* ☐

A local peak $(\xleftarrow{\beta}, \xrightarrow{\alpha})$ is *decreasing with respect to conversions*[4] if there exist conversions such that the constraints from Figure 4.5a are satisfied. Now we can state the main result underlying soundness of the conversion version of decreasing diagrams.

Lemma 4.32. *Let $\mathcal{B}$ be a labeled ARS and $\prec$ be a transitive and well-founded relation on the labels. If all local peaks of $\mathcal{B}$ are decreasing with respect to conversions, then all peaks of $\mathcal{B}$ are decreasing (with respect to valleys).*

Proof. Similar to [138] we follow the proof of the valley version (see Lemma 4.26). In contrast to Lemma 4.26 we do not get decreasingness of the local peak $(\xleftarrow{\beta}, \xrightarrow{\alpha})$

[4]Please note the asymmetry to the definition of local decreasingness (Definition 4.25).



(a) Lemma 4.33 (case $\rightarrow$). (b) Lemma 4.33 (case $\leftarrow$). (c) Lemma 4.35. (d) Lemma 4.35.

Figure 4.6.: Lemmata 4.33 and 4.35.

(in Figure 4.4a) by assumption. Instead our assumption yields local decreasingness with respect to conversions, i.e., as depicted in Figure 4.5a. We close the conversion into a valley as outlined in Figure 4.5b. To this end we use Lemmata 4.33 and 4.35 (see below) and conclude the valleys as shown in Figure 4.5b. Note that for the final application of Lemma 4.33 we apply Lemma 4.29 first, to combine the sequences and conversions into a single conversion. Lemma 4.13 (lifted to rewriting sequences) then shows decreasingness of the diagram. □

The main structure of our proof follows the one from [138]. However, there the proofs of two key results are sketchy and informal. We identified the statements as Lemmata 4.33 and 4.35 and provide formal proofs. Note that to establish these properties we can use the induction hypothesis (from the proof of Lemma 4.32), e.g., peaks whose measure is smaller than $|(\xrightarrow{\beta}, \xrightarrow{\alpha})|$ can be completed into a decreasing diagram.

Lemma 4.33. *Let all peaks smaller than $|(\xrightarrow{\beta}, \xrightarrow{\alpha})|$ have a decreasing diagram. Then for any M with $M \preccurlyeq_{\mathsf{mul}} \{\#\alpha, \beta\#\}$ we have $\overset{\curlyvee M}{\twoheadleftarrow\!\!\twoheadrightarrow} \subseteq \overset{\curlyvee M}{\twoheadrightarrow} \cdot \overset{\curlyvee M}{\twoheadleftarrow}$.*

Proof. By induction on the conversion $\overset{\curlyvee M}{\twoheadleftarrow\!\!\twoheadrightarrow}$. The base case is trivial. In the step case we have $\overset{\curlyvee M}{\leftrightarrow} \cdot \overset{\curlyvee M}{\twoheadleftarrow\!\!\twoheadrightarrow}$. The induction hypothesis yields $\overset{\curlyvee M}{\leftrightarrow} \cdot \overset{\curlyvee M}{\twoheadrightarrow} \cdot \overset{\curlyvee M}{\twoheadleftarrow}$. We consider two cases. If the first step is from left to right, i.e., $\overset{\curlyvee M}{\rightarrow}$ then the result follows from Lemma 4.29(2) (see Figure 4.6a). In the other case we have $\overset{\curlyvee M}{\leftarrow} \cdot \overset{\curlyvee M}{\twoheadrightarrow} \cdot \overset{\curlyvee M}{\twoheadleftarrow}$. Since the peak $\overset{\curlyvee M}{\leftarrow} \cdot \overset{\curlyvee M}{\twoheadrightarrow}$ has a smaller measure than $(\xrightarrow{\beta}, \xrightarrow{\alpha})$ it can be completed into a decreasing diagram and Lemma 4.31 in combination with Lemma 4.29(2) yields the result (see Figure 4.6b). □

To show the second key result we establish a useful decomposition result on sequences.

Lemma 4.34. *Let* $\overset{\sigma}{\twoheadrightarrow}$ *be a sequence and* $\sigma = \sigma_1\sigma_2$*. Then there are sequences* $\overset{\sigma_1}{\twoheadrightarrow}$ *and* $\overset{\sigma_2}{\twoheadrightarrow}$ *such that* $\overset{\sigma}{\twoheadrightarrow} = \overset{\sigma_1}{\twoheadrightarrow} \cdot \overset{\sigma_2}{\twoheadrightarrow}$*.*

Proof. By induction on the sequence $\overset{\sigma}{\twoheadrightarrow}$. □

Below $\xrightarrow{\alpha}^{=}$ stands for $\xrightarrow{\alpha}$ (one step) or $\overset{[]}{\twoheadrightarrow}$ (zero steps). Please note the similarity of the following result to the explicit characterization of local decreasingness (cf. Figure 4.3a).

Lemma 4.35. *Let all peaks smaller than* $|(\xleftarrow{\beta}, \xrightarrow{\alpha})|$ *have a decreasing diagram. Then the peak* $(\overset{\curlyvee\beta}{\twoheadleftarrow}, \xrightarrow{\alpha}^{=})$ *can be closed by* $\overset{\curlyvee\alpha\beta}{\twoheadrightarrow} \cdot \overset{\curlyvee\alpha\beta}{\twoheadleftarrow} \cdot {}^{=}\xleftarrow{\alpha} \cdot \overset{\curlyvee\beta}{\twoheadleftarrow}$ *(see Figure 4.6d).*

Proof. Since $|(\overset{\curlyvee\beta}{\twoheadleftarrow}, \xrightarrow{\alpha}^{=})|$ is smaller than $|(\xleftarrow{\beta}, \xrightarrow{\alpha})|$, it can be completed into a decreasing diagram $(\overset{\tau}{\twoheadrightarrow}, \overset{\sigma}{\twoheadrightarrow}, \overset{\sigma'}{\twoheadrightarrow}, \overset{\tau'}{\twoheadrightarrow})$ (see Figure 4.6c). First we show $\tau' \subseteq \curlyvee\alpha\beta$. From decreasingness and Lemma 4.10 we get $|\tau'| -s \curlyvee\sigma \preccurlyeq_{\mathsf{mul}} |\tau|$. The assumption $\tau \subseteq \curlyvee\beta$ and Lemma 4.14 yields $|\tau| \subseteq \curlyvee\beta$. Using Lemma 4.30 we obtain $|\tau'| -s \curlyvee\sigma \subseteq \curlyvee\beta$, i.e. $|\tau'| \subseteq \curlyvee\beta \cup \curlyvee\sigma$. The assumption $\sigma \subseteq \alpha$ yields $\curlyvee\sigma \subseteq \curlyvee\alpha$ and hence we conclude by Lemmata 4.6(1) and 4.16(2).

Next we show that $\overset{\sigma'}{\twoheadrightarrow}$ can be decomposed into $\overset{\sigma_1}{\twoheadrightarrow}$, $\xrightarrow{\sigma_2}^{=}$, and $\overset{\sigma_3}{\twoheadrightarrow}$ with $\sigma_1 \subseteq \curlyvee\beta$, $\sigma_2 \subseteq \{\alpha\}$, $\mathsf{length}\ \sigma_2 \leq 1$, and $\sigma_3 \subseteq \curlyvee\alpha\beta$. To this end we first observe that Lemma 4.17 also holds if β is not a single label but a sequence (here τ). Then from decreasingness we obtain $\sigma' = \sigma_1\sigma_2\sigma_3$, $\sigma_1 \subseteq \curlyvee\tau$, $\mathsf{length}\ \sigma_2 \leq 1$ $\sigma_2 \subseteq \{\alpha\}$, and $\sigma_3 \subseteq \curlyvee\alpha\sigma$. Lemma 4.34 lifts the decomposition of labels to a decomposition of sequences and we can conclude. □

An ARS $\mathcal{A}$ is *locally decreasing with respect to conversions* if there exists a transitive and well-founded relation $\prec$ on the labels such that all local peaks are decreasing with respect to conversions for (a labeled version of) $\mathcal{A}$. Finally we arrive at the main result for soundness:

Corollary 4.36 ([138, Theorem 3])**.** *A locally decreasing with respect to conversions ARS is confluent.* □

4.5. Meanderings

In this section we discuss differences between our formalization and (the proofs from) [138, 139].

Within Isabelle (Abstract_Rewriting.thy) an ARS is a binary relation while in [139] the ARS also contains the domain of the relation. A similar statement holds for labeled ARSs.

General multisets are used in [139], which can represent sets and finite multisets in one go wheres our formalization clearly separates the two concepts. The reason is purely practical, i.e., the Isabelle library already contains the dedicated theories Set.thy and Multiset.thy. The only (negligible) disadvantage we have experienced from this design choice is the need for multiple definitions of the down-set (for lists, sets, and multisets) and for Lemma 4.6(1). On the other hand, this saved us from formalizing *general multisets*, which we anticipate as a significant endeavour on its own. Moreover, [139] uses a different multiset extension than Multiset.thy. The latter defines the multiset extension as the transitive closure of the "one-step" multiset extension.

Definition 4.37. The *one-step multiset extension* (denoted $\prec_{\mathsf{mult1}}$) of $\prec$ is defined by

$$M \prec_{\mathsf{mult1}} N \text{ if } \exists\ a\ I\ K.\ M = I + K,\ N = I + \{\#a\#\},\ \forall\ b \in K.\ b \prec a$$

and the *multiset extension* of $\prec$ (denoted $\prec_{\mathsf{mult}}$) is the transitive closure of $\prec_{\mathsf{mult1}}$.

Based on the results in Multiset.thy and Definition 4.3(1) we have proven these two definitions equivalent for any transitive base relation.

Lemma 4.38. *If $\prec$ is transitive then $\prec_{\mathsf{mult}}$ and $\prec_{\mathsf{mul}}$ coincide.* □

Moreover we proved the claim in Definition 4.3.

Lemma 4.39. *We have that $\preccurlyeq_{\mathsf{mul}}$ is the reflexive closure of $\prec_{\mathsf{mul}}$.* □

Proof. First we show the inclusion from left to right. Let $M \preccurlyeq_{\mathsf{mul}} N$. If $J = \{\#\}$ then $M = N$ and the result follows. If $J \neq \{\#\}$ then $M \prec_{\mathsf{mul}} N$ and we are done.

For the reverse inclusion let (M, N) be in the reflexive closure of $\prec_{\mathsf{mul}}$. If $M = N$ then we finish with $I = M$, $K = J = \{\#\}$. In the other case we get suitable I, J, and K from the definition of $\prec_{\mathsf{mul}}$. □

Our formalization is first performed for sequences (of labels) and then lifted to labeled rewrite sequences (conversions), a step which is left implicit in [139]. After introducing labeled rewriting, we proved useful results in Isabelle (Lemmata 4.19 and 4.29).

In addition to the algebraic proof of Lemma 4.6(3) from [139] our formalization contains an alternative one. Our proof of Lemma 4.8(1) differs from the informal one in [139]. Also the formal proof of Lemma 4.13 differs from the sketch given for [139, Proposition 3.4], requiring auxiliary results (Lemmata 4.14 and 4.16).

There are some (tiny) differences between [139, Theorem 3.7] and Lemma 4.26. In [139] a measure on *diagrams* is used. However, since the closing/joining steps of the diagram are just obtained by the induction hypothesis the measure must be on *peaks* (which is used in [138]). Moreover, since in either case the measure is a multiset it is hard to relate arbitrary multisets to a peak. Hence we lifted the order on labels $\prec$ to peaks $\prec_{\mathsf{peak}}$ (Section 4.3.5) and used well-founded induction on this order. In the formalization of Lemma 4.26 (Footnote 3) we identified a necessary step to apply the induction hypothesis. Another aspect where our formalization deviates from [139] is that the original work uses families of labeled ARSs whereas our formalization considers a single labeled ARS only. Hence [139, Theorem 3.7] states the main result on families of ARSs whereas our Lemma 4.26 makes a statement about a single ARS.

Concerning [139] our formal proofs for the alternative formulation of local decreasingness (Lemma 4.13) differs from the one in [139, 140]. While this alternative formulation of local decreasingness was not needed to obtain the main result underlying the valley version ([139, Main Theorem 3.7], i.e., Lemma 4.26), it was (in a generalized formulation) essential for the main result underlying the conversion version ([138, Theorem 3], i.e., Lemma 4.32). Furthermore we gave formal proofs for two (informal) key observations made in the proof of [138, Theorem 3], resulting in Lemmata 4.33 and 4.35. Especially the latter has a non-trivial formal proof, since the induction hypothesis yields decreasingness (see Figure 4.6c) but not the desired decomposition of the joining sequences (see Figure 4.6d), in contrast to what the proof in [138] conveys.

4.6. Conclusion

In this paper we have described a formalization of decreasing diagrams in the theorem prover Isabelle following the original proofs from [138, 139]. In Sections 4.3.3 and 4.3.4 our formal proofs deviate from the either informal or implicit ones in [139] and we also elaborate on Lemma 4.35, a result which is implicitly used in [138]. To show the applicability of our formalization we performed a mechanical proof of Newman's lemma using decreasing diagrams (following [139, Corollary 4.4]). Our formalization has few dependencies on existing theories. From Abstract_Rewriting.thy we employ some properties for unlabeled abstract rewriting (and the definition of confluence). The theory Multiset.thy provides standard multiset operations and a well-foundedness proof of the multiset extension of a well-founded relation. Note that some of our results on multisets (a formalized proof of [139, Lemma 2.6(3)], i.e., Lemma 4.6(3)) might be of interest for a larger community.

In [26] a "point version" of decreasing diagrams is introduced, where objects are labeled instead of steps. It is unknown if the point version is equivalent to the standard one. Parts of [26] have been formalized in Coq but 29 axioms are assumed, i.e., not proven in the theorem prover. Furthermore the more useful alternative representation of local decreasingness (Lemma 4.13) is not considered in [26]. The same holds for the conversion version. Hence [26] is only a *partial* formalization and essentially different from ours.

We anticipate that our contribution paves the way for future work in several directions. One possibility is the formalization of confluence results that can be proven with decreasing diagrams (e.g. Toyama's theorem [180]). The benefit might be two-fold. On the one hand side the proof by decreasing diagrams might be easier to formalize and furthermore proofs by decreasing diagrams are constructive, cf. [144]. Another idea would be the certification of confluence proofs (based on decreasing diagrams) given by automated confluence provers.[5] Both aims require to lift our formalization from abstract rewriting to term rewriting, which is a natural idea for future work.

Acknowledgments

This research is supported by FWF P22467. We thank the anonymous reviewers, Bertram Felgenhauer, Nao Hirokawa, and Aart Middeldorp for helpful comments. Bertram Felgenhauer contributed an initial proof of Lemma 4.6(3) and located the formalization of [26].

[5]Certification is already established in the termination community where it has shown tools as well as termination criteria unsound.

Part III

Complexity

5. Modular Complexity Analysis for Term Rewriting

Publication Details

Harald Zankl and Martin Korp, Modular Complexity Analysis for Term Rewriting, Logical Methods in Computer Science 10(1:19), 33 pages, 2014. Special issue of RTA 2010.
doi: `10.2168/LMCS-10(1:19)2014`

Harald Zankl and Martin Korp, Modular Complexity Analysis via Relative Complexity, In Proceedings of the 21st International Conference on Rewriting Techniques and Applications (RTA 2010), Leibniz International Proceedings in Informatics 6, pp. 385–400, Schloss Dagstuhl, 2010.
doi: `10.4230/LIPIcs.RTA.2010.385`

Abstract

All current investigations to analyze the derivational complexity of term rewrite systems are based on a single termination method, possibly preceded by transformations. However, the exclusive use of direct criteria is problematic due to their restricted power. To overcome this limitation the article introduces a modular framework which allows to infer (polynomial) upper bounds on the complexity of term rewrite systems by combining different criteria. Since the fundamental idea is based on relative rewriting, we study how matrix interpretations and match-bounds can be used and extended to measure complexity for relative rewriting, respectively. The modular framework is proved strictly more powerful than the conventional setting. Furthermore, the results have been implemented and experiments show significant gains in power.

5.1. Introduction

Term rewriting is a Turing complete model of computation. As an immediate consequence all interesting properties are undecidable. Nevertheless many powerful techniques have been developed to establish *termination*. The majority of these techniques have been automated successfully. This development has been stimulated by the international competition of termination tools.[1] Most automated analyzers gain their power from a modular treatment of rewrite systems (typically via the dependency pair framework [13, 68, 174]).

For terminating rewrite systems Hofbauer and Lautemann [77] consider the length of derivations as a measurement for the complexity of rewrite systems. The resulting notion of *derivational complexity* relates the length of a rewrite sequence to the size of its starting term. Thereby it is, e.g., a suitable metric for the complexity of deciding the word problem for a given confluent and terminating rewrite system (since the decision procedure rewrites terms to normal form). If one regards a rewrite system as a program and wants to estimate the maximal number of computation steps needed to evaluate an expression to a result, then the special shape of the starting terms—a function applied to data which is in normal form—can be taken into account. Hirokawa and Moser [75] identified this special form of complexity and named it *runtime complexity*.

To show (feasible) upper complexity bounds currently few techniques are known. Typically termination criteria are restricted such that complexity bounds can be inferred. The early work by Hofbauer [80] considers polynomial interpretations, suitably restricted, to admit quadratic derivational complexity. Match-bounds [55] and arctic matrix interpretations [104] induce linear upper bounds on the derivational complexity and triangular matrix interpretations [126] admit at most polynomially long derivations (the dimension of the matrices yields the degree of the polynomial) in the size of the starting term. All these methods share the property that until now they have been used directly only, meaning that a single termination technique has to orient all rules in one go. However, using direct criteria exclusively is problematic due to their restricted power.

In [75, 76] Hirokawa and Moser lifted many aspects of the dependency pair framework from termination analysis into the complexity setting, resulting in the notion of weak dependency pairs. So for the special case of runtime complexity for the first time a modular approach has been introduced. There the modular aspect amounts to using different interpretation based criteria for (parts of the) weak dependency graph and the usable rules. However, still all rewrite rules considered must be oriented strictly in one go and only restrictive criteria may be applied for the usable rules. A further drawback of weak dependency pairs is that they may only be used for bounding runtime complexity while there seems to be no hope to generalize the method to derivational complexity.

In this article we present a different approach which admits a fully modular

[1] `http://termcomp.uibk.ac.at`

treatment. The approach is general enough that it applies to derivational complexity (and hence also to runtime complexity) and basic enough that it allows to combine completely different complexity criteria such as match-bounds and (triangular) matrix interpretations. By the modular combination of different base methods also gains in power are achieved. These gains come in two flavors. On the one hand our approach allows to obtain lower complexity bounds for several rewrite systems where bounds have already been established before and on the other hand we found bounds for systems that could not be dealt with so far automatically. More specifically, there are systems where the modular combination of different criteria allows to establish an upper bound while any of the involved methods cannot succeed on its own.

The remainder of the article is organized as follows. In Section 5.2 preliminaries about term rewriting and complexity analysis are fixed. Afterwards, Section 5.3 familiarizes the reader with the concept of a suitable complexity measurement for relative rewriting. Furthermore, it formulates a modular framework for complexity analysis based on relative complexity. Criteria for measuring relative complexity via interpretations and match-bounds are presented in Sections 5.4 and 5.5, respectively. In Section 5.6 we show that the modular setting is strictly more powerful than the conventional approach. Our results have been implemented in the complexity prover CaT. The technical details can be inferred from Section 5.7. Section 5.8 is devoted to demonstrate the power of the modular treatment by means of an empirical evaluation. Section 5.9 concludes.

This article is a restructured and extended version of [206]. It also incorporates the results from the two notes [204, 207] presented at informal workshops. Furthermore results and presentation have been generalized to address both derivational and runtime complexity.

5.2. Preliminaries

We assume familiarity with (relative) term rewriting [17, 56, 172]. Let $\mathcal{F}$ be a signature and $\mathcal{V}$ a disjoint set of variables. The set of terms over $\mathcal{F}$ and $\mathcal{V}$ is denoted by $\mathcal{T}(\mathcal{F},\mathcal{V})$ and the set of ground terms over $\mathcal{F}$ by $\mathcal{T}(\mathcal{F})$. We write $\mathsf{Fun}(t)$ for the set of function symbols occurring in a term t. The size of a term t is denoted $|t|$ and $\|t\|$ computes the number of occurrences of function symbols in t. A term t is called *linear* if any variable x occurs at most once in t. Positions are used to address symbol occurrences in terms. Given a term t we use $\mathsf{Pos}(t)$ to denote the set of positions induced by the term t and we write $t(p)$ with $p \in \mathsf{Pos}(t)$ for the symbol at position p in the term t. The subset of positions $p \in \mathsf{Pos}(t)$ such that $t(p) \in \mathcal{F}$ is denoted by $\mathsf{Pos}_{\mathcal{F}}(t)$.

A *rewrite rule* is a pair of terms (l, r), written $l \to r$ such that l is not a variable and all variables in r are contained in l. A rewrite rule $l \to r$ is *size-preserving* (*size-decreasing*) if $|l| = |r|$ ($|l| > |r|$). A *term rewrite system* (TRS for short) is a set of rewrite rules. For complexity analysis we assume TRSs to be finite and terminating. A TRS $\mathcal{R}$ is said to be *duplicating* if there exist a rewrite rule $l \to r \in \mathcal{R}$ and a variable x that occurs more often in r than in l. A TRS $\mathcal{R}$ is called *linear* (*left-linear*, *right-linear*) if for all rewrite rules $l \to r \in \mathcal{R}$ the terms l and r (l, r) are linear. We call a TRS $\mathcal{R}$ *collapsing* if it contains a rewrite rule $l \to r$ such that r is a variable. The *defined symbols* of a TRS $\mathcal{R}$ are all function symbols f for which there is a rewrite rule $l \to r$ in $\mathcal{R}$ such that $f = l(\epsilon)$. In the following we denote this set of function symbols by $\mathsf{Def}(\mathcal{R})$. Those function symbols of $\mathcal{R}$ which are not defined are called *constructor symbols*. So the set of all constructor symbols is defined as $\mathsf{Con}(\mathcal{R}) = \mathcal{F} \setminus \mathsf{Def}(\mathcal{R})$.

A *rewrite relation* is a binary relation on terms that is closed under contexts and substitutions. For a TRS $\mathcal{R}$ we define $\to_{\mathcal{R}}$ to be the smallest rewrite relation that contains $\mathcal{R}$. As usual $\to^*$ denotes the reflexive and transitive closure of $\to$ and $\to^m$ the m-th iterate of $\to$. A *relative* TRS $\mathcal{R}/\mathcal{S}$ is a pair of TRSs $\mathcal{R}$ and $\mathcal{S}$ with the induced rewrite relation $\to_{\mathcal{R}/\mathcal{S}} = \to^*_{\mathcal{S}} \cdot \to_{\mathcal{R}} \cdot \to^*_{\mathcal{S}}$. In the sequel we will sometimes identify a TRS $\mathcal{R}$ with the relative TRS $\mathcal{R}/\varnothing$ and vice versa. Furthermore properties defined for TRSs (as the ones above) naturally extend to relative TRSs.

The *derivation height* of a term t with respect to a relation $\to$ is defined as follows: $\mathsf{dh}(t, \to) = \mathsf{sup}\,\{m \mid \exists u\ t \to^m u\}$. The *complexity* of a relation $\to$ with respect to a (possibly infinite) set of terms (or language) L, denoted by $\mathsf{cp}_L(n, \to)$, computes the maximal derivation height of all terms in L up to size n and is defined as $\mathsf{cp}_L(n, \to) = \mathsf{sup}\,\{\mathsf{dh}(t, \to) \mid t \in L \text{ and } |t| \leqslant n\}$. Sometimes we say that a TRS $\mathcal{R}$ (relative TRS $\mathcal{R}/\mathcal{S}$) has linear, quadratic, etc. or polynomial complexity with respect to L if $\mathsf{cp}_L(n, \to_{\mathcal{R}})$ ($\mathsf{cp}_L(n, \to_{\mathcal{R}/\mathcal{S}})$) can be bounded by a linear, quadratic, etc. function or polynomial in n. Let $\mathcal{R}$ be a TRS over some signature $\mathcal{F}$. The *derivational complexity* of $\mathcal{R}$, abbreviated by $\mathsf{dc}(n, \mathcal{R})$ and defined as $\mathsf{dc}(n, \mathcal{R}) = \mathsf{cp}_{\mathcal{T}(\mathcal{F},\mathcal{V})}(n, \to_{\mathcal{R}})$, computes the complexity of $\to_{\mathcal{R}}$ with respect to all terms. In contrast, the *runtime complexity* of $\mathcal{R}$ considers the maximal derivation height of constructor-based terms only, i.e., $\mathsf{rc}(n, \mathcal{R}) = \mathsf{cp}_{\mathcal{T}_{\mathsf{Con}}(\mathcal{R},\mathcal{V})}(n, \to_{\mathcal{R}})$. Here, the set of *constructor-based terms* $\mathcal{T}_{\mathsf{Con}}(\mathcal{R}, \mathcal{V})$ is defined as the set of all terms $t = f(t_1, \ldots, t_m)$ such that $f \in \mathsf{Def}(\mathcal{R})$ and $t_i \in \mathcal{T}(\mathsf{Con}(\mathcal{R}), \mathcal{V})$ for all $i \in \{1, \ldots, m\}$.

For functions $f, g\colon \mathbb{N} \to \mathbb{N}$ we write $f(n) = \mathcal{O}(g(n))$ if there are constants $M, N \in \mathbb{N}$ such that $f(n) \leqslant M \cdot g(n)$ for all $n \geqslant N$. Furthermore, $f(n) = \Omega(g(n))$ if $g(n) = \mathcal{O}(f(n))$ and $f(n) = \Theta(g(n))$ if $f(n) = \mathcal{O}(g(n))$ and $f(n) = \Omega(g(n))$.

5.3. Modular Complexity via Relative Complexity

In this section we present the basic idea that allows a modular treatment of complexity proofs. To this end we introduce complexity analysis for relative rewriting, i.e., given a relative TRS $\mathcal{R}/\mathcal{S}$ only the $\mathcal{R}$-steps contribute to the complexity. To estimate the derivational complexity of a relative TRS $\mathcal{R}/\mathcal{S}$, a pair of orderings $(\succ, \succeq)$ will be used such that $\mathcal{R} \subseteq {\succ}$ and $\mathcal{S} \subseteq {\succeq}$. The necessary properties of these orderings are given in the next definition.

Definition 5.1. A *complexity pair* $(\succ, \succeq)$ consists of two finitely branching rewrite relations $\succ$ and $\succeq$ that are *compatible*, i.e., ${\succeq} \cdot {\succ} \subseteq {\succ}$ and ${\succ} \cdot {\succeq} \subseteq {\succ}$. We call a relative TRS $\mathcal{R}/\mathcal{S}$ *compatible* with a complexity pair $(\succ, \succeq)$ if $\mathcal{R} \subseteq {\succ}$ and $\mathcal{S} \subseteq {\succeq}$.

The next lemma states that given a relative TRS $\mathcal{R}/\mathcal{S}$ and a compatible complexity pair $(\succ, \succeq)$, the $\succ$ ordering is crucial for estimating the derivational complexity of $\mathcal{R}/\mathcal{S}$. Intuitively the result states that every $\mathcal{R}/\mathcal{S}$-step gives rise to at least one $\succ$-step.

Lemma 5.2. *Let $\mathcal{R}/\mathcal{S}$ be a relative TRS compatible with a complexity pair $(\succ, \succeq)$. Then for any term t we have $\mathsf{dh}(t, \succ) \geqslant \mathsf{dh}(t, \rightarrow_{\mathcal{R}/\mathcal{S}})$.*

Proof. By assumption $\mathcal{R}/\mathcal{S}$ is compatible with $(\succ, \succeq)$. Since $\succ$ and $\succeq$ are rewrite relations $\rightarrow_{\mathcal{R}} \subseteq {\succ}$ and $\rightarrow_{\mathcal{S}} \subseteq {\succeq}$ holds. From the compatibility of $\succ$ and $\succeq$ we obtain $\rightarrow_{\mathcal{R}/\mathcal{S}} \subseteq {\succ}$. Hence for any sequence

$$t \rightarrow_{\mathcal{R}/\mathcal{S}} t_1 \rightarrow_{\mathcal{R}/\mathcal{S}} t_2 \rightarrow_{\mathcal{R}/\mathcal{S}} \cdots$$

also

$$t \quad \succ \quad t_1 \quad \succ \quad t_2 \quad \succ \quad \cdots$$

holds. The result follows immediately from this. □

Obviously $\succ$ must be at least well-founded if *finite* complexities should be estimated. Because we are especially interested in feasible upper bounds the following corollary is specialized to polynomials.

Corollary 5.3. *Let $\mathcal{R}/\mathcal{S}$ be a relative TRS compatible with a complexity pair $(\succ, \succeq)$. If the complexity of $\succ$ with respect to some language L is linear, quadratic, etc. or polynomial then the complexity of $\mathcal{R}/\mathcal{S}$ with respect to L is linear, quadratic, etc. or polynomial.*

Proof. By Lemma 5.2. □

This corollary allows to investigate the complexity of (compatible) complexity pairs instead of the complexity of the underlying relative TRS. Sections 5.4 and 5.5 are dedicated to formulate powerful complexity pairs. A severe drawback of complexity pairs is that given a relative TRS $\mathcal{R}/\mathcal{S}$ all rules in $\mathcal{R}$ must be oriented strictly. In the following we present a modular approach which allows to combine different techniques for estimating the complexity of a relative TRS $\mathcal{R}/\mathcal{S}$ with respect to a language L. The fundamental idea is based on the following simple procedure. Instead of computing the complexity of $\mathcal{R}/\mathcal{S}$ at once we try to bound the complexity of $\mathcal{R}/\mathcal{S}$ by splitting $\mathcal{R}$ into smaller components $\mathcal{R}_1$ and $\mathcal{R}_2$. Here $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$. The aim is to over-estimate $\mathsf{dh}(t, \rightarrow_{\mathcal{R}/\mathcal{S}})$ by $\mathsf{dh}(t, \rightarrow_{\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})}) + \mathsf{dh}(t, \rightarrow_{\mathcal{R}_2/(\mathcal{R}_1 \cup \mathcal{S})})$. For each relative TRS $\mathcal{R}_i/(\mathcal{R}_{3-i} \cup \mathcal{S})$ with $i \in \{1, 2\}$ we can proceed in two directions: we can either split up $\mathcal{R}_i$ into smaller components or over-estimate $\mathsf{dh}(t, \rightarrow_{\mathcal{R}_i/(\mathcal{R}_{3-i} \cup \mathcal{S})})$ by applying some suitable method. (Section 5.7 shows that this choice is performed automatically.) Finally the complexity of the original system is determined by summing up all intermediate results. The next lemma states the main observation in this direction.

Lemma 5.4. *Let $(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}$ be a relative TRS and let t be a terminating term. Then* $\mathsf{dh}(t, \rightarrow_{\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})}) + \mathsf{dh}(t, \rightarrow_{\mathcal{R}_2/(\mathcal{R}_1 \cup \mathcal{S})}) \geqslant \mathsf{dh}(t, \rightarrow_{(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}})$.

Proof. We abbreviate $\mathcal{R}_1 \cup \mathcal{R}_2$ by $\mathcal{R}$ and $\mathcal{R}_{3-i} \cup \mathcal{S}$ by $\mathcal{S}_i$ for $i \in \{1, 2\}$. Assume that $\mathsf{dh}(t, \rightarrow_{\mathcal{R}/\mathcal{S}}) = m$. Then there exists a rewrite sequence

$$t \rightarrow_{\mathcal{R}/\mathcal{S}} t_1 \rightarrow_{\mathcal{R}/\mathcal{S}} t_2 \rightarrow_{\mathcal{R}/\mathcal{S}} \cdots \rightarrow_{\mathcal{R}/\mathcal{S}} t_{m-1} \rightarrow_{\mathcal{R}/\mathcal{S}} t_m \tag{1}$$

of length m. Next we investigate this sequence for every relative TRS $\mathcal{R}_i/\mathcal{S}_i$ $(1 \leqslant i \leqslant 2)$ where m_i overestimates how often rules from $\mathcal{R}_i$ have been applied in the original sequence. Fix i. If the sequence (1) does not contain an $\mathcal{R}_i$ step then $t \rightarrow^m_{\mathcal{S}_i} t_m$ and $m_i = 0$. In the other case there exists a maximal (with respect to m_i) sequence

$$t \rightarrow_{\mathcal{R}_i/\mathcal{S}_i} t_{j_1} \rightarrow_{\mathcal{R}_i/\mathcal{S}_i} t_{j_2} \rightarrow_{\mathcal{R}_i/\mathcal{S}_i} \cdots \rightarrow_{\mathcal{R}_i/\mathcal{S}_i} t_{j_{m_i-1}} \rightarrow_{\mathcal{R}_i/\mathcal{S}_i} t_m \tag{2}$$

where $1 \leqslant j_1 < j_2 < \cdots < j_{m_i} = m$. Together with the fact that every rewrite rule in $\mathcal{R}$ is contained in $\mathcal{R}_1$ or $\mathcal{R}_2$ we have $m_1 + m_2 \geqslant m$. If $m_i = 0$ we obviously have $\mathsf{dh}(t, \rightarrow_{\mathcal{R}_i/\mathcal{S}_i}) \geqslant m_i$ and if $t \rightarrow^{m_i}_{\mathcal{R}_i/\mathcal{S}_i} t_m$ with $m_i > 0$ we know that $\mathsf{dh}(t, \rightarrow_{\mathcal{R}_i/\mathcal{S}_i}) \geqslant m_i$ by the choice of sequence (2). (Note that in both cases it can happen that $\mathsf{dh}(t, \rightarrow_{\mathcal{R}_i/\mathcal{S}_i}) > m_i$ because sequence (1) need not be maximal with respect to $\rightarrow_{\mathcal{R}_i/\mathcal{S}_i}$.) Putting things together yields

$$\mathsf{dh}(t, \rightarrow_{\mathcal{R}_1/\mathcal{S}_1}) + \mathsf{dh}(t, \rightarrow_{\mathcal{R}_2/\mathcal{S}_2}) \geqslant m_1 + m_2 \geqslant m = \mathsf{dh}(t, \rightarrow_{\mathcal{R}/\mathcal{S}})$$

which concludes the proof. □

As already indicated in the proof, the statement of the above lemma does not hold for equality. This is illustrated by the following example.

Example 5.5. Consider the relative TRS $\mathcal{R}/\mathcal{S}$ with $\mathcal{R} = \{\mathsf{a} \to \mathsf{b}, \mathsf{a} \to \mathsf{c}\}$ and $\mathcal{S} = \varnothing$. We have $\mathsf{a} \to_{\mathcal{R}/\mathcal{S}} \mathsf{b}$ or $\mathsf{a} \to_{\mathcal{R}/\mathcal{S}} \mathsf{c}$. Hence $\mathsf{dh}(\mathsf{a}, \to_{\mathcal{R}/\mathcal{S}}) = 1$. However, the sum of the derivation heights $\mathsf{dh}(\mathsf{a}, \to_{\{\mathsf{a} \to \mathsf{b}\}/\{\mathsf{a} \to \mathsf{c}\}})$ and $\mathsf{dh}(\mathsf{a}, \to_{\{\mathsf{a} \to \mathsf{c}\}/\{\mathsf{a} \to \mathsf{b}\}})$ is 2.

Although for Lemma 5.4 equality cannot be established the next result states that for complexity analysis this does not matter.

Theorem 5.6. *Let $(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}$ be a relative TRS and L be a set of terminating terms. Then $\mathsf{cp}_L(n, \to_{(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}}) = \Theta(\mathsf{cp}_L(n, \to_{\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})}) + \mathsf{cp}_L(n, \to_{\mathcal{R}_2/(\mathcal{R}_1 \cup \mathcal{S})}))$.*

Proof. We have to show that there are constants M, N and M', N' such that for any term $t \in L$ the following two properties hold (for N and N' choose 0, i.e., a term t being a normal form):

- $\mathsf{dh}(t, \to_{(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}}) \leqslant M \cdot (\mathsf{dh}(t, \to_{\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})}) + \mathsf{dh}(t, \to_{\mathcal{R}_2/(\mathcal{R}_1 \cup \mathcal{S})}))$
- $M' \cdot \mathsf{dh}(t, \to_{(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}}) \geqslant \mathsf{dh}(t, \to_{\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})}) + \mathsf{dh}(t, \to_{\mathcal{R}_2/(\mathcal{R}_1 \cup \mathcal{S})})$

The result then follows from this. Lemma 5.4 shows the first property with $M = 1$. For the second property we reason as follows. Let $i \in \{1, 2\}$ and $\mathcal{S}_i = \mathcal{R}_{3-i} \cup \mathcal{S}$. Since $t \to_{\mathcal{R}_i/\mathcal{S}_i} t'$ implies $t \to^{+}_{(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}} t'$ also $\mathsf{dh}(t, \to_{(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}}) \geqslant \mathsf{dh}(t, \to_{\mathcal{R}_i/\mathcal{S}_i})$. The claim is shown by choosing $M' = 2$. $\square$

Theorem 5.6 allows to split a relative TRS $(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}$ into *smaller* components $\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})$ and $\mathcal{R}_2/(\mathcal{R}_1 \cup \mathcal{S})$ and evaluate the complexities of these components (e.g., by different complexity pairs) independently. Note that this approach is not restricted to relative rewriting. To estimate the complexity of a (non-relative) TRS $\mathcal{R}$ just consider the relative TRS $\mathcal{R}/\varnothing$. The next example shows how proofs in the modular framework look like. Section 5.7 gives more details on proof trees.

Example 5.7. Proofs in the modular setting can be viewed as trees. We sketch such a proof in Figure 5.1 using the TRS $\mathcal{R}$ consisting of the following five rules:

$$\begin{aligned}
1&: & \mathsf{rev}(x) &\to \mathsf{rev}'(x, \mathsf{nil}) \\
2&: & \mathsf{rev}'(\mathsf{nil}, y) &\to y \\
3&: & \mathsf{rev}'(\mathsf{cons}(x, y), z) &\to \mathsf{rev}'(y, \mathsf{append}(\mathsf{cons}(x, \mathsf{nil}), z)) \\
4&: & \mathsf{append}(\mathsf{nil}, y) &\to y \\
5&: & \mathsf{append}(\mathsf{cons}(x, y), z) &\to \mathsf{cons}(x, \mathsf{append}(y, z))
\end{aligned}$$

$$\begin{tikzcd}
& \mathcal{R} \arrow[d, "\mathcal{O}(1)"] & \\
& \mathcal{R}/\varnothing \arrow[dl, "\mathcal{O}(1)"'] \arrow[dr, "\mathcal{O}(1)"] & \\
\{2,4\}/\{1,3,5\} \arrow[d, "\mathcal{O}(n^2)"'] & & \{1,3,5\}/\{2,4\} \arrow[d, "\mathcal{O}(n^3)"] \\
\varnothing/\mathcal{R} & & \{1\}/\{2,3,4,5\} \arrow[d, "\mathcal{O}(n)"] \\
& & \varnothing/\mathcal{R}
\end{tikzcd}$$

Figure 5.1.: Sketch of a modular complexity proof.

The root node of the tree is the TRS of interest and the other nodes are relative rewrite systems representing intermediate complexity problems. The edges indicate the (derivational) complexity of the proof steps. It is possible to apply Theorem 5.6 explicitly to split a problem into two (or more) problems as demonstrated in the second node. Such situations do not affect the complexity of the given problem which justifies the labels $\mathcal{O}(1)$. The remaining proof steps measure the complexity of the rewrite rules that are moved from the first into the second component (relative to the remaining rules). These steps rely on an implicit application of Theorem 5.6. For instance in the proof tree shown in Figure 5.1 there is an edge from $\{1,3,5\}/\{2,4\}$ to $\{1\}/\{2,3,4,5\}$ labeled $\mathcal{O}(n^3)$, stating that the (derivational) complexity of $\{3,5\}/\{1,2,4\}$ is at most cubic. This step is sound because from Theorem 5.6 we know that computing an upper bound on $\{1\}/\{2,3,4,5\}$ and $\{3,5\}/\{1,2,4\}$ suffice to get a valid upper bound on $\{1,3,5\}/\{2,4\}$. In Sections 5.4 and 5.5 we study criteria that allow to perform such proof steps. Since the leaves in the tree give rise to constant complexity, the complexity of the original problem can be overestimated by summing up the complexities annotated to the edges; yielding a cubic upper bound in this exemplary case. Later (Example 5.53) we will see that this bound is not tight.

In the next two sections we study how matrix interpretations and the match-bounds technique can be suited for relative complexity analysis.

5.4. Matrix Interpretations

This section is aimed at formulating complexity pairs based on matrix interpretations [44]. Since our interest is in polynomial upper bounds, triangular matrix interpretations [126] and arctic matrix interpretations [103] are considered. The last part of this section generalizes the weight gap principle from [75] to (a restriction of) triangular matrix interpretations and relative rewriting.

5.4.1. Preliminaries

An *$\mathcal{F}$-algebra* $\mathcal{A}$ consists of a non-empty carrier A and a set of interpretations $f_{\mathcal{A}}$ for every $f \in \mathcal{F}$. By $[\alpha]_{\mathcal{A}}(\cdot)$ we denote the usual evaluation function of $\mathcal{A}$ according to an assignment α. An $\mathcal{F}$-algebra $\mathcal{A}$ together with two relations $\succ$ and $\succeq$ on A is called a *monotone algebra* if every $f_{\mathcal{A}}$ is monotone with respect to $\succ$ and $\succeq$, $\succ$ is a well-founded order, and $\succ$ and $\succeq$ are compatible. Any monotone algebra $(\mathcal{A}, \succ, \succeq)$ induces a well-founded order on terms, i.e., $s \succ_{\mathcal{A}} t$ if for any assignment α the condition $[\alpha]_{\mathcal{A}}(s) \succ [\alpha]_{\mathcal{A}}(t)$ holds. The order $\succeq_{\mathcal{A}}$ is defined similarly. A relative TRS $\mathcal{R}/\mathcal{S}$ is *compatible* with a monotone algebra $(\mathcal{A}, \succ, \succeq)$ if $\mathcal{R}/\mathcal{S}$ is compatible with $(\succ_{\mathcal{A}}, \succeq_{\mathcal{A}})$.

5.4.2. Triangular Matrix Interpretations

Matrix interpretations $(\mathcal{M}, \succ, \succeq)$ (often just denoted $\mathcal{M}$) are a special form of monotone algebras. Here the carrier is $\mathbb{N}^d$ for some fixed dimension $d \in \mathbb{N} \setminus \{0\}$. The order $\succeq$ is the point-wise extension of $\geqslant_{\mathbb{N}}$ to vectors and $\vec{u} \succ \vec{v}$ if $u_1 >_{\mathbb{N}} v_1$ and $\vec{u} \succeq \vec{v}$. If every $f \in \mathcal{F}$ of arity n is interpreted as $f_{\mathcal{M}}(\vec{x_1}, \dots, \vec{x_n}) = F_1\vec{x_1} + \cdots + F_n\vec{x_n} + \vec{f}$ where $F_i \in \mathbb{N}^{d \times d}$ for all $1 \leqslant i \leqslant n$ and $\vec{f} \in \mathbb{N}^d$ then monotonicity of $\succ$ is achieved by demanding $F_{i(1,1)} \geqslant 1$ for any $f \in \mathcal{F}$ and $1 \leqslant i \leqslant n$. Such interpretations have been introduced in [44].

A matrix interpretation where for every $f \in \mathcal{F}$ all F_i ($1 \leqslant i \leqslant n$ where n is the arity of f) are upper triangular is called *triangular matrix interpretation* (abbreviated by TMI). A square matrix A of dimension d is of *upper triangular* shape if $A_{(i,i)} \leqslant 1$ and $A_{(i,j)} = 0$ if $i > j$ for all $1 \leqslant i, j \leqslant d$. For historic reasons a TMI based on matrices of dimension one is also called *strongly linear interpretation* (SLI for short). In [126] it is shown that the derivational complexity of a TRS $\mathcal{R}$ is bounded by a polynomial of degree d if there exists a TMI $\mathcal{M}$ of dimension d such that $\mathcal{R} \subseteq \succ_{\mathcal{M}}$. For our setting the following formulation is more useful.

Theorem 5.8. *Let $\mathcal{M}$ be a TMI of dimension d over a signature $\mathcal{F}$. Then $(\succ_{\mathcal{M}}, \succeq_{\mathcal{M}})$ is a complexity pair. Furthermore $\mathsf{cp}_{\mathcal{T}(\mathcal{F},\mathcal{V})}(n, \succ_{\mathcal{M}}) = \mathcal{O}(n^d)$.*

Proof. Straightforward from [126, Theorem 6]. □

The following example familiarizes the reader with TMIs.

Example 5.9. Consider the relative TRS $\mathcal{R}/\mathcal{S}$ over the signature $\mathcal{F} = \{\mathsf{f}, \mathsf{g}\}$ defined as $\mathcal{R} = \{\mathsf{f}(\mathsf{f}(x)) \to \mathsf{f}(\mathsf{g}(\mathsf{f}(x)))\}$ and $\mathcal{S} = \{\mathsf{f}(x) \to x\}$. Then the TMI $\mathcal{M}$ of dimension two with

$$\mathsf{f}_{\mathcal{M}}(\vec{x}) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \vec{x} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad \mathsf{g}_{\mathcal{M}}(\vec{x}) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \vec{x}$$

induces the complexity pair $(\succ_{\mathcal{M}}, \succsim_{\mathcal{M}})$. Furthermore $\mathcal{R}/\mathcal{S}$ is compatible with the complexity pair $(\succ_{\mathcal{M}}, \succsim_{\mathcal{M}})$. Theorem 5.8 gives a quadratic upper bound on $\mathsf{cp}_{\mathcal{T}(\mathcal{F},\mathcal{V})}(n, \succ_{\mathcal{M}})$. Hence the derivational complexity of $\mathcal{R}/\mathcal{S}$ is at most quadratic by Corollary 5.3. It is easy to see (cf. Example 5.11) that this bound is not tight. We remark that there cannot exist an SLI that establishes a linear upper bound because no SLI can orient the rule $\mathsf{f}(\mathsf{f}(x)) \to \mathsf{f}(\mathsf{g}(\mathsf{f}(x)))$ strictly.

5.4.3. Arctic Matrix Interpretations

We define $\mathbb{A} = \mathbb{N} \cup \{-\infty\}$. For matrices $A \in \mathbb{A}^{n \times m}$ and $B \in \mathbb{A}^{m \times p}$ the operation $\otimes$ yields an $n \times p$ matrix and is defined as: $(A \otimes B)_{(i,j)} = \mathsf{max}_{1 \leqslant k \leqslant m}\{A_{(i,k)} + B_{(k,j)}\}$ where $+$ and max are extended naturally to deal with $-\infty$ (see [103]). Furthermore $x >_{\mathbb{A}} y$ if and only if $x >_{\mathbb{N}} y$ or $x = y = -\infty$, and finally $x \geqslant_{\mathbb{A}} y$ if and only if $x \geqslant_{\mathbb{N}} y$ or $y = -\infty$.[2]

An *arctic matrix interpretation* $(\mathcal{A}, \succ, \succsim)$ (abbreviated by AMI and often just denoted $\mathcal{A}$) is a special form of a monotone algebra. Here the carrier is $\mathbb{A}^d$ for some fixed dimension $d \in \mathbb{N} \setminus \{0\}$. The orders $\succsim$ and $\succ$ are the point-wise extensions of $\geqslant_{\mathbb{A}}$ and $>_{\mathbb{A}}$ to vectors, respectively. Every unary function symbol $f \in \mathcal{F}$ is interpreted as $f_{\mathcal{A}}(\vec{x}) = F \otimes \vec{x}$ where $F \in \mathbb{A}^{d \times d}$ and every constant c as $c_{\mathcal{A}} = \vec{c}$ where $\vec{c} \in \mathbb{A}^d$. Monotonicity of $\succ$ is achieved by the restriction to at most unary function symbols and by demanding that $F_{(1,1)}$ and c_1 are different from $-\infty$ for unary function symbols f and constants c, respectively. In [103] it is shown that the derivational complexity of a TRS $\mathcal{R}$, which contains unary and constant function symbols only, is at most linear if there exists an AMI $\mathcal{A}$ (of some dimension d) such that $\mathcal{R} \subseteq \succ_{\mathcal{A}}$.

Theorem 5.10. *Let $\mathcal{A}$ be an AMI of dimension d over a signature $\mathcal{F}$ that contains constants and unary function symbols only. Then $(\succ_{\mathcal{A}}, \succsim_{\mathcal{A}})$ is a complexity pair. Furthermore $\mathsf{cp}_{\mathcal{T}(\mathcal{F},\mathcal{V})}(n, \succ_{\mathcal{A}}) = \mathcal{O}(n)$.*

Proof. Straightforward from [103, Lemma 17]. □

[2]Note that $-\infty >_{\mathbb{A}} -\infty$ and hence $>_{\mathbb{A}}$ is not well-founded. Hence such comparisons are disallowed at certain matrix positions.

Example 5.11. Consider the TRSs from Example 5.9. Then the AMI $\mathcal{A}$ satisfying

$$\mathsf{f}_{\mathcal{A}}(\vec{x}) = \begin{pmatrix} 1 & 3 \\ 0 & 3 \end{pmatrix} \vec{x} \qquad\qquad \mathsf{g}_{\mathcal{A}}(\vec{x}) = \begin{pmatrix} 0 & 1 \\ -\infty & -\infty \end{pmatrix} \vec{x}$$

induces the complexity pair $(\succ_{\mathcal{A}}, \succeq_{\mathcal{A}})$ and $\mathcal{R}/\mathcal{S}$ is compatible with $(\succ_{\mathcal{A}}, \succeq_{\mathcal{A}})$. Theorem 5.10 gives a linear upper bound on $\mathsf{cp}_{\mathcal{T}(\mathcal{F},\mathcal{V})}(n, \succ_{\mathcal{A}})$. Hence the derivational complexity of $\mathcal{R}/\mathcal{S}$ is at most linear by Corollary 5.3. It is easy to see that this bound is tight.

5.4.4. Complexity Gap Principle

An obvious question is whether it suffices to estimate polynomial complexity of $(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}$ by establishing polynomial upper bounds on the complexities of $\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})$ and $\mathcal{R}_2/\mathcal{S}$ (in contrast to $\mathcal{R}_2/(\mathcal{R}_1 \cup \mathcal{S})$ as in Theorem 5.6). The following example by Hofbauer [78] shows that in the general case the complexity of $(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}$ might be much larger than the sum of the components above; even for systems where both parts have linear complexity. Here $\mathcal{S} = \varnothing$.

Example 5.12. Consider the TRS $\mathcal{R}_1$ consisting of the single rule

$$\mathsf{c}(\mathsf{L}(x)) \to \mathsf{R}(x)$$

and the TRS $\mathcal{R}_2$ consisting of the rewrite rules

$$\mathsf{R}(\mathsf{a}(x)) \to \mathsf{b}(\mathsf{b}(\mathsf{R}(x))) \qquad \mathsf{R}(x) \to \mathsf{L}(x) \qquad \mathsf{b}(\mathsf{L}(x)) \to \mathsf{L}(\mathsf{a}(x))$$

The derivational complexity of the relative TRS $\mathcal{R}_1/\mathcal{R}_2$ is linear, due to the SLI that just counts the c's. The derivational complexity of $\mathcal{R}_2$ is linear as well since the system can be proved terminating by the match-bound technique [55]. However, the TRS $\mathcal{R}_1 \cup \mathcal{R}_2$ admits exponentially long derivations in the size of the starting term:

$$\begin{aligned} \mathsf{c}^n(\mathsf{L}(\mathsf{a}(x))) &\to \mathsf{c}^{n-1}(\mathsf{R}(\mathsf{a}(x))) \quad \to \mathsf{c}^{n-1}(\mathsf{b}(\mathsf{b}(\mathsf{R}(x)))) \to \mathsf{c}^{n-1}(\mathsf{b}(\mathsf{b}(\mathsf{L}(x)))) \\ &\to \mathsf{c}^{n-1}(\mathsf{b}(\mathsf{L}(\mathsf{a}(x)))) \to \mathsf{c}^{n-1}(\mathsf{L}(\mathsf{a}(\mathsf{a}(x)))) \to^* \mathsf{L}(\mathsf{a}^{2^n}(x)) \end{aligned}$$

Under certain circumstances the problem from the preceding example does not occur. Inspired by the weight gap principle of Hirokawa and Moser [75] (which was developed to estimate weak dependency pair steps relative to usable rule steps), below we state abstract criteria on $\mathcal{R}_1$ and $\mathcal{R}_2$ such that the complexity of $\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})$ and $\mathcal{R}_2/\mathcal{S}$ determines the complexity of $(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}$.

Theorem 5.13 (Complexity Gap Principle)**.** *Let $(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}$ be a relative TRS and L be a set of terminating terms. If there exist a complexity pair $(\succ, \succsim)$ and a constant Δ such that $\mathcal{R}_2/\mathcal{S}$ is compatible with $(\succ, \succsim)$ and $u \to_{\mathcal{R}_1} v$ implies $\mathsf{dh}(u, \succ) + \Delta \geqslant \mathsf{dh}(v, \succ)$ then $\mathsf{cp}_L(n, \to_{(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}}) = \mathcal{O}(\mathsf{cp}_L(n, \to_{\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})}) + \mathsf{cp}_L(n, \succ))$.*

Proof. We show that under the above assumptions, for any term $s \in L$ there exists a constant M such that $\mathsf{dh}(s, \to_{(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}}) \leqslant M \cdot \mathsf{dh}(s, \to_{\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})}) + \mathsf{dh}(s, \succ)$. Consider a derivation of maximal length in $(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}$, written as follows where $s = s_0$:

$$s_0 \to^{k_0}_{\mathcal{R}_2/\mathcal{S}} \cdot \to^*_{\mathcal{S}} t_0 \to_{\mathcal{R}_1} s_1 \to^{k_1}_{\mathcal{R}_2/\mathcal{S}} \cdot \to^*_{\mathcal{S}} t_1 \to_{\mathcal{R}_1} \cdots \to_{\mathcal{R}_1} s_m \to^{k_m}_{\mathcal{R}_2/\mathcal{S}} \cdot \to^*_{\mathcal{S}} t_m \qquad (3)$$

We have $\mathsf{dh}(s_0, \to_{(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}}) \leqslant \mathsf{dh}(s_0, \to_{\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})}) + \sum_{0 \leqslant i \leqslant m} k_i$ since sequence (3) is maximal. Because $\mathcal{R}_2/\mathcal{S}$ is compatible with $(\succ, \succsim)$ we conclude that $\mathsf{dh}(s_0, \succ) \geqslant \mathsf{dh}(t_0, \succ) + k_0$. From the assumption, $\mathsf{dh}(t_0, \succ) + \Delta \geqslant \mathsf{dh}(s_1, \succ)$ follows and hence $\mathsf{dh}(s_0, \succ) + \Delta \geqslant \mathsf{dh}(s_1, \succ) + k_0$. Repeating this argument establishes that $\mathsf{dh}(s_0, \succ) + m \cdot \Delta \geqslant \sum_{0 \leqslant i \leqslant m} k_i$. Because $m \leqslant \mathsf{dh}(s_0, \to_{\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})})$ (note that equality does not hold since sequence (3) need not be maximal for $\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})$) we obtain $\mathsf{dh}(s_0, \to_{(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}}) \leqslant \mathsf{dh}(s_0, \to_{\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})}) + \mathsf{dh}(s_0, \succ) + \mathsf{dh}(s_0, \to_{\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})}) \cdot \Delta$ which simplifies to $\mathsf{dh}(s_0, \to_{(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}}) \leqslant (\Delta + 1) \cdot \mathsf{dh}(s_0, \to_{\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})}) + \mathsf{dh}(s_0, \succ)$. Finally, taking $M = \Delta + 1$ concludes the proof. ☐

To implement the above theorem the question arises which further requirements besides compatibility of $\mathcal{R}_2/\mathcal{S}$ with a complexity pair $(\succ, \succsim)$ are required such that for any terms u and v a step $u \to_{\mathcal{R}_1} v$ implies the desired $\mathsf{dh}(u, \succ) + \Delta \geqslant \mathsf{dh}(s, \succ)$ for some constant Δ. One idea is to test $\mathsf{dh}(l, \succ) + \Delta \geqslant \mathsf{dh}(r, \succ)$ explicitly for any $l \to r \in \mathcal{R}_1$ and demand that the complexity pair $(\succ, \succsim)$ then satisfies $\mathsf{dh}(C[l\sigma], \succ) + \Delta \geqslant \mathsf{dh}(C[r\sigma], \succ)$ for all contexts C and substitutions σ.

As we know from [75], SLIs can be used to get a concrete instance of Theorem 5.13 with respect to derivational complexity, if $\mathcal{S}$ is empty. Below we state the result in the relative setting, which is more useful for our purposes.

Corollary 5.14. *Let $(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}$ be a relative TRS, $\mathcal{R}_1$ be non-duplicating, and $\mathcal{R}_2/\mathcal{S}$ be compatible with an SLI. Then the following relationship is satisfied $\mathsf{dc}(n, (\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}) = \mathcal{O}(\mathsf{dc}(n, \mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})) + n)$.*

Proof. Follows from Theorems 5.13 and 5.8 using the complexity pair $(\succ_{\mathcal{M}}, \succsim_{\mathcal{M}})$ induced by the SLI $\mathcal{M}$. ☐

An immediate consequence of the above corollary is that for a relative TRS $\mathcal{R}/\mathcal{S}$ we can shift rewrite rules in $\mathcal{R}$ that are strictly oriented by an SLI $\mathcal{M}$ into the $\mathcal{S}$-component, provided that $\mathcal{R}$ is non-duplicating and all rules in $\mathcal{S}$ behave nicely with respect to $\succsim_{\mathcal{M}}$. Note that the above corollary does not require that all rules from $\mathcal{R}$ are (strictly) oriented. This causes some kind of non-determinism which is demonstrated in the next example.

Example 5.15. Consider the TRS ($\mathsf{Bouchare\_06/12}$)[3] consisting of the rules:

$$1\colon \mathsf{b}(\mathsf{b}(x) \to \mathsf{a}(\mathsf{a}(\mathsf{a}(x))) \quad 2\colon \mathsf{b}(\mathsf{a}(\mathsf{b}(x))) \to \mathsf{a}(x) \quad 3\colon \mathsf{b}(\mathsf{a}(\mathsf{a}(x))) \to \mathsf{b}(\mathsf{a}(\mathsf{b}(x)))$$

The SLI $\mathcal{M}$ with $\mathsf{a}_{\mathcal{M}}(x) = x + 2$ and $\mathsf{b}_{\mathcal{M}}(x) = x + 1$ transforms the TRS into $\{1\}/\{2, 3\}$ which is compatible with the AMI $\mathcal{A}$ (where all matrix coefficients are smaller than two)

$$\mathsf{a}_{\mathcal{A}}(\vec{x}) = \begin{pmatrix} 0 & 0 & 0 \\ -\infty & -\infty & 0 \\ -\infty & 0 & -\infty \end{pmatrix} \vec{x} \qquad \mathsf{b}_{\mathcal{A}}(\vec{x}) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ -\infty & 0 & -\infty \end{pmatrix} \vec{x}$$

showing linear derivational complexity of this TRS. If a different SLI is used in the first step, e.g., the one that counts just b's then the intermediate problem $\{3\}/\{1, 2\}$ remains to be solved. For this problem there exists no AMI of dimension three where all entries are less than 2 (but there exists one where all entries are less than 3). For an implementation this means that depending on the rules the SLI orients, later techniques may succeed or fail.

Next we remark on another subtlety of Theorem 5.13. Assume that $\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})$ is compatible with a complexity pair $(\succ, \succsim)$. Then $(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}$ is transformed into the problem $\mathcal{R}_2/(\mathcal{R}_1 \cup \mathcal{S})$ and this proof step estimates the complexity of $\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})$. If the complexity gap principle is used the situation changes. Since it does not require (weak) compatibility with $\mathcal{R}_1$, it does not make a statement about the complexity of $\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})$. Instead it states that the complexity of $(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}$ is dominated by the complexity of $\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})$ or the complexity of $\mathcal{R}_2/\mathcal{S}$. This behavior is illustrated in the next example.

Example 5.16. Consider the relative TRS $\mathcal{R}$ consisting of the two rules

$$1 : \mathsf{c}(x) \to \mathsf{a}(x) \qquad\qquad 2 : \mathsf{a}(\mathsf{b}(x)) \to \mathsf{b}(\mathsf{b}(\mathsf{c}(x)))$$

We observe that the derivational complexity of the TRS $\mathcal{R}$ is at least exponential because

$$\mathsf{a}^n(\mathsf{b}(x)) \to^2 \mathsf{a}^{n-1}(\mathsf{b}(\mathsf{b}(\mathsf{a}(x)))) \to^4 \mathsf{a}^{n-2}(\mathsf{b}(\mathsf{b}(\mathsf{b}(\mathsf{b}(\mathsf{a}(x)))))) \to^8 \cdots \to^{2^n} \mathsf{b}^{2^n}(\mathsf{a}(x))$$

[3]Labels in sans-serif font refer to TRSs from the TPDB 7.0.2, see `http://termination-portal.org`.

Obviously both rules are applied exponentially often in this sequence. Nevertheless by an SLI that counts c's Corollary 5.14 can be applied to $\mathcal{R}$ to obtain the relative TRS $\{2\}/\{1\}$. As remarked earlier this step does not yield an upper bound on the complexity of the TRS $\{1\}/\{2\}$ but only on the TRS $\{1\}$.

Next we give counterexamples that TMIs, AMIs, and match-bounds cannot be used to implement Theorem 5.13. A suitable but severe restriction of TMIs is considered in Theorem 5.18.

Example 5.17. Recall the two TRSs $\mathcal{R}_1$ and $\mathcal{R}_2$ from Example 5.12. Here $\mathcal{S} = \varnothing$. Since $\mathsf{dc}(n, \mathcal{R}_1/\mathcal{R}_2) = \mathcal{O}(n)$ and $\mathsf{dc}(n, \mathcal{R}_1 \cup \mathcal{R}_2) = \Omega(2^n)$ any method that establishes $\mathsf{dc}(n, \mathcal{R}_2) = \mathcal{O}(n^k)$ for some $k \in \mathbb{N}$ cannot be used to implement the complexity gap principle.

Since the TMI $\mathcal{M}$ with

$$\mathsf{a}_{\mathcal{M}}(\vec{x}) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \vec{x} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad \mathsf{R}_{\mathcal{M}}(\vec{x}) = \begin{pmatrix} 1 & 3 \\ 0 & 0 \end{pmatrix} \vec{x} + \begin{pmatrix} 2 \\ 0 \end{pmatrix}$$

$$\mathsf{b}_{\mathcal{M}}(\vec{x}) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \vec{x} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \mathsf{L}_{\mathcal{M}}(\vec{x}) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \vec{x}$$

orients all rules in $\mathcal{R}_2$ strictly—and hence gives a quadratic upper bound on $\mathsf{dc}(n, \mathcal{R}_2)$—in general TMIs cannot adhere to Theorem 5.13. The problem for the interpretation above is that although there exists a Δ with $\mathsf{dh}(l, \succ) + \Delta \geqslant \mathsf{dh}(r, \succ)$ for all $l \to r \in \mathcal{R}_1$ this property is not closed under substitutions. (The situation is different, however, if the matrix interpretation has constant growth, see Theorem 5.18 below.)

Similarly, the AMI $\mathcal{A}$ (inducing at most linear derivational complexity of $\mathcal{R}_2$) with

$$\mathsf{a}_{\mathcal{A}}(\vec{x}) = \begin{pmatrix} 0 & -\infty \\ 3 & 3 \end{pmatrix} \vec{x} \qquad \mathsf{R}_{\mathcal{A}}(\vec{x}) = \begin{pmatrix} 1 & 3 \\ 0 & 2 \end{pmatrix} \vec{x}$$

$$\mathsf{b}_{\mathcal{A}}(\vec{x}) = \begin{pmatrix} 1 & 2 \\ -\infty & 0 \end{pmatrix} \vec{x} \qquad \mathsf{L}_{\mathcal{A}}(\vec{x}) = \begin{pmatrix} 0 & -\infty \\ -\infty & -\infty \end{pmatrix} \vec{x}$$

violates the same requirement in Theorem 5.13 as the TMI $\mathcal{M}$ above.

A similar reasoning also holds for match-bounds; one easily verifies that match-bounds apply to the TRS $\mathcal{R}_2$ and hence this system admits linear derivational complexity. The problem in this setting is that a valid termination proof of $\mathcal{R}_2$ using match-bounds does not necessarily yield a rewrite relation $\succ$ such that $\mathsf{dh}(u, \succ) + \Delta \geqslant \mathsf{dh}(v, \succ)$ whenever $u \to_{\mathcal{R}_1} v$, as required by Theorem 5.13.

Finally we present a criterion that allows to implement Theorem 5.13 based on TMIs. To this end we introduce the following concepts. A matrix interpretation $\mathcal{M}$ has *constant growth* if there is a matrix A such that for any $p \in \mathbb{N}$ and matrices $M_1, \ldots, M_p$ in $\mathcal{M}$ we have $M_1 \cdot \ldots \cdot M_p \leqslant A$. Here $\leqslant$ is the pointwise extension of $\leqslant_{\mathbb{N}}$ to matrices. Because of the shape of matrix interpretations for terms s and t there exist $k \in \mathbb{N}$, matrices $S_1, \ldots, S_k, T_1, \ldots, T_k$, and vectors $\vec{s}$, $\vec{t}$ such that $[\alpha]_{\mathcal{M}}(s) = S_1\alpha(x_1) + \cdots + S_k\alpha(x_k) + \vec{s}$ and $[\alpha]_{\mathcal{M}}(t) = T_1\alpha(x_1) + \cdots + T_k\alpha(x_k) + \vec{t}$. In such a case we denote the non-constant part of the interpretation of s by $[\alpha]^{\mathtt{ncp}}_{\mathcal{M}}(s) = S_1\alpha(x_1) + \cdots + S_k\alpha(x_k)$; similarly for t. We write $s \succeq^{\mathtt{ncp}}_{\mathcal{M}} t$ if $[\alpha]^{\mathtt{ncp}}_{\mathcal{M}}(s) \succeq [\alpha]^{\mathtt{ncp}}_{\mathcal{M}}(t)$ holds for all assignments α. Note that this condition can effectively be tested by requiring $S_i \geqslant T_i$ $(1 \leqslant i \leqslant k)$.

Theorem 5.18. *Let $(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}$ be a relative TRS, L a set of terminating terms, $\mathcal{M}$ a matrix interpretation with constant growth, $\mathcal{R}_1 \subseteq \succeq^{\mathtt{ncp}}_{\mathcal{M}}$, and $\mathcal{R}_2/\mathcal{S}$ be compatible with $\mathcal{M}$. Then $\mathsf{cp}_L(n, \to_{(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}}) = \mathcal{O}(\mathsf{cp}_L(n, \to_{\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})}) + n)$.*

Proof. Throughout this proof we assume that $L = \mathcal{T}(\mathcal{F}, \mathcal{V})$. Since the matrix interpretation $\mathcal{M}$ has constant growth we have $\mathsf{cp}_L(n, \succ_{\mathcal{M}}) = \mathcal{O}(n)$. Since $\mathcal{R}_2/\mathcal{S}$ is compatible with the complexity pair $(\succ_{\mathcal{M}}, \succeq_{\mathcal{M}})$ using Theorem 5.13 it remains to show that there is a constant Δ such that $u \to_{\mathcal{R}_1} v$ implies $\mathsf{dh}(u, \succ_{\mathcal{M}}) + \Delta \geqslant \mathsf{dh}(v, \succ_{\mathcal{M}})$. As $\mathcal{M}$ has constant growth there is a matrix A such that $A \geqslant M_1 \cdot \ldots \cdot M_p$ for any $p \in \mathbb{N}$ where the M_i's are matrices occurring in $\mathcal{M}$. Let $\delta = \mathsf{max}\,\{\vec{r} \mid l \to r \in \mathcal{R}_1\}$ (here $\vec{r}$ is the constant part of the interpretation of r and max denotes the pointwise maximum of vectors). Note that δ is a vector.

Let $\Delta = (A\delta)_{11}$. The derivation height of a term t wrt. $\succ_{\mathcal{M}}$ is determined by the first component of the vector $[\alpha]_{\mathcal{M}}(t)$. Hence $\mathsf{dh}(u, \succ_{\mathcal{M}}) + \Delta \geqslant \mathsf{dh}(v, \succ_{\mathcal{M}})$ whenever $[\alpha]_{\mathcal{M}}(u) + A\delta \geqslant [\alpha]_{\mathcal{M}}(v)$. To show the latter let $l \to r \in \mathcal{R}_1$, $u = C[l\sigma]$, $v = C[r\sigma]$, $[\alpha]_{\mathcal{M}}(l) = L_1\alpha(x_1) + \cdots + L_k\alpha(x_k) + \vec{l}$, and $[\alpha]_{\mathcal{M}}(r) = R_1\alpha(x_1) + \cdots + R_k\alpha(x_k) + \vec{r}$.

By definition of δ we have $\vec{l} + \delta \geqslant \vec{r}$. Since $\mathcal{R}_1 \subseteq \succeq^{\mathtt{ncp}}_{\mathcal{M}}$ we have $L_i \geqslant R_i$ for all $1 \leqslant i \leqslant k$ and hence $L_1\alpha(x_1) + \cdots + L_k\alpha(x_k) + \vec{l} + \delta \geqslant R_1\alpha(x_1) + \cdots + R_k\alpha(x_k) + \vec{r}$ for any α, hence $L_1\alpha(x_1\sigma) + \cdots + L_k\alpha(x_k\sigma) + \vec{l} + \delta \geqslant R_1\alpha(x_1\sigma) + \cdots + R_k\alpha(x_k\sigma) + \vec{r}$ for any σ. The latter implies $D(L_1\alpha(x_1\sigma) + \cdots + L_k\alpha(x_k\sigma) + \vec{l} + \delta) \geqslant D(R_1\alpha(x_1\sigma) + \cdots + R_k\alpha(x_k\sigma) + \vec{r})$ for any non-negative matrix D and especially we get $DL_1\alpha(x_1\sigma) + \cdots + DL_k\alpha(x_k\sigma) + D\vec{l} + A\delta \geqslant DR_1\alpha(x_1\sigma) + \cdots + DR_k\alpha(x_k\sigma) + D\vec{r}$ if $A \geqslant D$ (which is no restriction since $\mathcal{M}$ has constant growth and any D that can occur is a matrix product of the shape $M_1 \cdot \ldots \cdot M_p \leqslant A$ for some $p \in \mathbb{N}$). The proof concludes by the observation that the above inequation implies $[\alpha]_{\mathcal{M}}(C[l\sigma]) + A\delta \geqslant [\alpha]_{\mathcal{M}}(C[r\sigma])$ for any context C. $\square$

We conclude this section with a discussion of the above theorem. Due to [131, Theorem 9] TMIs where each matrix M satisfies $M_{(i,i)} < 1$ for any $i \geqslant 2$ have constant growth. Since SLIs trivially adhere to this restriction Theorem 5.18 subsumes Corollary 5.14. The next example shows that this inclusion is strict.

Example 5.19. Let $\mathcal{R}_1 = \{\mathsf{a}(x) \to \mathsf{c}(x)\}$, $\mathcal{R}_2 = \{\mathsf{a}(\mathsf{b}(\mathsf{a}(x))) \to \mathsf{a}(\mathsf{b}(\mathsf{b}(\mathsf{a}(x))))\}$, and $\mathcal{S} = \varnothing$. Then the TMI $\mathcal{M}$ with

$$\mathsf{a}_{\mathcal{M}}(\vec{x}) = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \vec{x} \qquad \mathsf{b}_{\mathcal{M}}(\vec{x}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} + \vec{x}$$

where $\mathsf{c}_{\mathcal{M}}(\vec{x}) = \mathsf{a}_{\mathcal{M}}(\vec{x})$ has constant growth and transforms $(\mathcal{R}_1 \cup \mathcal{R}_2)/\mathcal{S}$ into $\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})$ according to Theorem 5.18. However, there exists no SLI that orients the rule in $\mathcal{R}_2$ strictly which shows that Corollary 5.14 cannot achieve this step.

5.5. Relative Match-Bounds

In this section we illustrate how the match-bound technique can be used to prove relative termination and estimate complexity bounds for relative rewriting. To maximize the power of the method we combine the ideas in [186] with the ones in [206]. Preliminaries for match-bounds are introduced in Section 5.5.1. Section 5.5.2 shows how the technique works for linear systems before Section 5.5.3 extends applicability to non-left-linear systems. Automation is addressed in Section 5.5.4. Throughout this section we consider $L \subseteq \mathcal{T}(\mathcal{F})$ which does not affect the results by assuming that the signature $\mathcal{F}$ always contains a constant.

5.5.1. Preliminaries

Let $\mathcal{F}$ be a signature, $\mathcal{R}$ a TRS over $\mathcal{F}$, and $L \subseteq \mathcal{T}(\mathcal{F})$ a set of ground terms. The set $\{t \in \mathcal{T}(\mathcal{F}) \mid s \to_{\mathcal{R}}^* t \text{ for some } s \in L\}$ of reducts of L is denoted by $\to_{\mathcal{R}}^*(L)$. Given a set $N \subseteq \mathbb{N}$ of natural numbers, the signature $\mathcal{F} \times N$ is denoted by $\mathcal{F}_N$. Here function symbols (f, c) with $f \in \mathcal{F}$ and $c \in N$ have the same arity as f and are written as f_c. The mappings $\mathsf{lift}_c \colon \mathcal{F} \to \mathcal{F}_{\mathbb{N}}$, $\mathsf{base} \colon \mathcal{F}_{\mathbb{N}} \to \mathcal{F}$, and $\mathsf{height} \colon \mathcal{F}_{\mathbb{N}} \to \mathbb{N}$ are defined as $\mathsf{lift}_c(f) = f_c$, $\mathsf{base}(f_c) = f$, and $\mathsf{height}(f_c) = c$ for all $f \in \mathcal{F}$ and $c \in \mathbb{N}$. They are extended to terms, sets of terms, and TRSs in the obvious way. The TRS $\mathsf{raise}(\mathcal{F})$ over the signature $\mathcal{F}_{\mathbb{N}}$ consists of all rules $f_c(x_1, \ldots, x_n) \to f_{c+1}(x_1, \ldots, x_n)$ with f an n-ary function symbol in $\mathcal{F}$, $c \in \mathbb{N}$, and $x_1, \ldots, x_n$ pairwise distinct variables. The restriction of $\mathsf{raise}(\mathcal{F})$ to the signature $\mathcal{F}_{\{0,\ldots,c\}}$ is denoted by $\mathsf{raise}_c(\mathcal{F})$. For terms $s, t \in \mathcal{T}(\mathcal{F}_{\mathbb{N}}, \mathcal{V})$ we write $s \uparrow t$ for the least term u with $s \to_{\mathsf{raise}(\mathcal{F})}^* u$

and $t \to^*_{\mathsf{raise}(\mathcal{F})} u$. Here least refers to the (sum of the) lengths of the joining sequences. We extend this notion to $\uparrow S$ for finite non-empty sets $S \subseteq \mathcal{T}(\mathcal{F}_\mathbb{N}, \mathcal{V})$ in the obvious way. Note that $\uparrow S$ is undefined whenever S contains two terms s and t such that $\mathsf{base}(s) \neq \mathsf{base}(t)$. The TRS $\mathsf{match}(\mathcal{R})$ over the signature $\mathcal{F}_\mathbb{N}$ consists of all rewrite rules $l' \to \mathsf{lift}_c(r)$ for which there exists a rule $l \to r \in \mathcal{R}$ such that $\mathsf{base}(l') = l$ and $c = 1 + \mathsf{min}\{\mathsf{height}(l'(p)) \mid p \in \mathsf{Pos}_\mathcal{F}(l)\}$. Here $c \in \mathbb{N}$. The restriction of $\mathsf{match}(\mathcal{R})$ to the signature $\mathcal{F}_{\{0,\ldots,c\}}$ is denoted by $\mathsf{match}_c(\mathcal{R})$. To be able to apply the match-bound technique to non-left-linear TRSs we define the relation $\xrightarrow{\mathsf{r}}_{\mathsf{match}(\mathcal{R})}$ on $\mathcal{T}(\mathcal{F}_\mathbb{N}, \mathcal{V})$ as follows: $s \xrightarrow{\mathsf{r}}_{\mathsf{match}(\mathcal{R})} t$ if and only if there exist a rewrite rule $l \to r \in \mathsf{match}(\mathcal{R})$, a position $p \in \mathsf{Pos}(s)$, a context C, and terms $s_1, \ldots, s_n$ such that $l = C[x_1, \ldots, x_n]$ with all variables displayed, $s|_p = C[s_1, \ldots, s_n]$, $\mathsf{base}(s_i) = \mathsf{base}(s_j)$ whenever $x_i = x_j$ for all $i, j \in \{1, \ldots, n\}$, and $t = s[r\sigma]_p$. Here the substitution σ is defined as follows:

$$\sigma(x) = \begin{cases} \uparrow\{s_i \mid x_i = x \text{ with } i \in \{1, \ldots, n\}\} & \text{if } x \in \{x_1, \ldots, x_n\} \\ x & \text{otherwise} \end{cases}$$

Let L be a set of ground terms. A TRS $\mathcal{R}$ is called *match-bounded* for L if there exists a $c \in \mathbb{N}$ such that the maximum height of function symbols occurring in terms in $\to^*_{\mathsf{match}(\mathcal{R})}(\mathsf{lift}_0(L))$ is at most c. Similarly, a TRS $\mathcal{R}$ is called *match-raise-bounded* for L if there exists a $c \in \mathbb{N}$ such that the maximum height of function symbols occurring in terms belonging to $\xrightarrow{\mathsf{r}}^*_{\mathsf{match}(\mathcal{R})}(\mathsf{lift}_0(L))$ is at most c. If we want to make the bound c precise, we say that $\mathcal{R}$ is match(-raise)-bounded for L *by* c. If we do not specify the set of terms L then it is assumed that $L = \mathcal{T}(\mathcal{F})$. The main result underlying the match-bound technique states that a TRS $\mathcal{R}$ is terminating for a language L if $\mathcal{R}$ is linear and match-bounded for L or $\mathcal{R}$ is non-duplicating and match-raise-bounded for L.

In order to prove that a TRS $\mathcal{R}$ is match(-raise)-bounded for some language L, the idea is to construct a (quasi-deterministic and raise-consistent) tree automaton that is compatible with $\mathsf{match}(\mathcal{R})$ and $\mathsf{lift}_0(L)$. In the following we briefly recall the most important definitions in this connection. A *tree automaton* $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ consists of a signature $\mathcal{F}$, a finite set of states Q, a set of final states $Q_f \subseteq Q$, and a set of transitions Δ of the form $f(q_1, \ldots, q_n) \to q$ or $p \to q$ where f is an n-ary function symbol in $\mathcal{F}$ and $p, q, q_1, \ldots, q_n \in Q$. The language $\mathcal{L}(\mathcal{A})$ of $\mathcal{A}$ is the set of ground terms $t \in \mathcal{T}(\mathcal{F})$ such that $t \to^*_\Delta q$ for some $q \in Q_f$. We say that $\mathcal{A}$ is *compatible* with a TRS $\mathcal{R}$ and a language L if $L \subseteq \mathcal{L}(\mathcal{A})$ and for each rewrite rule $l \to r \in \mathcal{R}$ and state substitution $\sigma\colon \mathsf{Var}(l) \to Q$ such that $l\sigma \to^*_\Delta q$ it holds that $r\sigma \to^*_\Delta q$. For left-linear $\mathcal{R}$ it is known that $\to^*_\mathcal{R}(L) \subseteq \mathcal{L}(\mathcal{A})$ whenever $\mathcal{A}$ is compatible with $\mathcal{R}$ and L [53]. To obtain a similar result for non-left-linear TRSs, in [106] quasi-deterministic automata are introduced. Let $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ be a tree automaton. We say that a state p *subsumes* a state q if p is final when q is final and for all transitions $f(u_1, \ldots, q, \ldots, u_n) \to u \in \Delta$, the transition $f(u_1, \ldots, p, \ldots, u_n) \to u$ belongs to Δ. For a left-hand side $l \in \mathsf{lhs}(\Delta)$ of a transition, the set $\{q \mid l \to q \in \Delta\}$ of possible right-hand sides is denoted by $Q(l)$. The automaton $\mathcal{A}$ is said to be *quasi-deterministic* if for every $l \in \mathsf{lhs}(\Delta)$ there exists a state $p \in Q(l)$ which subsumes every other state in $Q(l)$. In general, $Q(l)$ may contain more than one state that satisfies the above property. In the following we assume that there is a unique designated state in $Q(l)$, which we denote by p_l. The set of all designated states is denoted by Q_d and the restriction of Δ to transitions $l \to q$ that satisfy $q = p_l$ is denoted by Δ_d. In [106] it is shown that the tree automaton induced by Δ_d is deterministic and accepts the same language as $\mathcal{A}$. For non-left-linear TRSs $\mathcal{R}$ we modify the above definition of compatibility by demanding that the tree automaton $\mathcal{A}$ is quasi-deterministic and for each rewrite rule $l \to r \in \mathcal{R}$ and state substitution $\sigma\colon \mathsf{Var}(l) \to Q_d$ with $l\sigma \to^*_{\Delta_d} q$ it holds that $r\sigma \to^*_{\Delta} q$. To ensure that quasi-deterministic and compatible tree automata can be used to prove match-raise-boundedness of a TRS $\mathcal{R}$ it must be guaranteed that the obtained tree automata are closed under the implicit raise-steps caused by the relation $\xrightarrow{\mathsf{r}}_{\mathsf{match}(\mathcal{R})}$. To this end we additionally require that the resulting tree automata fulfill the property defined below. Let $\mathcal{A} = (\mathcal{F}_N, Q, Q_f, \Delta)$ be a tree automaton with N a finite subset of $\mathbb{N}$. We say that $\mathcal{A}$ is *raise-consistent* if for every transition $f_c(q_1, \ldots, q_n) \to q \in \Delta$ and left-hand side $f_d(q_1, \ldots, q_n) \in \mathsf{lhs}(\Delta)$ with $c <_{\mathbb{N}} d$, the transition $f_d(q_1, \ldots, q_n) \to q$ belongs to Δ.

By a remark in [55] we know that the derivation height of a term in L is at most linear in the size of the term whenever $\mathcal{R}$ is match-bounded for L. It is easy to extend this result to match-raise-boundedness and hence to non-duplicating TRSs. To this end we need the following notions. Let $\mathcal{M}\mathsf{ul}(\mathbb{N})$ denote the set of all finite multisets over $\mathbb{N}$. For any $M \in \mathcal{M}\mathsf{ul}(\mathbb{N})$ we write $M(n)$ to denote how often the number $n \in \mathbb{N}$ occurs in M. Let $M, N \in \mathcal{M}\mathsf{ul}(\mathbb{N})$ be two multisets. We write $M \cup N$ for the multiset sum of M and N where $(M \cup N)(n) = M(n) + N(n)$ for all $n \in \mathbb{N}$ and $M \subseteq N$ for the multiset inclusion, i.e., $M(n) \leqslant N(n)$ for all $n \in \mathbb{N}$. The multiset difference $M \setminus N$ is defined as $(M \setminus N)(n) = M(n) - N(n)$ if $M(n) > N(n)$ and $(M \setminus N)(n) = 0$ otherwise, for all $n \in \mathbb{N}$. We write $M \succ_{\mathsf{mul}} N$ if there are multisets X and Y such that $N = (M \setminus X) \cup Y$, $X \neq \varnothing$, and for all $m \in Y$ there is an $n \in X$ such that $n <_{\mathbb{N}} m$. We write $M \succeq_{\mathsf{mul}} N$ if $M \succ_{\mathsf{mul}} N$ or $M = N$. Let $\mathcal{F}$ be some signature. We extend the orderings $\succ_{\mathsf{mul}}$ and $\succeq_{\mathsf{mul}}$ to terms over the signature $\mathcal{F}_{\mathbb{N}}$ as follows: we have $s \succ_{\mathsf{mul}} t$ if $\mathcal{H}(s) \succ_{\mathsf{mul}} \mathcal{H}(t)$ and $s \succeq_{\mathsf{mul}} t$ if $\mathcal{H}(s) \succeq_{\mathsf{mul}} \mathcal{H}(t)$ for terms $s, t \in \mathcal{T}(\mathcal{F}_{\mathbb{N}}, \mathcal{V})$. Here $\mathcal{H}(t) = \{\mathsf{height}(t(p)) \mid p \in \mathsf{Pos}_{\mathcal{F}}(t)\}$ denotes the multiset of the heights of function symbols occurring in the term t.

Theorem 5.20. *Let $\mathcal{R}$ be a TRS and L be a language. If $\mathcal{R}$ is linear and match-bounded or non-duplicating and match-raise-bounded for L then we have $\mathsf{cp}_L(n, \to_{\mathcal{R}}) = \mathcal{O}(n)$.*

Proof. Assume that $\mathcal{R}$ is match-raise-bounded for L and hence terminating on L. (Note that for a linear TRS $\mathcal{R}$, match-boundedness coincides with match-raise-boundedness.) Let

$$t \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t_{m-1} \to_{\mathcal{R}} t_m$$

be an arbitrary (terminating) rewrite sequence with $t \in L$. Since every $\to_{\mathcal{R}}$ rewrite sequence can be lifted to a $\xrightarrow{\mathsf{r}}_{\mathsf{match}(\mathcal{R})}$ rewrite sequence [105, Lemma 12] we obtain a derivation

$$t' \xrightarrow{\mathsf{r}}_{\mathsf{match}(\mathcal{R})} t'_1 \xrightarrow{\mathsf{r}}_{\mathsf{match}(\mathcal{R})} \cdots \xrightarrow{\mathsf{r}}_{\mathsf{match}(\mathcal{R})} t'_{m-1} \xrightarrow{\mathsf{r}}_{\mathsf{match}(\mathcal{R})} t'_m$$

such that $t' = \mathsf{lift}_0(t)$ and $\mathsf{base}(t'_i) = t_i$ for all $i \in \{1, \ldots, m\}$. The proof of [105, Lemma 8] yields that for any non-duplicating TRS $\mathcal{R}$ we have $\xrightarrow{\mathsf{r}}_{\mathsf{match}(\mathcal{R})} \subseteq \succ_{\mathsf{mul}}$. It follows that $t'_i \succ_{\mathsf{mul}} t'_{i+1}$ for all $i \in \{0, \ldots, m-1\}$. Here $t'_0 = t'$. Since $\mathcal{R}$ is match-raise-bounded for L, all terms in this latter sequence belong to $\mathcal{T}(\mathcal{F}_{\{0,\ldots,c\}})$ for some $c \in \mathbb{N}$. Let k be the maximal number of function symbols occurring in some right-hand side in $\mathcal{R}$. Due to a remark in [42] we know that the length of the $\succ_{\mathsf{mul}}$ chain from t' to t'_m is bounded by $\|t'\| \cdot (k+1)^c$. Since $\|t'\| = \|t\|$ and the $\succ_{\mathsf{mul}}$ chain starting at t' is at least as long as the lifted and hence original rewrite sequence, we conclude that the length of the $\mathcal{R}$-rewrite sequence starting at the term t is bounded by $\|t\| \cdot (k+1)^c$. ☐

Based on Theorem 5.20 it is easy to use the match-bound technique to estimate the complexity of a relative TRS $\mathcal{R}/\mathcal{S}$; just check for match(-raise)-boundedness of $\mathcal{R} \cup \mathcal{S}$. This process either succeeds by proving that the combined TRS is match(-raise)-bounded, or, when $\mathcal{R} \cup \mathcal{S}$ cannot be proved to be match(-raise)-bounded, it fails. Since the construction of a (quasi-deterministic, raise-consistent, and) compatible tree automaton does not terminate when applied to TRSs that are not match(-raise)-bounded, the latter situation typically does not happen. This behavior causes a serious problem since we cannot benefit from relative rewriting, i.e., $\mathcal{R}/\mathcal{S}$ is match(-raise)-bounded if and only if $\mathcal{R} \cup \mathcal{S}$ is. In [186] this problem has been addressed by specifying an upper bound on the heights that can be introduced by rewrite rules in $\mathsf{match}(\mathcal{S})$. So one tries to find a $c \in \mathbb{N}$ such that the maximum height of function symbols occurring in reductions with the TRS $\mathsf{match}_{c+1}(\mathcal{R}) \cup \mathsf{match}_c(\mathcal{S}) \cup \mathsf{lift}_c(\mathcal{S})$ is at most c. If such a bound can be established we know that $\mathcal{R}/\mathcal{S}$ is terminating and in addition that it admits at most linear complexity. In the following we extend this approach to better suit relative rewriting. To this end we introduce a new enrichment $\mathsf{match}\text{-}\mathsf{RT}^c(\mathcal{R}/\mathcal{S})$ where the rewrite rules in

$\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})$ which originate from size-preserving or size-decreasing rules in $\mathcal{S}$ are labeled in such a way that they do not increase the heights of the function symbols in a contracted redex.

To simplify the presentation we first consider linear TRSs only. The extension to non-duplicating TRSs is explained in Section 5.5.3.

5.5.2. RT-Bounds for Left-Linear Relative TRSs

As proposed in [186] we design the new enrichment $\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})$ such that rules originating from $\mathcal{S}$ may introduce function symbols with height at most c. In addition (as in [206]) we try to keep the heights of the function symbols in a contracted redex if a size-preserving or size-decreasing rewrite rule in $\mathcal{S}$ (after dropping all heights) is applied.

Definition 5.21. Let $\mathcal{S}$ be a TRS over a signature $\mathcal{F}$ and $c \in \mathbb{N}$. The TRS $\mathsf{match\text{-}RT}^c(\mathcal{S})$ over the signature $\mathcal{F}_{\mathbb{N}}$ consists of all rules $l' \to \mathsf{lift}_d(r)$ such that $\mathsf{base}(l') \to r \in \mathcal{S}$ and

$$d = \begin{cases} \min\{c, \mathsf{height}(l'(\epsilon))\} & \text{if } \|\mathsf{base}(l')\| \geqslant \|r\| \text{ and} \\ & \mathsf{lift}_{\mathsf{height}(l'(\epsilon))}(\mathsf{base}(l')) = l' \\ \min\{c, 1 + \mathsf{height}(l'(p)) \mid p \in \mathsf{Pos}_{\mathcal{F}}(l')\} & \text{otherwise} \end{cases}$$

For a relative TRS $\mathcal{R}/\mathcal{S}$ we define $\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})$ as $\mathsf{match}(\mathcal{R})/\mathsf{match\text{-}RT}^c(\mathcal{S})$. Let $d \in \mathbb{N}$. The restriction of $\mathsf{match\text{-}RT}^c(\mathcal{S})$ to the signature $\mathcal{F}_{\{0,\ldots,d\}}$ is denoted by $\mathsf{match\text{-}RT}^c_d(\mathcal{S})$. Likewise the relative TRS $\mathsf{match\text{-}RT}^c_d(\mathcal{R}/\mathcal{S})$ is defined as $\mathsf{match}_d(\mathcal{R})/\mathsf{match\text{-}RT}^c_d(\mathcal{S})$. In case $c = d$ then $\mathsf{match\text{-}RT}^c_d(\mathcal{R}/\mathcal{S})$ is abbreviated by $\mathsf{match\text{-}RT}_c(\mathcal{R}/\mathcal{S})$ and $\mathsf{match\text{-}RT}^c_d(\mathcal{S}) = \mathsf{match\text{-}RT}_c(\mathcal{S})$.

The idea behind the requirement $\|\mathsf{base}(l')\| \geqslant \|r\|$ in the above definition is that such rules cannot yield an increase with respect to the multiset measure of heights. Let us illustrate the above definition on an example.

Example 5.22. Consider the relative TRS $\mathcal{R}/\mathcal{S}$ with $\mathcal{R}$ consisting of the rewrite rule

$$1\colon \mathsf{rev}(x) \to \mathsf{rev}'(x, \mathsf{nil})$$

and $\mathcal{S}$ consisting of the rewrite rules

$$2\colon \mathsf{rev}'(\mathsf{nil}, y) \to y \qquad\qquad 3\colon \mathsf{rev}'(\mathsf{cons}(x, y), z) \to \mathsf{rev}'(y, \mathsf{cons}(x, z))$$

Then the rewrite rules

$$\begin{array}{ll} \mathsf{rev}_0(x) \to \mathsf{rev}'_1(x, \mathsf{nil}_1) & \mathsf{rev}_1(x) \to \mathsf{rev}'_2(x, \mathsf{nil}_2) \\ \mathsf{rev}_2(x) \to \mathsf{rev}'_3(x, \mathsf{nil}_3) & \cdots \end{array}$$

belong to $\mathsf{match}(\mathcal{R})$ and $\mathsf{match\text{-}RT}^1(\mathcal{S})$ contains the rules

$$\begin{array}{rl} \mathsf{rev}'_0(\mathsf{nil}_0, y) \to y & \mathsf{rev}'_0(\mathsf{cons}_0(x, y), z) \to \mathsf{rev}'_0(y, \mathsf{cons}_0(x, z)) \\ \mathsf{rev}'_0(\mathsf{nil}_1, y) \to y & \mathsf{rev}'_0(\mathsf{cons}_1(x, y), z) \to \mathsf{rev}'_1(y, \mathsf{cons}_1(x, z)) \\ \cdots & \mathsf{rev}'_2(\mathsf{cons}_1(x, y), z) \to \mathsf{rev}'_1(y, \mathsf{cons}_1(x, z)) \end{array}$$

Both TRSs together constitute $\mathsf{match\text{-}RT}^1(\mathcal{R}/\mathcal{S})$.

The new enrichment $\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})$ allows to prove the complexity of the rewrite rules in $\mathcal{R}$ relative to the rules in $\mathcal{S}$.

Definition 5.23. Let $\mathcal{R}/\mathcal{S}$ be a relative TRS. We call $\mathcal{R}/\mathcal{S}$ *match-RT-bounded* for a language L if there exists a $c \in \mathbb{N}$ such that the height of function symbols occurring in terms in $\to^*_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})}(\mathsf{lift}_0(L))$ is at most c.

An immediate consequence of the next lemma is that every derivation in $\mathcal{R}/\mathcal{S}$ can be lifted to a $\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})$-sequence of the same length. This result is used later on to infer termination and complexity results for relative rewriting.

Lemma 5.24. *Let $\mathcal{R}/\mathcal{S}$ be a left-linear relative TRS and $c \in \mathbb{N}$. If $u \to_{\mathcal{R}} v$ ($u \to_{\mathcal{S}} v$) then for all terms u' with $\mathsf{base}(u') = u$ there exists a term v' such that $\mathsf{base}(v') = v$ and $u' \to_{\mathsf{match}(\mathcal{R})} v'$ ($u' \to_{\mathsf{match\text{-}RT}^c(\mathcal{S})} v'$).*

Proof. Straightforward. □

To be able to prove that a relative TRS $\mathcal{R}/\mathcal{S}$ admits a linear upper complexity bound whenever it is match-RT-bounded for a language L we slightly modify the orderings $\succ_{\mathsf{mul}}$ and $\succeq_{\mathsf{mul}}$. Let $M, N \in \mathcal{M}\mathsf{ul}(\mathbb{N})$ be multisets. The function $\mathsf{drop}_n(M)$ removes all occurrences of the number $n \in \mathbb{N}$ from M. So for all $m \in \mathbb{N}$ we have $\mathsf{drop}_n(M)(m) = 0$ if $m = n$ and $\mathsf{drop}_n(M)(m) = M(m)$ otherwise. The orderings $\succ^c_{\mathsf{mul}}$ and $\succeq^c_{\mathsf{mul}}$ are defined as $M \succ^c_{\mathsf{mul}} N$ if $\mathsf{drop}_c(M) \succ_{\mathsf{mul}} \mathsf{drop}_c(N)$ and $M \succeq^c_{\mathsf{mul}} N$ if $\mathsf{drop}_c(M) \succeq_{\mathsf{mul}} \mathsf{drop}_c(N)$. Let $\mathcal{F}$ be some signature. We extend $\succ^c_{\mathsf{mul}}$ and $\succeq^c_{\mathsf{mul}}$ to terms over the signature $\mathcal{F}_{\mathbb{N}}$ as follows: we have $s \succ^c_{\mathsf{mul}} t$ if $\mathcal{H}(s) \succ^c_{\mathsf{mul}} \mathcal{H}(t)$ and $s \succeq^c_{\mathsf{mul}} t$ if $\mathcal{H}(s) \succeq^c_{\mathsf{mul}} \mathcal{H}(t)$ for terms $s, t \in \mathcal{T}(\mathcal{F}_{\mathbb{N}}, \mathcal{V})$. The basic idea behind the new orderings $\succ^c_{\mathsf{mul}}$ and $\succeq^c_{\mathsf{mul}}$ is that rewrite rules in $\mathsf{match\text{-}RT}_c(\mathcal{R}/\mathcal{S})$ which originate from $\mathcal{R}$ are compatible with $\succ^c_{\mathsf{mul}}$ and the rules originating from $\mathcal{S}$ are compatible with $\succeq^c_{\mathsf{mul}}$. However there is one problem. If $\mathcal{R}$ contains a collapsing rule $l \to r$ then the rule $\mathsf{lift}_c(l) \to \mathsf{lift}_c(r)$ appears in $\mathsf{match\text{-}RT}_c(\mathcal{R}/\mathcal{S})$ which cannot be oriented via the ordering $\succ^c_{\mathsf{mul}}$ although $\mathsf{lift}_c(l) \succ_{\mathsf{mul}} \mathsf{lift}_c(r)$. The problem is that collapsing rewrite rules do not increase the heights of function symbols in a contracted redex because the right-hand sides consist just of single variables. To avoid this problem we assume in the following that $\mathcal{R}$ is non-collapsing. For collapsing $\mathcal{R}$ one could follow the approach in [206] which can handle collapsing rewrite rules because it does not not use an upper bound c to limit the heights that can be introduced by the enriched system. However, a disadvantages of this approach is that the heights of a contracted redex are increased more often. So, apart from the collapsing case the approach presented here is more powerful than the one introduced in [206] and completely subsumes the approach in [186].

Lemma 5.25. *Let $\mathcal{R}$ and $\mathcal{S}$ be two non-duplicating TRSs and $c \in \mathbb{N}$. If $\mathcal{R}$ is non-collapsing then $\to_{\mathsf{match}_c(\mathcal{R})} \subseteq \succ^c_{\mathsf{mul}}$ and $\to_{\mathsf{match\text{-}RT}_c(\mathcal{S})} \subseteq \succeq^c_{\mathsf{mul}}$.*

Proof. From the proof of [55, Lemma 17] we know that for a non-duplicating TRS $\mathcal{R}$ and terms s and t such that $s \to_{\mathsf{match}_c(\mathcal{R})} t$ we have $s \succ_{\mathsf{mul}} t$. So there are multisets X and Y such that $\mathcal{H}(t) = (\mathcal{H}(s) \setminus X) \cup Y$, $X \neq \varnothing$, and for all $d' \in Y$ there is a $d \in X$ such that $d <_{\mathbb{N}} d'$. Because $\mathcal{R}$ is non-collapsing we know from the definition of $\mathsf{match}_c(\mathcal{R})$ that there is a $d \in X$ such that $d <_{\mathbb{N}} c$ and $d <_{\mathbb{N}} d'$ for all $d' \in Y$. From this it follows that $\mathsf{drop}_c(\mathcal{H}(t)) = (\mathsf{drop}_c(\mathcal{H}(s)) \setminus \mathsf{drop}_c(X)) \cup \mathsf{drop}_c(Y)$, $\mathsf{drop}_c(X) \neq \varnothing$, and for all $d' \in \mathsf{drop}_c(Y)$ there is a $d \in \mathsf{drop}_c(X)$ such that $d <_{\mathbb{N}} d'$. As an immediate consequence we have $\mathsf{drop}_c(\mathcal{H}(s)) \succ_{\mathsf{mul}} \mathsf{drop}_c(\mathcal{H}(t))$ and hence $s \succ^c_{\mathsf{mul}} t$.

Now let s and t be terms and $l \to r$ be a rewrite rule in $\mathsf{match\text{-}RT}_c(\mathcal{S})$ such that $s \to_{\{l \to r\}} t$. According to Definition 5.21 we have to consider two cases. The first case amounts to $\|l\| \geqslant \|r\|$ where all function symbols in l and r have the same heights. But then non-duplication of $\mathcal{S}$ implies $\mathcal{H}(s) \supseteq \mathcal{H}(t)$ and thus $s \succeq^c_{\mathsf{mul}} t$. In the other case if $l \to r$ is non-collapsing and $l \notin \mathsf{lift}_c(\mathsf{base}(l))$ then we obtain $s \succ^c_{\mathsf{mul}} t$ as before and hence also $s \succeq^c_{\mathsf{mul}} t$. If $l \in \mathsf{lift}_c(\mathsf{base}(l))$ then $\mathsf{drop}_c(\mathcal{H}(s)) \supseteq \mathsf{drop}_c(\mathcal{H}(t))$ since $\mathsf{drop}_c(\mathcal{H}(l)) = \mathsf{drop}_c(\mathcal{H}(r)) = \varnothing$ and if $l \to r$ is collapsing then $\mathcal{H}(s) \supseteq \mathcal{H}(t)$ since $\mathcal{H}(r) = \varnothing$. Hence in both situations $s \succeq^c_{\mathsf{mul}} t$. $\square$

Since the length of every $\succ^c_{\mathsf{mul}}$ chain is bounded by a function linear in the size of the starting term—if the size-increase of the terms in the chain can be bounded by a constant—we can prove that the complexity induced by the relative TRS $\mathcal{R}/\mathcal{S}$ on some language L is at most linear if $\mathcal{R}/\mathcal{S}$ is match-RT-bounded for L.

Theorem 5.26. *Let $\mathcal{R}/\mathcal{S}$ be a linear relative TRS and $\mathcal{R}$ be non-collapsing. If $\mathcal{R}/\mathcal{S}$ is match-RT-bounded for a language L then $\mathcal{R}/\mathcal{S}$ is terminating on L and $\mathsf{cp}_L(n, \to_{\mathcal{R}/\mathcal{S}}) = \mathcal{O}(n)$.*

Proof. First we show that $\mathcal{R}/\mathcal{S}$ is terminating on L. Assume to the contrary that there is an infinite rewrite sequence of the form

$$t_1 \to_{\mathcal{R}/\mathcal{S}} t_2 \to_{\mathcal{R}/\mathcal{S}} t_3 \to_{\mathcal{R}/\mathcal{S}} \cdots$$

with $t_1 \in L$. Because $\mathcal{R} \cup \mathcal{S}$ is left-linear and $\mathcal{R}/\mathcal{S}$ is match-RT-bounded for L by a $c \in \mathbb{N}$, according to Lemma 5.24, the above derivation can be lifted to an infinite $\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})$ rewrite sequence

$$t'_1 \to_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})} t'_2 \to_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})} t'_3 \to_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})} \cdots$$

starting from $t'_1 = \mathsf{lift}_0(t_1)$ such that $\mathsf{base}(t'_i) = t_i$ for all $i \geqslant 1$ and the height of every function symbol occurring in a term in the lifted sequence is at most c. Hence the employed rewrite rules in the derivation emanating from t'_1 must come from $\mathsf{match\text{-}RT}_c(\mathcal{R}/\mathcal{S})$. With help of Lemma 5.25, transitivity of $\succeq^c_{\mathsf{mul}}$, and compatibility of the orderings $\succ^c_{\mathsf{mul}}$ and $\succeq^c_{\mathsf{mul}}$ we deduce that $t'_i \succ^c_{\mathsf{mul}} t'_{i+1}$ for all $i \geqslant 1$. However, this is excluded because $<_{\mathbb{N}}$ is well-founded on $\{0, \ldots, c\}$ and hence $\succ^c_{\mathsf{mul}}$ is well-founded on $\mathcal{T}(\mathcal{F}_{\{0,\ldots,c\}}, \mathcal{V})$.

To prove the second part of the theorem, consider an arbitrary (terminating) rewrite sequence

$$u \to_{\mathcal{R}/\mathcal{S}} u_1 \to_{\mathcal{R}/\mathcal{S}} \cdots \to_{\mathcal{R}/\mathcal{S}} u_m$$

with $u \in L$. This rewrite sequence can be lifted to a $\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})$-sequence of the same length

$$u' \to_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})} u'_1 \to_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})} \cdots \to_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})} u'_m$$

similar as before such that $u' = \mathsf{lift}_0(u)$ and $u'_i \succ^c_{\mathsf{mul}} u'_{i+1}$ for all $i \in \{0, \ldots, m-1\}$. Here $u'_0 = u'$ and $c \in \mathbb{N}$ such that the relative TRS $\mathcal{R}/\mathcal{S}$ is match-RT-bounded for L by c. Similar as in the proof of Theorem 5.20 we can conclude that the length of the $\mathcal{R}/\mathcal{S}$-rewrite sequence starting at the term u is bounded by $\|u\| \cdot (k+1)^c$ where k is the maximal number of function symbols occurring in some right-hand side in $\mathcal{R} \cup \mathcal{S}$; just replace $\succ_{\mathsf{mul}}$ by $\succ^c_{\mathsf{mul}}$. $\square$

We conclude this subsection with an example.

Example 5.27. The relative TRS $\mathcal{R}/\mathcal{S}$ of Example 5.22 is match-RT-bounded for $\mathcal{T}(\mathcal{F})$ by 1. Here $\mathcal{F} = \{\mathsf{nil}, \mathsf{cons}, \mathsf{rev}, \mathsf{rev}'\}$. Due to Theorem 5.26 we can conclude that $\mathcal{R}/\mathcal{S}$ admits at most linear derivational complexity. In Section 5.5.4 it is explained how match-RT-boundedness can be checked automatically.

5.5.3. Raise-RT-Bounds for Non-Left-Linear Relative TRSs

In order to generalize Theorem 5.26 to non-duplicating relative TRSs we consider the relation $\xrightarrow{\mathsf{r}}_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})}$ instead of $\rightarrow_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})}$ which uses raise-rules to deal with non-left-linearity. Thereby the rewrite relation $\xrightarrow{\mathsf{r}}_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})}$ is defined as $\xrightarrow{\mathsf{r}}^*_{\mathsf{match\text{-}RT}^c(\mathcal{S})} \cdot \xrightarrow{\mathsf{r}}_{\mathsf{match}(\mathcal{R})} \cdot \xrightarrow{\mathsf{r}}^*_{\mathsf{match\text{-}RT}^c(\mathcal{S})}$ where $\xrightarrow{\mathsf{r}}_{\mathsf{match\text{-}RT}^c(\mathcal{S})}$ is defined similar to $\xrightarrow{\mathsf{r}}_{\mathsf{match}(\mathcal{R})}$ (but based on $\mathsf{match\text{-}RT}^c(\mathcal{S})$ instead of $\mathsf{match}(\mathcal{R})$). This is essential to lift rewrite sequences in the relative TRS $\mathcal{R}/\mathcal{S}$ to sequences in $\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})$.

Definition 5.28. Let $\mathcal{R}/\mathcal{S}$ be a relative TRS. We call $\mathcal{R}/\mathcal{S}$ *match-raise-RT-bounded* for a language L if there exists a number $c \in \mathbb{N}$ such that the height of function symbols occurring in terms belonging to $\xrightarrow{\mathsf{r}}^*_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})}(\mathsf{lift}_0(L))$ is at most c.

Note that for left-linear relative TRSs, match-raise-RT-boundedness coincides with match-RT-boundedness. By using the relation $\xrightarrow{\mathsf{r}}_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})}$ every derivation induced by the relative TRS $\mathcal{R}/\mathcal{S}$ can be simulated via the rewrite rules in $\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})$.

Lemma 5.29. *Let $\mathcal{R}/\mathcal{S}$ be a relative TRS and $c \in \mathbb{N}$. If $u \rightarrow_{\mathcal{R}} v$ ($u \rightarrow_{\mathcal{S}} v$) then for all terms u' with $\mathsf{base}(u') = u$ there exists a term v' such that $\mathsf{base}(v') = v$ and $u' \xrightarrow{\mathsf{r}}_{\mathsf{match}(\mathcal{R})} v'$ ($u' \xrightarrow{\mathsf{r}}_{\mathsf{match\text{-}RT}^c(\mathcal{S})} v'$).*

Proof. Straightforward. □

Before we can prove that match-raise-RT-boundedness of $\mathcal{R}/\mathcal{S}$ induces a linear upper bound on the complexity we have to ensure that the raise-rules implicitly used by the relation $\xrightarrow{\mathsf{r}}_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})}$ can be oriented via $\succeq^c_{\mathsf{mul}}$.

Lemma 5.30. *For any signature $\mathcal{F}$ and $c \in \mathbb{N}$ it holds that $\rightarrow_{\mathsf{raise}_c(\mathcal{F})} \subseteq \succeq^c_{\mathsf{mul}}$.*

Proof. Assume that there are terms s and t such that $s \rightarrow_{\mathsf{raise}_c(\mathcal{F})} t$. According to the definition of $\mathsf{raise}_c(\mathcal{F})$ we have $\mathcal{H}(t) = (\mathcal{H}(s) \setminus \{d\}) \cup \{d+1\}$ for some height $d <_{\mathbb{N}} c$. Thus $s \succ^c_{\mathsf{mul}} t$ and hence $s \succeq^c_{\mathsf{mul}} t$ according to the definition of $\succeq^c_{\mathsf{mul}}$. □

Using Lemma 5.30 it is easy to extend Theorem 5.26 to TRSs that are non-linear but non-duplicating.

Theorem 5.31. *Let $\mathcal{R}/\mathcal{S}$ be a non-duplicating relative TRS and let $\mathcal{R}$ be non-collapsing. If $\mathcal{R}/\mathcal{S}$ is match-raise-RT-bounded for a language L then $\mathcal{R}/\mathcal{S}$ is terminating on L. Furthermore, $\mathsf{cp}_L(n, \rightarrow_{\mathcal{R}/\mathcal{S}}) = \mathcal{O}(n)$.*

Proof. First we show that $\mathcal{R}/\mathcal{S}$ is terminating on L. Assume to the contrary that there is an infinite rewrite sequence of the form

$$t_1 \to_{\mathcal{R}/\mathcal{S}} t_2 \to_{\mathcal{R}/\mathcal{S}} t_3 \to_{\mathcal{R}/\mathcal{S}} \cdots$$

with $t_1 \in L$. Let $\mathcal{R}/\mathcal{S}$ be match-raise-RT-bounded for L by a $c \in \mathbb{N}$. Lemma 5.29 yields an infinite $\xrightarrow{\mathsf{r}}_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})}$ rewrite sequence

$$t_1' \xrightarrow{\mathsf{r}}_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})} t_2' \xrightarrow{\mathsf{r}}_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})} t_3' \xrightarrow{\mathsf{r}}_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})} \cdots$$

starting from $t_1' = \mathsf{lift}_0(t_1)$ such that $\mathsf{base}(t_i') = t_i$ for all $i \geqslant 1$. Because $\mathcal{R}/\mathcal{S}$ is match-raise-RT-bounded for L by c, the height of every function symbol occurring in a term in the lifted sequence is at most c. Hence the employed rewrite rules in the derivation emanating from t_1' must come from $\mathsf{match\text{-}RT}_c(\mathcal{R}/\mathcal{S})$. With help of Lemmata 5.25 and 5.30, transitivity of $\succeq^c_{\mathsf{mul}}$, and compatibility of $\succ^c_{\mathsf{mul}}$ and $\succeq^c_{\mathsf{mul}}$ we deduce that $t_i' \succ^c_{\mathsf{mul}} t_{i+1}'$ for all $i \geqslant 1$. (Note that Lemma 5.25 requires that $\mathcal{R}/\mathcal{S}$ is non-duplicating.) However, this is excluded because $<_{\mathbb{N}}$ is well-founded on $\{0, \ldots, c\}$ and hence $\succ^c_{\mathsf{mul}}$ is well-founded on $\mathcal{T}(\mathcal{F}_{\{0,\ldots,c\}}, \mathcal{V})$.

To prove the second part of the theorem, consider an arbitrary (terminating) rewrite sequence

$$u \to_{\mathcal{R}/\mathcal{S}} u_1 \to_{\mathcal{R}/\mathcal{S}} \cdots \to_{\mathcal{R}/\mathcal{S}} u_m$$

with $u \in L$. This rewrite sequence can be lifted to a $\xrightarrow{\mathsf{r}}_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})}$-sequence of the same length

$$u' \xrightarrow{\mathsf{r}}_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})} u_1' \xrightarrow{\mathsf{r}}_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})} \cdots \xrightarrow{\mathsf{r}}_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})} u_m'$$

similar as before such that $u' = \mathsf{lift}_0(u)$ and $u_i' \succ^c_{\mathsf{mul}} u_{i+1}'$ for all $i \in \{0, \ldots, m-1\}$. Here $u_0' = u'$ and $c \in \mathbb{N}$ such that the relative TRS $\mathcal{R}/\mathcal{S}$ is match-raise-RT-bounded for L by c. Similar as in the proof of Theorem 5.20 we can conclude that the length of the $\mathcal{R}/\mathcal{S}$-rewrite sequence starting at the term u is bounded by $\|u\| \cdot (k+1)^c$ where k is the maximal number of function symbols occurring in some right-hand side in $\mathcal{R} \cup \mathcal{S}$; just replace $\succ_{\mathsf{mul}}$ by $\succ^c_{\mathsf{mul}}$. □

5.5.4. Automation

To automatically prove that a given relative TRS is match(-raise)-RT-bounded for some language L we use (quasi-deterministic, raise-consistent, and) compatible tree automata. Here a tree automaton $\mathcal{A}$ is said to be *compatible* with a relative TRS $\mathcal{R}/\mathcal{S}$ and a language L if $\mathcal{A}$ is compatible with $\mathcal{R} \cup \mathcal{S}$ and L.

Lemma 5.32. *Let $\mathcal{R}/\mathcal{S}$ be a left-linear relative TRS, L a language, and $c \in \mathbb{N}$. Let $\mathcal{A}$ be a tree automaton. If $\mathcal{A}$ is compatible with the relative TRS $\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})$ and $\mathsf{lift}_0(L)$ such that the height of each function symbol occurring in transitions in $\mathcal{A}$ is at most c then $\mathcal{R}/\mathcal{S}$ is match-RT-bounded for L.*

Proof. Easy consequence of Definition 5.23 and the fact that compatible tree automata are closed under left-linear rewriting. □

In case of non-left-linear TRSs we obtain the following result.

Lemma 5.33. *Let $\mathcal{R}/\mathcal{S}$ be a relative TRS, L a language, and $c \in \mathbb{N}$. Let $\mathcal{A}$ be a quasi-deterministic and raise-consistent tree automaton. If $\mathcal{A}$ is compatible with $\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})$ and $\mathsf{lift}_0(L)$ such that the height of each function symbol occurring in transitions in $\mathcal{A}$ is at most c then $\mathcal{R}/\mathcal{S}$ is match-raise-RT-bounded for L.*

Proof. Straightforward by using the fact that quasi-deterministic, raise-consistent and compatible tree automata are closed under rewriting. □

To prove that a relative TRS $\mathcal{R}/\mathcal{S}$ is match(-raise)-RT-bounded for a set of terms L we construct a (quasi-deterministic and raise-consistent) tree automaton $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ that is compatible with the rewrite rules of $\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})$ and $\mathsf{lift}_0(L)$. Since the set $\rightarrow^*_{\mathsf{match\text{-}RT}^c(\mathcal{R}/\mathcal{S})}(\mathsf{lift}_0(L))$ need not be regular, even for left-linear $\mathcal{R}$ and $\mathcal{S}$ and regular L (see [55]) we cannot hope to give an exact automaton construction. The general idea [53, 55] is to look for violations of the compatibility requirement: $l\sigma \rightarrow^*_\Delta q$ ($l\sigma \rightarrow^*_{\Delta_d} q$) and $r\sigma \not\rightarrow^*_\Delta q$ for some rewrite rule $l \rightarrow r$, state substitution $\sigma\colon \mathsf{Var}(l) \rightarrow Q$ ($\sigma\colon \mathsf{Var}(l) \rightarrow Q_d$), and state $q \in Q$ ($q \in Q_d$). Then we add new states and transitions to the current automaton to ensure $r\sigma \rightarrow^*_\Delta q$. After $r\sigma \rightarrow^*_\Delta q$ has been established, we repeat this process until a (quasi-deterministic, raise-consistent, and) compatible automaton is obtained. Note that this may never happen if new states are repeatedly added. To guess an appropriate c we start with $c = 0$. As soon as a new transition $f_d(q_1, \dots, q_n) \rightarrow q$ with $d >_\mathbb{N} c$ is added to the constructed tree automaton, we set $c = d$ and proceed with the construction.

Example 5.34. We show that the relative TRS $\mathcal{R}/\mathcal{S}$ of Example 5.22 over the signature $\mathcal{F} = \{\mathsf{nil}, \mathsf{cons}, \mathsf{rev}, \mathsf{rev'}\}$ is match-RT-bounded for $\mathcal{T}(\mathcal{F})$ by constructing a compatible tree automaton. As starting point we consider the initial tree automaton

$$\mathsf{nil}_0 \rightarrow 1 \qquad \mathsf{cons}_0(1,1) \rightarrow 1 \qquad \mathsf{rev}_0(1) \rightarrow 1 \qquad \mathsf{rev}'_0(1,1) \rightarrow 1$$

which accepts all ground terms over the enriched signature $\mathsf{lift}_0(\mathcal{F})$. The first compatibility violation we consider is caused by the rewrite rule $\mathsf{rev}_0(x) \to_{\mathsf{match}(\mathcal{R})} \mathsf{rev}'_1(x, \mathsf{nil}_1)$. We have $\mathsf{rev}_0(1) \to 1$ but not $\mathsf{rev}'_1(1, \mathsf{nil}_1) \to^* 1$. To solve this violation we add the transitions $\mathsf{nil}_1 \to 2$ and $\mathsf{rev}'_1(1, 2) \to 1$. The compatibility violation caused by the rewrite rule $\mathsf{rev}'_1(\mathsf{nil}_0, y) \to_{\mathsf{match\text{-}RT}^1(\mathcal{S})} y$ and the derivation $\mathsf{rev}'_1(\mathsf{nil}_0, 2) \to^* 1$ is solved by adding the transition $2 \to 1$. Note that we are currently using $\mathsf{match\text{-}RT}^1(\mathcal{S})$ because the maximal height of a function symbol occurring in the underlying tree automaton is 1. Next we consider the compatibility violation $\mathsf{rev}'_1(\mathsf{cons}_0(1, 1), 2) \to^* 1$ but $\mathsf{rev}'_1(1, \mathsf{cons}_1(1, 2)) \not\to^* 1$ induced by the rule $\mathsf{rev}'_1(\mathsf{cons}_0(x, y), z) \to_{\mathsf{match\text{-}RT}^1(\mathcal{S})} \mathsf{rev}'_1(y, \mathsf{cons}_1(x, z))$. In order to ensure $\mathsf{rev}'_1(1, \mathsf{cons}_1(1, 2)) \to^* 1$ we reuse the transition $\mathsf{rev}'_1(1, 2) \to 1$ and add the new transition $\mathsf{cons}_1(1, 2) \to 2$. Finally, $\mathsf{rev}'_0(\mathsf{cons}_1(1, 2), 1) \to^* 1$ and $\mathsf{rev}'_0(\mathsf{cons}_1(x, y), z) \to_{\mathsf{match\text{-}RT}^1(\mathcal{S})} \mathsf{rev}'_1(y, \mathsf{cons}_1(x, z))$ give rise to the transition $\mathsf{cons}_1(1, 1) \to 2$. After this step, the obtained tree automaton is compatible with $\mathsf{match\text{-}RT}^1(\mathcal{R}/\mathcal{S})$. Hence $\mathcal{R}/\mathcal{S}$ is match-RT-bounded for $\mathcal{T}(\mathcal{F})$ by 1. Due to Theorem 5.26 we can conclude that $\mathcal{R}/\mathcal{S}$ admits at most linear complexity. We remark that the ordinary match-bound technique (Theorem 5.20) fails on $\mathcal{R}/\mathcal{S}$ because $\mathcal{R} \cup \mathcal{S}$ induces quadratic complexity:

$$\begin{aligned}\mathsf{rev}^n(x)\sigma^m &\to \mathsf{rev}^{n-1}(\mathsf{rev}'(x, \mathsf{nil}))\sigma^m \to^m \mathsf{rev}^{n-1}(\mathsf{rev}'(\mathsf{nil}, x))\sigma^m \\ &\to \mathsf{rev}^{n-1}(x)\sigma^m \to^{(n-1)(m+2)} x\sigma^m\end{aligned}$$

with $\sigma = \{x \mapsto \mathsf{cons}(y, x)\}$) for all $n, m \geqslant 1$.

5.6. Assessment

In this section we compare the complexity proving power of the direct and the modular setting on a theoretical level. Gains in power in practice are reported in Section 5.8. In the first part of this section we show that for TMIs of dimension one, i.e. SLIs, in theory both approaches are equivalent but in the general case the modular setting allows TMIs of smaller dimensions to succeed. Since the dimension of the TMI corresponds to the degree of the polynomial bound the modular setting allows to establish tighter bounds. The second part of this section shows that the modular setting is strictly more powerful than the direct one, i.e., there are systems where the modular setting admits a complexity proof but all involved methods cannot succeed on its own in the direct setting. To make the presentation easier we assume the original problems to be standard (in contrast to relative) TRSs. This has no effect on the results. The next lemma states that for SLIs in theory there is no difference in power between the two settings.

Lemma 5.35. *Let $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ be a TRS. There exists an SLI $\mathcal{M}$ compatible with $\mathcal{R}$ if and only if there exist SLIs $\mathcal{M}_1$ and $\mathcal{M}_2$ such that $\mathcal{M}_1$ is compatible with $\mathcal{R}_1/\mathcal{R}_2$ and $\mathcal{M}_2$ is compatible with $\mathcal{R}_2/\mathcal{R}_1$.*

Proof. The implication from left to right obviously holds since $\mathcal{M}$ is a suitable candidate for $\mathcal{M}_1$ and $\mathcal{M}_2$. For the reverse direction we construct an SLI $\mathcal{M}$ compatible with $\mathcal{R}$ based on the SLIs $\mathcal{M}_1$ and $\mathcal{M}_2$. To this end let $f_{\mathcal{M}_1}(x_1, \ldots, x_m) = x_1 + \cdots + x_m + f_1$ and $f_{\mathcal{M}_2}(x_1, \ldots, x_m) = x_1 + \cdots + x_m + f_2$. It is straightforward to check that $f_{\mathcal{M}}(x_1, \ldots, x_m) = x_1 + \cdots + x_m + f_1 + f_2$ for any $f \in \mathcal{F}$ yields an SLI $\mathcal{M}$ compatible with $\mathcal{R}$. □

Due to Theorem 5.6 the complexity is not affected when using the modular setting. Hence when using SLIs in theory both approaches can prove the same bounds. But experiments in Section 5.8 show that in practice proofs are easier to find in the modular setting since, e.g., the coefficients of the interpretations can be chosen smaller (cf. the proof of Lemma 5.35). If TMIs of larger dimensions are applied then just the only-if direction of Lemma 5.35 holds. This is shown with the help of the next example.

Example 5.36. Consider the TRS $\mathcal{R}$ (Strategy_removed_AG01/#4.21) consisting of the rules:

$$1\colon \mathsf{f}(\mathsf{1}) \to \mathsf{f}(\mathsf{g}(\mathsf{1})) \quad 2\colon \mathsf{f}(\mathsf{f}(x)) \to \mathsf{f}(x) \quad 3\colon \mathsf{g}(\mathsf{0}) \to \mathsf{g}(\mathsf{f}(\mathsf{0})) \quad 4\colon \mathsf{g}(\mathsf{g}(x)) \to \mathsf{g}(x)$$

The TMIs $\mathcal{M}_1$ and $\mathcal{M}_2$

$$\mathsf{f}_{\mathcal{M}_1}(\vec{x}) = \begin{pmatrix}1 & 1\\ 0 & 0\end{pmatrix}\vec{x} + \begin{pmatrix}0\\1\end{pmatrix} \quad \mathsf{g}_{\mathcal{M}_1}(\vec{x}) = \begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix}\vec{x} \quad \mathsf{0}_{\mathcal{M}_1} = \begin{pmatrix}0\\0\end{pmatrix} \quad \mathsf{1}_{\mathcal{M}_1} = \begin{pmatrix}0\\1\end{pmatrix}$$
$$\mathsf{g}_{\mathcal{M}_2}(\vec{x}) = \begin{pmatrix}1 & 1\\ 0 & 0\end{pmatrix}\vec{x} + \begin{pmatrix}0\\1\end{pmatrix} \quad \mathsf{f}_{\mathcal{M}_2}(\vec{x}) = \begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix}\vec{x} \quad \mathsf{1}_{\mathcal{M}_2} = \begin{pmatrix}0\\0\end{pmatrix} \quad \mathsf{0}_{\mathcal{M}_2} = \begin{pmatrix}0\\1\end{pmatrix}$$

establish quadratic upper bounds on the derivational complexity of the systems $\{3,4\}/\{1,2\}$ and $\{1,2\}/\{3,4\}$, respectively. Theorem 5.6 establishes a quadratic upper bound for $\mathcal{R}$.

Although for the TRS in Example 5.36 TMIs of dimension two could establish a quadratic upper bound on the derivational complexity in the modular setting, they cannot do so in the direct setting because of the next lemma. (We remark that there exist TMIs of dimension three that are compatible with this TRS).

Lemma 5.37. *The TRS* Strategy_removed_AG01/#4.21 *does not admit a TMI of dimension two compatible with it.*

Proof. To address all possible interpretations we extracted the set of constraints that represent a TMI of dimension two compatible with the TRS. MiniSmt [208] can detect unsatisfiability of these constraints. Details of this proof can be found at the web site in Footnote 6 on page 122. □

The next result shows that any direct proof transfers into the modular setting without increasing the bounds on the complexity.

Lemma 5.38. *Let $\succ$ be a finitely branching rewrite relation and let $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ be a TRS compatible with $\succ$. Then there exist complexity pairs $(\succ_1, \succeq_1)$ and $(\succ_2, \succeq_2)$ which are compatible with the relative TRSs $\mathcal{R}_1/\mathcal{R}_2$ and $\mathcal{R}_2/\mathcal{R}_1$, respectively. Furthermore for any language L we have $\mathsf{cp}_L(n, \succ) = \Theta(\mathsf{cp}_L(n, \succ_1) + \mathsf{cp}_L(n, \succ_2))$.*

Proof. Fix i. Let $(\succ_i, \succeq_i)$ be $(\succ, =)$. It is easy to see that $(\succ, =)$ is a complexity pair because $\succ$ and $=$ are compatible rewrite relations. It remains to show that for any term $t \in L$ we have $\mathsf{cp}_L(n, \succ) = \Theta(\mathsf{cp}_L(n, \succ_1) + \mathsf{cp}_L(n, \succ_2))$. To this end we observe that $\mathsf{dh}(t, \succ_1) + \mathsf{dh}(t, \succ_2) = 2 \cdot \mathsf{dh}(t, \succ)$ for all terms $t \in L$. Basic properties of the $\mathcal{O}$-notation yield the desired result. □

Due to Example 5.36 and Lemmata 5.37 and 5.38 we obtain that the modular setting allows to use TMIs of smaller dimensions than the direct one, which allows to establish tighter bounds. The next example (together with Lemma 5.38) shows that in theory the modular complexity setting is strictly more powerful than the direct one since it allows to combine different criteria to establish an upper complexity bound while any method on its own cannot succeed.

Example 5.39. Consider the TRS $\mathcal{R}$ (Transformed_CSR_04/Ex16_Luc06_GM) consisting of the rules:

$$\begin{array}{llll}
1\colon \mathsf{c} \to \mathsf{a} & 3\colon \mathsf{mark}(\mathsf{a}) \to \mathsf{a} & 5\colon \mathsf{g}(x,y) \to \mathsf{f}(x,y) & \\
2\colon \mathsf{c} \to \mathsf{b} & 4\colon \mathsf{mark}(\mathsf{b}) \to \mathsf{c} & 6\colon \mathsf{g}(x,x) \to \mathsf{g}(\mathsf{a},\mathsf{b}) & 7\colon \mathsf{mark}(\mathsf{f}(x,y)) \to \mathsf{g}(\mathsf{mark}(x),y)
\end{array}$$

The following SLI $\mathcal{M}$ with $\mathsf{a}_\mathcal{M} = 0$, $\mathsf{b}_\mathcal{M} = 0$, $\mathsf{c}_\mathcal{M} = 1$, $\mathsf{f}_\mathcal{M}(x,y) = x + y$, $\mathsf{g}_\mathcal{M}(x,y) = x + y + 1$, and $\mathsf{mark}_\mathcal{M}(x) = x + 2$ allows Corollary 5.14 to transform the TRS $\mathcal{R}$ into the relative TRS $\{6,7\}/\{1,2,3,4,5\}$. This problem can be split according to Theorem 5.6 into the two relative TRSs $\{6\}/\{1,2,3,4,5,7\}$ and $\{7\}/\{1,2,3,4,5,6\}$. Match-bounds (Theorem 5.31) can show a linear upper bound on the first problem. The following TMI $\mathcal{M}$

$$\mathsf{a}_\mathcal{M} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad \mathsf{f}_\mathcal{M}(\vec{x}, \vec{y}) = \vec{x} + \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \vec{y} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad \mathsf{mark}_\mathcal{M}(\vec{x}) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \vec{x}$$

where $\mathsf{a}_\mathcal{M} = \mathsf{b}_\mathcal{M} = \mathsf{c}_\mathcal{M}$ and $\mathsf{f}_\mathcal{M}(\vec{x},\vec{y}) = \mathsf{g}_\mathcal{M}(\vec{x},\vec{y})$ gives a quadratic upper bound on the second relative TRS, establishing a quadratic upper bound on the derivational complexity of $\mathcal{R}$. The quadratic bound is tight as $\mathcal{R}$ admits derivations

$$\mathsf{mark}^n(x)\sigma^m \to^m \mathsf{mark}^{n-1}(x)\tau^m\gamma \to^m \mathsf{mark}^{n-1}(x)\sigma^m\gamma \to^{2m(n-1)} x\sigma^m\gamma^n$$

of length $2mn$ where $\sigma = \{x \mapsto \mathsf{f}(x,y)\}$, $\tau = \{x \mapsto \mathsf{g}(x,y)\}$, and $\gamma = \{x \mapsto \mathsf{mark}(x)\}$. Last but not least we remark that none of the involved techniques can establish an upper bound on its own. In case of match-bounds this follows from the fact that $\mathcal{R}$ admits quadratic derivational complexity. The same reason also holds for Corollary 5.14 because SLIs induce linear complexity bounds. Finally, TMIs fail because they cannot orient the rewrite rule $\mathsf{g}(x,x) \to \mathsf{g}(\mathsf{a},\mathsf{b})$.

Hence we obtain the following corollary.

Corollary 5.40. *The modular complexity setting is strictly more powerful than the direct one.*

Proof. By Lemma 5.38 and Example 5.39. □

Next we consider the TRS Zantema_04/z086. The question about the derivational complexity of it has been stated as problem #105 on the RTA LooP.[4]

Example 5.41. Consider the TRS $\mathcal{R}$ (Zantema_04/z086) consisting of the rules:

$$1\colon \mathsf{a}(\mathsf{a}(x)) \to \mathsf{c}(\mathsf{b}(x)) \qquad 2\colon \mathsf{b}(\mathsf{b}(x)) \to \mathsf{c}(\mathsf{a}(x)) \qquad 3\colon \mathsf{c}(\mathsf{c}(x)) \to \mathsf{b}(\mathsf{a}(x))$$

Adian [1] showed that $\mathcal{R}$ admits at most quadratic derivational complexity. Since the proof is based on a low-level reasoning on the structure of $\mathcal{R}$, it is specific to this TRS and challenging for automation. With our approach we cannot prove the quadratic bound on the derivational complexity of $\mathcal{R}$. However, Corollary 5.14 permits to establish some progress. Using an SLI counting a's and b's, it suffices to determine the derivational complexity of $\{3\}/\{1,2\}$. This means that the rule $\mathsf{c}(\mathsf{c}(x)) \to \mathsf{b}(\mathsf{a}(x))$ relative to the other rules dominates the derivational complexity of $\mathcal{R}$. The benefit is that now, e.g., a TMI must only orient one rule strictly and the other two rules weakly (compared to all three rules strictly). It has to be clarified if the relative formulation of the problem can be used to simplify the proof in [1].

The next example shows that although the modular approach often allows to establish lower bounds compared to the direct one, further criteria for splitting TRSs should be investigated.

[4] `http://www.cs.tau.ac.il/~nachum/rtaloop/`

Example 5.42. Consider the TRS $\mathcal{R}$ (SK90/4.30) consisting of the following rules:

$$\begin{array}{lll}
1\colon \mathsf{f}(\mathsf{nil}) \to \mathsf{nil} & 3\colon \mathsf{f}(\mathsf{nil} \circ y) \to \mathsf{nil} \circ \mathsf{f}(y) & 5\colon \mathsf{f}((x \circ y) \circ z) \to \mathsf{f}(x \circ (y \circ z)) \\
2\colon \mathsf{g}(\mathsf{nil}) \to \mathsf{nil} & 4\colon \mathsf{g}(x \circ \mathsf{nil}) \to \mathsf{g}(x) \circ \mathsf{nil} & 6\colon \mathsf{g}(x \circ (y \circ z)) \to \mathsf{g}((x \circ y) \circ z)
\end{array}$$

In [126] a TMI compatible with $\mathcal{R}$ of dimension four is given showing that the derivational complexity is bounded by a polynomial of degree four. Using Theorem 5.6 with TMIs of dimension three yields a cubic upper bound, i.e., the TMI $\mathcal{M}_1$

$$\circ_{\mathcal{M}_1}(\vec{x},\vec{y}) = \begin{pmatrix}1&0&0\\0&0&1\\0&0&1\end{pmatrix}\vec{x} + \vec{y} + \begin{pmatrix}0\\0\\1\end{pmatrix} \qquad \mathsf{f}_{\mathcal{M}_1}(\vec{x}) = \begin{pmatrix}1&1&0\\0&0&1\\0&0&1\end{pmatrix}\vec{x} + \begin{pmatrix}0\\1\\0\end{pmatrix}$$

$$\mathsf{g}_{\mathcal{M}_1}(\vec{x}) = \begin{pmatrix}1&0&0\\0&0&1\\0&0&1\end{pmatrix}\vec{x} \qquad \mathsf{nil}_{\mathcal{M}_1} = \begin{pmatrix}1\\1\\1\end{pmatrix}$$

yields a cubic upper bound on $\{1,2,3\}/\{4,5,6\}$. So does the TMI $\mathcal{M}_2$ with

$$\mathsf{f}_{\mathcal{M}_2}(\vec{x}) = \mathsf{g}_{\mathcal{M}_1}(\vec{x}) \quad \mathsf{g}_{\mathcal{M}_2}(\vec{x}) = \mathsf{f}_{\mathcal{M}_1}(\vec{x}) \quad \circ_{\mathcal{M}_2}(\vec{x},\vec{y}) = \circ_{\mathcal{M}_1}(\vec{y},\vec{x}) \quad \mathsf{nil}_{\mathcal{M}_2} = \mathsf{nil}_{\mathcal{M}_1}$$

for $\{4,5,6\}/\{1,2,3\}$. Our approach enables showing a lower complexity than [126] but the derivational complexity of $\mathcal{R}$ is quadratic (see [126]). The quadratic lower bound is justified as $\mathcal{R}$ admits derivations

$$\mathsf{f}^n(x)\sigma^m \to^n \mathsf{nil} \circ \mathsf{f}^n(x)\sigma^{m-1} \to^n \cdots \to^n x\sigma^m\tau^n$$

of length nm where $\sigma = \{x \mapsto \mathsf{nil} \circ x\}$ and $\tau = \{x \mapsto \mathsf{f}(x)\}$. We stress that the recent approach in [187] allows to establish a quadratic upper bound. For a comment on the integration of this method into our setting we refer to Section 5.9.

5.7. Implementation

In Section 5.7.1 we first show how the various theorems from the previous sections can be implemented to obtain *some* complexity proof. Afterwards Section 5.7.2 is concerned with lowering the bounds starting from an existing complexity proof.

5.7.1. Establishing Bounds

To estimate the complexity of a TRS $\mathcal{R}$ with respect to a language L, we first transform $\mathcal{R}$ into the relative TRS $\mathcal{R}/\varnothing$. Obviously $\mathsf{cp}_L(n, \to_{\mathcal{R}}) = \mathsf{cp}_L(n, \to_{\mathcal{R}/\varnothing})$. If the input already is a relative TRS this step is omitted. Afterwards for a relative TRS $\mathcal{R}/\mathcal{S}$ we try to establish a bound on the complexity of $\mathcal{R}_2/(\mathcal{R}_1 \cup \mathcal{S})$ with respect to L for some $\mathcal{R}_1$, $\mathcal{R}_2$ with $\mathcal{R}_1 = \mathcal{R} \backslash \mathcal{R}_2$ and continue with the relative TRS $\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})$. This step is executed repeatedly until the remaining problem equals $\varnothing/(\mathcal{R} \cup \mathcal{S})$. Then the complexity of $\mathcal{R}/\mathcal{S}$ with respect to L is obtained by summing up all intermediate bounds. In order to establish a maximal number of complexity proofs we run all techniques from Sections 5.4 and 5.5 in parallel and the first technique that can shift some rules is used to achieve progress.

Note that the procedure sketched above contains an implicit application of Theorem 5.6, i.e., some method immediately proves a bound for $\mathcal{R}_2/(\mathcal{R}_1 \cup \mathcal{S})$ and leaves $\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S})$ as open proof obligation. In contrast to an explicit application of Theorem 5.6, here the method that establishes the bound on $\mathcal{R}_2/(\mathcal{R}_1 \cup \mathcal{S})$ can select the decomposition of $\mathcal{R}$ into $\mathcal{R}_1$ and $\mathcal{R}_2$ which is beneficial for performance. As an immediate consequence, proof trees degenerate to lists (cf. Example 5.50). In the following we describe the presented approach more formal and refer to it as the *complexity framework*.

Definition 5.43. A *complexity problem* (CP problem for short) is a pair $(\mathcal{R}/\mathcal{S}, L)$ consisting of a relative TRS $\mathcal{R}/\mathcal{S}$ and a language L.

To operate on CP problems so called *complexity processors* are used. Similar as in the dependency pair framework we distinguish between sound and complete processors. Here sound complexity processors are used to prove an upper bound on the complexity of a given CP problem whereas complete complexity processors are applied to derive lower bounds on the complexity.

Definition 5.44. A *complexity processor* (CP processor for short) is a function that takes a CP problem $(\mathcal{R}/\mathcal{S}, L)$ as input and as output it returns a set of pairs $\bigcup_{1 \leqslant i \leqslant m}\{((\mathcal{R}_i/\mathcal{S}_i, L_i), f_i)\}$.[5] Here $(\mathcal{R}_i/\mathcal{S}_i, L_i)$ is a complexity problem and $f_i \colon \mathbb{N} \to \mathbb{N}$ for each $1 \leqslant i \leqslant m$. A complexity processor is *sound* if

$$\mathsf{cp}_L(n, \to_{\mathcal{R}/\mathcal{S}}) = \mathcal{O}(f_1(n) + \cdots + f_m(n) + \mathsf{cp}_{L_1}(n, \to_{\mathcal{R}_1/\mathcal{S}_1}) + \cdots + \mathsf{cp}_{L_m}(n, \to_{\mathcal{R}_m/\mathcal{S}_m}))$$

and it is called *complete* if

$$\mathsf{cp}_L(n, \to_{\mathcal{R}/\mathcal{S}}) = \Omega(f_1(n) + \cdots + f_m(n) + \mathsf{cp}_{L_1}(n, \to_{\mathcal{R}_1/\mathcal{S}_1}) + \cdots + \mathsf{cp}_{L_m}(n, \to_{\mathcal{R}_m/\mathcal{S}_m}))$$

holds.

In the sequel zero denotes the constant zero function, i.e., $\mathsf{zero} \colon \mathbb{N} \to \mathbb{N}$ with $\mathsf{zero}(n) = 0$. Next we list some CP processors that can be derived from the previous sections. The first one is based on complexity pairs and can, e.g., be implemented by Theorems 5.8 and 5.10.

[5]For reasons of readability we write pairs $((\mathcal{R}_i/\mathcal{S}_i, L_i), f_i)$ as triples $(\mathcal{R}_i/\mathcal{S}_i, L_i, f_i)$.

Theorem 5.45. *The CP processor*

$$(\mathcal{R}/\mathcal{S}, L) \mapsto \begin{cases} \{(\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S}), L, f)\} & \text{if } \mathcal{R}_2/(\mathcal{R}_1 \cup \mathcal{S}) \text{ is compatible} \\ & \text{with a complexity pair } (\succ, \succsim) \\ \{(\mathcal{R}/\mathcal{S}, L, \mathsf{zero})\} & \text{otherwise} \end{cases}$$

where $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$*, and* $f(n) = \mathsf{cp}_L(n, \succ)$ *is sound.*

Proof. Follows from Corollary 5.3 and Theorem 5.6. □

The above processor is implemented by demanding that all rules in $\mathcal{R} \cup \mathcal{S}$ are weakly oriented while at least one rule in $\mathcal{R}$ is oriented strictly. Hence the decomposition of $\mathcal{R}$ into $\mathcal{R}_1$ and $\mathcal{R}_2$ is performed automatically. The next CP processor requires a mild condition on $\mathcal{R}_1$ only. Again the decomposition of $\mathcal{R}$ into $\mathcal{R}_1$ and $\mathcal{R}_2$ is performed automatically since in the implementation we just demand that the $\mathcal{S}$-rules are weakly oriented while the $\mathcal{R}$-rules may increase by a constant factor and at least one of the rules in $\mathcal{R}$ is oriented strictly.

Theorem 5.46. *The CP processor*

$$(\mathcal{R}/\mathcal{S}, L) \mapsto \begin{cases} \{(\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S}), L, f)\} & \text{if } \mathcal{M} \text{ is a matrix interpretation with} \\ & \text{constant growth, } \mathcal{R}_1 \subseteq \succsim_{\mathcal{M}}^{\mathtt{ncp}}\text{, and} \\ & \mathcal{R}_2/\mathcal{S} \text{ is compatible with } \mathcal{M} \\ \{(\mathcal{R}/\mathcal{S}, L, \mathsf{zero})\} & \text{otherwise} \end{cases}$$

where $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$*, and* $f(n) = n$ *is sound.*

Proof. Follows from Theorem 5.6 and Theorem 5.18. □

The next CP processor is based on match-bounds.

Theorem 5.47. *The CP processor*

$$(\mathcal{R}/\mathcal{S}) \mapsto \begin{cases} \{(\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S}), L, f)\} & \text{if } \mathcal{R}_2/(\mathcal{R}_1 \cup \mathcal{S}) \text{ is} \\ & \text{linear and match-RT-bounded for } L \text{ or} \\ & \text{non-duplicating and match-raise-RT-bounded} \\ & \text{for } L \\ \{(\mathcal{R}/\mathcal{S}, L, \mathsf{zero})\} & \text{otherwise} \end{cases}$$

where $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$*,* $\mathcal{R}_2$ *is non-collapsing, and* $f(n) = n$ *is sound.*

Proof. Follows from Theorems 5.6, 5.26, and 5.31. □

The above processor is implemented by considering for any rule $l \to r \in \mathcal{R}$ the decompositions $\mathcal{R}_2 = \{l \to r\}$ and $\mathcal{R}_1 = \mathcal{R} \setminus \mathcal{R}_2$ in parallel. The next CP processor we present is not implemented for finding a bound (cf. the discussion at the beginning of the section) but very suitable to tighten existing bounds (see Section 5.7.2).

Theorem 5.48. *The CP processor*

$$(\mathcal{R}/\mathcal{S}, L) \mapsto \{(\mathcal{R}_1/(\mathcal{R}_2 \cup \mathcal{S}), L, \mathsf{zero}), (\mathcal{R}_2/(\mathcal{R}_1 \cup \mathcal{S}), L, \mathsf{zero})\}$$

where $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ is sound and complete.

Proof. By Theorem 5.6. □

Finally, the main theorem states that the CP framework may be applied to complexity analysis. We say that P is a complexity proof for a relative TRS $\mathcal{R}/\mathcal{S}$ and a language L if all leaves in P are of the shape $\varnothing/(\mathcal{R} \cup \mathcal{S})$.

Theorem 5.49. *Let $\mathcal{R}/\mathcal{S}$ be a relative TRS and L be a language. Let P be a complexity proof for $\mathcal{R}/\mathcal{S}$ and L and $f_1, \ldots, f_m$ be the complexities in this proof. If all CP processors in P are sound then $\mathsf{cp}_L(n, \to_{\mathcal{R}/\mathcal{S}}) = \mathcal{O}(f_1(n) + \cdots + f_m(n))$. If all CP processors in P are complete then $\mathsf{cp}_L(n, \to_{\mathcal{R}/\mathcal{S}}) = \Omega(f_1(n) + \cdots + f_m(n))$.*

Proof. By Definition 5.44 as well as basic properties of $\mathcal{O}$-notation. □

We conclude the section with an (abstract) example which illustrates the behavior of the complexity framework.

Example 5.50. Consider the TRS $\mathcal{R}$ of Example 5.7 on page 91 with the complexity proof depicted in Figure 5.2. After transforming $\mathcal{R}$ into the relative TRS $\mathcal{R}/\varnothing$ the CP processor of Theorem 5.45 is applied twice. First the (derivational) complexity of the relative TRS $\{1,3,5\}/\{2,4\}$ is estimated by a polynomial of degree five. As a consequence, the rules 1, 3, and 5 are moved into the relative component yielding a CP problem consisting of the relative TRS $\{2,4\}/\{1,3,5\}$. After that the (derivational) complexity of $\{2,4\}/\{1,3,5\}$ is estimated by a quadratic bound. Since the remaining CP problem is of the shape $\varnothing/\mathcal{R}$ according to Theorem 5.49 the (derivational) complexity of $\mathcal{R}$ is at most quintic.

5.7.2. Tightening Bounds

In contrast to termination, which is a plain YES/NO question, complexity corresponds to an optimization problem. Hence automated tools should try to establish as tight bounds as possible. In the direct setting all complexity methods can be executed in parallel and after a fixed amount of time the tightest bound is reported. The next example shows such a case.

$$\begin{tikzcd}
\mathcal{R} \arrow[d, "\mathcal{O}(1)"] \\
\mathcal{R}/\varnothing \arrow[d, "\mathcal{O}(n^5)"] \\
\{2,4\}/\{1,3,5\} \arrow[d, "\mathcal{O}(n^2)"] \\
\varnothing/\mathcal{R}
\end{tikzcd}$$

Figure 5.2.: Sequential complexity proof.

Example 5.51. Consider the TRS $\mathcal{R}_{\mathsf{bits}}$ (nontermin/AG01/#4.28) consisting of the following five rules:

$$\begin{array}{rrclrrcl}
1\colon & \mathsf{half}(\mathsf{0}) & \to & \mathsf{0} & 4\colon & \mathsf{bits}(\mathsf{0}) & \to & \mathsf{0} \\
2\colon & \mathsf{half}(\mathsf{s}(\mathsf{0})) & \to & \mathsf{0} & 5\colon & \mathsf{bits}(\mathsf{s}(x)) & \to & \mathsf{s}(\mathsf{bits}(\mathsf{half}(\mathsf{s}(x)))) \\
3\colon & \mathsf{half}(\mathsf{s}(\mathsf{s}(x))) & \to & \mathsf{s}(\mathsf{half}(x)) & & & &
\end{array}$$

For this TRS the complexity analyzer $\mathsf{C}\!\!\!\mathsf{a}\mathsf{T}$ (cf. Section 5.8) finds a proof by root-labeling followed by a TMI of dimension two, establishing a quadratic upper bound within five seconds. However, after 90 seconds the tool finds the following AMI $\mathcal{A}$ that shows a linear upper bound:

$$\mathsf{bits}_{\mathcal{A}}(\vec{x}) = \begin{pmatrix} 0 & 1 & 2 \\ 0 & 4 & 5 \\ 0 & 6 & 7 \end{pmatrix} \vec{x} \qquad \mathsf{half}_{\mathcal{A}}(\vec{x}) = \begin{pmatrix} 1 & -\infty & -\infty \\ 1 & -\infty & -\infty \\ 1 & -\infty & -\infty \end{pmatrix} \vec{x}$$

$$\mathsf{s}_{\mathcal{A}}(\vec{x}) = \begin{pmatrix} 1 & 1 & -\infty \\ 7 & 0 & 2 \\ 1 & 6 & 0 \end{pmatrix} \vec{x} \qquad \mathsf{0}_{\mathcal{A}} = \begin{pmatrix} 0 \\ 0 \\ -\infty \end{pmatrix}$$

So, whenever the tool is allowed more than 90 seconds the linear bound can be reported and if the user sets the global timeout to less, then still the quadratic bound can be output.

In the modular setting this simple idea does not work because two problems emerge. The first problem is that the tool does not know how much time it may spend in a single proof step. If it spends too much then it may not finish the proof within the global time limit and if it spends too little then it can miss a low bound. The second problem is that in the modular setting separate criteria may make statements about the complexity of different rules. The question is then to identify the *better* bound. The next example demonstrates this scenario.

Example 5.52. Consider the TRSs

$$1\colon \mathsf{a}(\mathsf{b}(x)) \to \mathsf{b}(\mathsf{a}(x)) \qquad\qquad 2\colon \mathsf{g}(x,x) \to \mathsf{g}(\mathsf{c},\mathsf{d})$$

The TMI $\mathcal{M}$ with $\mathsf{g}_{\mathcal{M}}(\vec{x},\vec{y}) = \vec{x} + \vec{y}$ and

$$\mathsf{a}_{\mathcal{M}}(\vec{x}) = \begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}\vec{x} \quad \mathsf{b}_{\mathcal{M}}(\vec{x}) = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\vec{x} + \begin{pmatrix}0\\1\end{pmatrix} \quad \mathsf{c}_{\mathcal{M}}(\vec{x}) = \begin{pmatrix}0\\0\end{pmatrix} \quad \mathsf{d}_{\mathcal{M}}(\vec{x}) = \begin{pmatrix}0\\0\end{pmatrix}$$

establishes a quadratic upper bound on the complexity of $\{1\}/\{2\}$ whereas match-bounds yield a linear upper bound for $\{2\}/\{1\}$. The question now is with which remaining proof obligation ($\{2\}/\{1\}$ or $\{1\}/\{2\}$) the tool should continue. Note that both bounds are tight.

The following idea overcomes both problems: First we establish *some* complexity proof according to the procedure described at the beginning of Section 5.7.1 to obtain a bound for as many systems as possible. Afterwards we *optimize* this bound. The next example shows how the latter works.

Example 5.53. Consider the TRS $\mathcal{R}$ of Example 5.7 with the complexity proof depicted in Figure 5.3a. In this exemplary case one part in this proof, highlighted by a solid box, is overestimated by a cubic upper bound. Hence the complexity of the whole system is at most cubic. We remark that this proof step estimates the complexity of $\{3,5\}/\{1,2,4\}$. Now assume that the cubic bound is not optimal, i.e., there exists a proof (that may be longer and harder to find) that induces a quadratic upper bound on the complexity of $\{3,5\}/\{1,2,4\}$. Then the proof is optimized as illustrated in Figure 5.3b, i.e., $\{1,3,5\}/\{2,4\}$ is split into the problems $\{1\}/\{2,3,4,5\}$ and $\{3,5\}/\{1,2,4\}$ by an application of Theorem 5.6. After that, the proof part of $\{1\}/\{2,3,4,5\}$ is reused in the optimized proof (cf. the dashed boxes in Figure 5.3a and Figure 5.3b) whereas the original proof of $\{3,5\}/\{1,2,4\}$ is replaced by the new one, as indicated by the solid box in Figure 5.3b. Now, the proof in Figure 5.3b establishes a quadratic upper bound on the complexity of $\mathcal{R}$. For completeness we state that the proof in Figure 5.3a can be obtained from the sequential proof tree shown in Figure 5.2 by optimization. To show the procedure on a non-linear proof tree this presentation was chosen.

As the previous example demonstrates the basic idea is to replace single proof steps by new proofs that induce tighter bounds. This procedure is repeated until either the global time limit is reached or none of the bounds can be tightened further. Note that the transformation is sound by Theorem 5.49.

The final example in this section shows that it may be easier to find proofs in the modular setting.

$$\begin{tikzcd}
 & \mathcal{R} \arrow[d, "\mathcal{O}(1)"] & \\
 & \mathcal{R}/\varnothing \arrow[dl, "\mathcal{O}(1)"'] \arrow[dr, "\mathcal{O}(1)"] & \\
\{2,4\}/\{1,3,5\} \arrow[d, "\mathcal{O}(n^2)"'] & & \{1,3,5\}/\{2,4\} \arrow[d, "\mathcal{O}(n^3)"] \\
\varnothing/\mathcal{R} & & \{1\}/\{2,3,4,5\} \arrow[d, "\mathcal{O}(n)"] \\
 & & \varnothing/\mathcal{R}
\end{tikzcd}$$

(a) Initial proof.

$$\begin{tikzcd}
 & \mathcal{R} \arrow[d, "\mathcal{O}(1)"] & & \\
 & \mathcal{R}/\varnothing \arrow[dl, "\mathcal{O}(1)"'] \arrow[dr, "\mathcal{O}(1)"] & & \\
\{2,4\}/\{1,3,5\} \arrow[d, "\mathcal{O}(n^2)"'] & & \{1,3,5\}/\{2,4\} \arrow[dl, "\mathcal{O}(1)"'] \arrow[dr, "\mathcal{O}(1)"] & \\
\varnothing/\mathcal{R} & \{1\}/\{2,3,4,5\} \arrow[d, "\mathcal{O}(n)"'] & & \{3,5\}/\{1,2,4\} \arrow[d, "\mathcal{O}(n^2)"] \\
 & \varnothing/\mathcal{R} & & \{5\}/\{1,2,3,4\} \arrow[d, "\mathcal{O}(n^2)"] \\
 & & & \varnothing/\mathcal{R}
\end{tikzcd}$$

(b) Optimized proof.

Figure 5.3.: Tightening bounds.

Example 5.54. Recall the TRS from Example 5.51. Corollary 5.14 with an SLI that just counts function symbols allows to transform the initial problem into $\{5\}/\{1,2,3,4\}$. The AMI of dimension three with

$$\mathsf{bits}_{\mathcal{A}}(\vec{x}) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 3 & 2 \\ 0 & 2 & 2 \end{pmatrix} \vec{x} \qquad \mathsf{half}_{\mathcal{A}}(\vec{x}) = \begin{pmatrix} 0 & -\infty & -\infty \\ 0 & -\infty & -\infty \\ 0 & -\infty & -\infty \end{pmatrix} \vec{x}$$

$$\mathsf{s}_{\mathcal{A}}(\vec{x}) = \begin{pmatrix} 0 & -\infty & 0 \\ -\infty & -\infty & 3 \\ 4 & 0 & 0 \end{pmatrix} \vec{x} \qquad \mathsf{0}_{\mathcal{A}} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

allows to show linear derivational complexity of $\mathcal{R}_{\mathsf{bits}}$. Note that CaT finds this interpretation within three seconds whereas it took the tool 90 seconds to find a suitable interpretation for the direct setting.

5.8. Experimental Results

The techniques described in the preceding sections are implemented in the complexity analyzer CaT (freely available from `http://cl-informatik.uibk.ac.at/software/cat`) which is built on top of TTT2 [107], a powerful termination tool for TRSs.

	$\mathcal{O}(n^k)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	time
direct	315	202	234	259	1.7
direct$\star$	315	215	303	312	7.9
modular	334	208	228	261	2.6
modular$\star$	334	221	321	329	10.6
CaT	328	216	310	319	4.2
CaT$\star$	328	219	317	324	11.5

Table 5.1.: Derivational complexity of 1172 TRSs.

Below we report on the experiments[6] we performed. We considered the 2132 TRSs in version 7.0.2 of the TPDB without strategy or theory annotation. The 1172 non-duplicating systems of this collection have been used for experiments with derivational complexity (note that a duplicating system gives rise to at least exponentially long derivations). For runtime complexity we considered the 1339 systems that are not trivial, e.g., where the set of constructor-based terms is not finite and terminating. In this collection there are 910 non-duplicating systems. All tests have been performed on a server equipped with eight dual-core AMD Opteron® processors 885 running at a clock rate of 2.6 GHz and 64 GB of main memory. We remark that similar results have been obtained on a dual-core laptop. If a tool did not report an answer within 60 seconds, its execution was aborted.

As complexity preserving transformations we employ uncurrying [202] for applicative systems whenever it applies and root-labeling [162] in parallel to the base methods. As base methods we use the match-bounds technique as well as TMIs [126, 131] and AMIs [104] of dimensions one to five. The latter two are implemented by bit-blasting arithmetic operations to SAT [44]. All base methods are run in parallel and started upon program execution.

Our results are summarized in Tables 5.1 and 5.2. Here, direct refers to the conventional setting where all rules must be oriented at once whereas modular first transforms a TRS $\mathcal{R}$ into a relative TRS $\mathcal{R}/\varnothing$ before the CP processors from Section 5.7.1 (except Theorem 5.48) are employed. In the tables the postfix $\star$ indicates that after establishing a bound it is tried to be tightened as explained in Section 5.7.2. The columns $\mathcal{O}(n)$, $\mathcal{O}(n^2)$, ..., $\mathcal{O}(n^k)$ give the number of linear, quadratic, ..., polynomial upper bounds that could be established. We also list the average time (in seconds) needed for finding a bound in the last column. For reference we also give the data for the winners of the corresponding categories in the 2010 edition of the termination competition. For derivational complexity this is CaT and for runtime complexity this is TCT [16].

[6]Full details available from `http://cl-informatik.uibk.ac.at/software/cat/10lmcs`.

	$\mathcal{O}(n^k)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	time
direct	372	354	358	365	0.6
direct$\star$	372	355	370	371	1.1
modular	376	358	359	364	0.5
modular$\star$	376	362	374	375	1.2
TCT	354	351	353	354	4.8
TCT(1339)	363	360	362	363	4.8

Table 5.2.: Runtime complexity of 910 TRSs.

Table 5.1 shows the results for derivational complexity. Here the modular approach allows to prove significantly more polynomial bounds (column $\mathcal{O}(n^k)$) and furthermore these bounds are also smaller than for the direct approach (especially if tightening of bounds is used). The modular setting is slower since there typically more proofs are required to succeed. The rows postfixed $\star$ prove that refining bounds is beneficial, especially if all criteria are run in parallel, which is essential to maximize the total number of upper bounds. The 2010 version of CaT did not use tightening of bounds. To maximize the number of low bounds the tool executes criteria that yield larger complexity bounds slightly delayed. This explains why for CaT tightening bounds increases the global performance less compared to direct and modular. On the contrary, CaT misses some proofs compared to modular since (costly) criteria are not executed for up to 60 seconds.

Table 5.2 shows the results for runtime complexity on the 910 TRSs that are non-duplicating and non-trivial. Here, the starting language for the match-bounds technique has been restricted to constructor-based terms, i.e., no defined symbols are allowed below the root. This makes match-bounds a very powerful technique for runtime complexity, explaining the high number of linear bounds.[7] We remark that in contrast to the criteria we employ TCT can also estimate polynomial bounds for duplicating systems based on weak dependency pairs [75, 76]. The row TCT(1339) corresponds to TCT run on all 1339 TRSs in the benchmark for runtime complexity. Hence this row includes duplicating TRSs. For 9 of these TCT can prove a polynomial upper bound.

For further comparison with other tools we refer the reader to the international termination competition (referenced in Footnote 1 on page 86). Since 2008, when the complexity categories have been installed in the termination competition, CaT won the division for derivational complexity every year.

[7]In the 2010 competition CaT proved upper bounds on the derivational complexity also in the category for runtime complexity. This explains why our methods here outperform TCT while in the competition CaT came second in this division.

5.9. Conclusion

In this article we have introduced a modular approach for estimating the complexity of TRSs by considering relative rewriting. We showed how existing criteria (for full rewriting) can be lifted into the relative setting. The modular approach is easy to implement and has been proved strictly more powerful than traditional methods in theory and practice. Since the modular method allows to combine different criteria, typically smaller complexity bounds are achieved than with the direct setting. Furthermore the modular treatment allows to establish bounds for systems where each of the involved basic methods alone fails. Although originally developed for derivational complexity our results directly apply to more restrictive notions of complexity, e.g., runtime complexity (see also below). Finally we remark that our setting allows a more fine-grained complexity analysis, i.e., while traditionally quadratic derivational complexity ensures that any rule is applied at most quadratically often, our approach can make different statements about single rules. Hence even if a proof attempt does not succeed completely, it may highlight the problematic rules.

We remark that complexity proofs using TMIs (for relative rewriting) can be certified with CeTA [176].

As related work we mention [78] which also considers relative rewriting for complexity analysis. However, there the complexity of $\mathcal{R}_1 \cup \mathcal{R}_2$ is investigated by considering $\mathcal{R}_1/\mathcal{R}_2$ and $\mathcal{R}_2$. Hence [78] also gives rise to a modular reasoning but the obtained complexities are typically beyond polynomials. For runtime complexity analysis Hirokawa and Moser [75, 76] consider weak dependency pair steps relative to the usable rules, i.e., $\mathsf{WDP}(\mathcal{R})/\mathsf{UR}(\mathcal{R})$. However, since in the current formulation of weak dependency pairs some complexity might be hidden in the usable rules they do not really obtain a relative problem. As a consequence they can only apply restricted criteria for the usable rules. Note that our approach can directly be used to show bounds on $\mathsf{WDP}(\mathcal{R})/\mathsf{UR}(\mathcal{R})$ by considering $\mathsf{WDP}(\mathcal{R}) \cup \mathsf{UR}(\mathcal{R})$. Due to Corollary 5.14 this problem can be transformed into an (unrestricted) relative problem $\mathsf{WDP}(\mathcal{R})/\mathsf{UR}(\mathcal{R})$ whenever the constraints in [75] are satisfied. Moreover, if somehow the *problematic* usable rules could be determined and shifted into the $\mathsf{WDP}(\mathcal{R})$ component, then this *improved* version of weak dependency pairs corresponds to a relative problem without additional restrictions, admitting further benefit from our contributions.

Recently two approaches were proposed which admit polynomially bounded matrix interpretations going beyond TMIs. While [187] considers weighted automata, in [131] (joint) spectral radius theory is employed. For ease of presentation these criteria have not been considered in this work but since both are based on matrix interpretations, they perfectly suit our modular setting.

For future work we plan to investigate criteria that allow to analyze the complexity of a TRS $\mathcal{R}$ by the complexities of $\mathcal{R}_1$ and $\mathcal{R}_2$ where $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$. We anticipate that results from modularity [135] are helpful for this aim.

Acknowledgments

We thank Johannes Waldmann for directing our attention to Example 5.12 and Martin Avanzini for providing a binary of the 2010 competition version of TCT.

6. Revisiting Matrix Interpretations for Polynomial Derivational Complexity of Term Rewriting

Publication Details

Friedrich Neurauter, Harald Zankl, and Aart Middeldorp, Revisiting Matrix Interpretations for Polynomial Derivational Complexity of Term Rewriting, In Proceedings of the 17th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-17), Lecture Notes in Computer Science (Advanced Research in Computing and Software Science) 6397, pp. 550–564, Springer, 2010.
doi: `10.1007/978-3-642-16242-8_39`

Abstract

Matrix interpretations can be used to bound the derivational complexity of term rewrite systems. In particular, triangular matrix interpretations over the natural numbers are known to induce polynomial upper bounds on the derivational complexity of (compatible) rewrite systems. Using techniques from linear algebra, we show how one can generalize the method to matrices that are not necessarily triangular but nevertheless polynomially bounded. Moreover, we show that our approach also applies to matrix interpretations over the real (algebraic) numbers. In particular, it allows triangular matrix interpretations to infer tighter bounds than the original approach.

6.1. Introduction

Many powerful techniques for establishing termination of term rewrite systems have been developed in the course of time, most of which have been automated successfully, as is evident in the results of the (annual) international competition for termination and complexity tools.[1] Moreover, Hofbauer and Lautemann observe in [77] that "proving termination with one of these specific techniques in general proves more than just the absence of infinite derivations. It turns out that in many cases such a proof implies an upper bound on the maximal length of derivations", which they consider as a natural measure for the complexity of (terminating) term rewrite systems. More precisely, the resulting notion of *derivational complexity* relates the length of a longest derivation to the size of its initial term. For example, polynomial interpretations imply a double-exponential upper bound on the derivational complexity [77]. However, since term rewriting is a model of computation and algorithms of polynomial complexity are widely accepted as *feasible*, one is especially interested in polynomial derivational complexity. But currently only few techniques for establishing feasible upper complexity bounds are known. Commonly, they are stripped-down variants of existing termination techniques. For example, if a term rewrite system can be shown terminating by a matrix interpretation (over the natural numbers) [44, 79] that orients all rewrite rules strictly, then its derivational complexity is at most exponential. However, by restricting the shape of the matrices to upper triangular form, one obtains a method for establishing polynomial derivational complexity [126], where the degree of the polynomial depends on the dimension of the matrices. Using match-bounds [55] or arctic matrix interpretations [103], linear derivational complexity can be inferred.

In this paper we investigate the method of (triangular) matrix interpretations that is widely used in current automated termination and complexity tools. Using techniques from linear algebra, we show how one can generalize the method of triangular matrix interpretations, as introduced in [126], to matrix interpretations that are not necessarily triangular but nevertheless induce polynomial upper bounds on the derivational complexity of compatible term rewrite systems. Moreover, we show that our approach also applies to matrix interpretations over the real (algebraic) numbers. In particular, we also show how one can infer tighter bounds from triangular matrix interpretations by examining the diagonal structure of upper triangular (complexity) matrices.

The remainder of this paper is organized as follows. Section 6.2 introduces basic notions of term rewriting and some mathematical prerequisites. In Section 6.3, we review matrix interpretations in the context of complexity analysis of term rewriting, before presenting our main result in Section 6.4. In Section 6.5, we give details on implementation-specific issues. Finally, we provide experimental results in Section 6.6, before concluding in Section 6.7.

[1] `http://termcomp.uibk.ac.at`

6.2. Preliminaries

We assume familiarity with the basics of term rewriting [17, 172]. Let $\mathcal{V}$ denote a countably infinite set of variables and $\mathcal{F}$ a fixed-arity signature. The set of *terms* over $\mathcal{F}$ and $\mathcal{V}$ is denoted by $\mathcal{T}(\mathcal{F},\mathcal{V})$. The *size* $|t|$ of a term t is defined as the number of symbols occurring in it and the *depth* of t is defined as follows: if t is a variable or a constant, then $\mathrm{depth}(t) := 0$, otherwise $\mathrm{depth}(f(t_1,\dots,t_n)) := 1 + \max\{\mathrm{depth}(t_i) \mid 1 \leqslant i \leqslant n\}$. A *rewrite rule* is a pair of terms written as $l \to r$, such that l is not a variable and all variables in r are contained in l. A *term rewrite system* $\mathcal{R}$ (TRS for short) over $\mathcal{T}(\mathcal{F},\mathcal{V})$ is a set of rewrite rules. For complexity analysis we assume TRSs to be finite. The rewrite relation induced by $\to$ is denoted by $\to_{\mathcal{R}}$. As usual, $\to^*_{\mathcal{R}}$ denotes the reflexive transitive closure of $\to_{\mathcal{R}}$ and $\to^n_{\mathcal{R}}$ its n-th iterate. A term $s \in \mathcal{T}(\mathcal{F},\mathcal{V})$ is called a *normal form* if there is no term t such that $s \to_{\mathcal{R}} t$.

The *derivation height* of a term t with respect to a TRS $\mathcal{R}$ is defined as follows: $\mathsf{dh}(t, \to_{\mathcal{R}}) := \max\{n \mid \exists u\ t \to^n_{\mathcal{R}} u\}$. The *derivational complexity* function of a terminating TRS $\mathcal{R}$ computes the maximal derivation height of all terms up to a given size, i.e., $\mathsf{dc}_{\mathcal{R}}\colon \mathbb{N} \setminus \{0\} \to \mathbb{N}, k \mapsto \max\{\mathsf{dh}(t, \to_{\mathcal{R}}) \mid |t| \leqslant k\}$. Sometimes we say that $\mathcal{R}$ has linear, quadratic, etc. derivational complexity if $\mathsf{dc}_{\mathcal{R}}(k)$ can be bounded by a linear, quadratic, etc. polynomial in k.

An important concept for establishing termination of TRSs is the notion of well-founded monotone algebras. An *$\mathcal{F}$-algebra* $\mathcal{A}$ consists of a non-empty carrier A and interpretation functions $f_{\mathcal{A}}\colon A^n \to A$ for every n-ary $f \in \mathcal{F}$. By $[\alpha]_{\mathcal{A}}(\cdot)\colon \mathcal{T}(\mathcal{F},\mathcal{V}) \to A$ we denote the usual evaluation function of $\mathcal{A}$ with respect to a variable assignment $\alpha\colon \mathcal{V} \to A$. A *well-founded monotone $\mathcal{F}$-algebra* is a pair $(\mathcal{A}, >_A)$, where $\mathcal{A}$ is an $\mathcal{F}$-algebra and $>_A$ is a well-founded order on A such that every $f_{\mathcal{A}}$ is strictly monotone in all arguments (with respect to $>_A$). A well-founded monotone algebra naturally induces an order $\succ_{\mathcal{A}}$ on terms: $s \succ_{\mathcal{A}} t$ if $[\alpha]_{\mathcal{A}}(s) >_A [\alpha]_{\mathcal{A}}(t)$ for all assignments α of elements of A to the variables in s and t. Finally, it is well-known that a TRS $\mathcal{R}$ is terminating if and only if it is *compatible* with a well-founded monotone algebra $(\mathcal{A}, >_A)$, where compatibility means that $l \succ_{\mathcal{A}} r$ for every rewrite rule $l \to r \in \mathcal{R}$.

Linear Algebra. As usual, we denote by $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ and $\mathbb{R}$ the sets of natural, integer, rational and real numbers. Given some $D \in \{\mathbb{N},\mathbb{Z},\mathbb{Q},\mathbb{R}\}$ and $m \in D$, $>_D$ denotes the natural order of the respective domain and $D_m := \{x \in D \mid x \geqslant m\}$; e.g., $\mathbb{R}_0$ refers to the set of all non-negative real numbers. For any ring R (e.g., $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$), we denote the ring of all n-dimensional square matrices over R by $R^{n\times n}$, and $R[x_1,\dots,x_n]$ denotes the associated *polynomial ring* in *n indetermi-*

nates $x_1, \ldots, x_n$. In the special case $n = 1$, a polynomial $P \in R[x]$ can be written as follows: $P(x) = \sum_{k=0}^{d} a_k x^k$ ($d \in \mathbb{N}$). For the largest k such that $a_k \neq 0$, we call $a_k x^k$ the *leading term* of P, a_k its *leading coefficient* and k its *degree*. P is said to be *monic* if its leading coefficient is one. Moreover, it is said to be *linear*, *quadratic*, *cubic* etc. if its degree is one, two, three etc.

We say that a matrix is *non-negative* if all its entries are non-negative. Abusing notation, we denote the set of all non-negative n-dimensional square matrices of $\mathbb{Z}^{n\times n}$ by $\mathbb{N}^{n\times n}$. An *upper triangular* matrix is a matrix, where all entries below the main diagonal are zero. An *upper triangular complexity* matrix is a non-negative upper triangular matrix whose diagonal entries are at most one and whose top-left entry is exactly one. As usual, we denote the *transpose* of a matrix (vector) A by A^T. The *characteristic polynomial* of a square matrix $A \in R^{n\times n}$ is defined as $\chi_A(\lambda) := \det(\lambda I_n - A)$, where I_n denotes the n-dimensional identity matrix and det the determinant of a matrix. It is monic and its degree is n. The equation $\chi_A(\lambda) = 0$ is called the *characteristic equation* of A. The *eigenvalues* of A are precisely the solutions of its characteristic equation, and the *spectral radius* $\rho(A)$ of A is the maximum of the absolute values of all eigenvalues. By m_λ we denote the *multiplicity* of the eigenvalue λ. A non-zero vector x is an *eigenvector* of A if $Ax = \lambda x$ for some eigenvalue λ of A. The Cayley-Hamilton theorem [151] states that every matrix satisfies its own characteristic equation, that is, $\chi_A(A) = 0$, and it holds for square matrices over commutative rings.

Recurrence Relations. Informally, a recurrence relation is an equation that recursively defines a sequence; each element of the sequence is defined as a function of the preceding elements. For example, the Fibonacci numbers are defined by $F_n = F_{n-1} + F_{n-2}$ with $F_0 = 0$ and $F_1 = 1$. Solving a recurrence relation means obtaining a closed-form solution; in this example, a non-recursive function of n.

A *linear homogeneous recurrence relation with constant coefficients* is an equation of the form $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_d a_{n-d}$, where the $d \geqslant 1$ *coefficients* $c_1, \ldots, c_d$ are constants with $c_d \neq 0$. The same coefficients yield the *characteristic polynomial* $\chi(\lambda) := \lambda^d - c_1\lambda^{d-1} - c_2\lambda^{d-2} - \cdots - c_d$ whose d roots play a key role in the solution of a recurrence relation (cf. [29, 30]). To be precise, if $\lambda_1, \lambda_2, \ldots, \lambda_r$ ($1 \leqslant r \leqslant d$) are the distinct (possibly complex) roots of the characteristic polynomial such that λ_i is of multiplicity m_i ($i = 1, 2, \ldots, r$), then the general solution of the recurrence relation is given by

$$a_n = \sum_{i=1}^{r} (c_{i1} + c_{i2} n + \cdots + c_{im_i} n^{m_i - 1}) \lambda_i^n$$

where the c_{ik}'s are (complex) constants. Any real solution is of this form as well, with the imaginary part zero. Moreover, if the coefficients of $\chi(\lambda)$ are real numbers, its non-real roots always come in conjugate pairs; i.e., if $\lambda_j := r_j(\cos(\phi_j)+i\sin(\phi_j))$ is a root of $\chi(\lambda)$, then so is its complex conjugate $\lambda_j^* := r_j(\cos(\phi_j) - i\sin(\phi_j))$. In this case, avoiding the use of complex numbers, the most general real solution can be written as

$$\begin{aligned} a_n = {} & \sum_i (c_{i1} + c_{i2}n + \cdots + c_{im_i}n^{m_i-1})\lambda_i^n \\ & + \sum_j (d_{j1} + d_{j2}n + \cdots + d_{jm_j}n^{m_j-1})r_j^n \cos(n\phi_j) \\ & + \sum_j (d'_{j1} + d'_{j2}n + \cdots + d'_{jm_j}n^{m_j-1})r_j^n \sin(n\phi_j) \end{aligned}$$

where the c_{ik}'s, d_{jk}'s and d'_{jk}'s are real constants, the λ_i's the distinct real roots of $\chi(\lambda)$ and the λ_j's, $\lambda_j := r_j(\cos(\phi_j)+i\sin(\phi_j))$, the distinct complex roots (modulo conjugates).

6.3. Matrix Interpretations and Derivational Complexity

Next we review the method of matrix interpretations in the context of complexity analysis of term rewriting. Matrix interpretations [44, 79] were originally introduced over the natural numbers. Later on they were lifted to the reals [2, 52, 208] using the same technique that was already used to lift polynomial interpretations from $\mathbb{N}$ to $\mathbb{R}$ (cf. [80]). Similarly, the first results relating matrix interpretations and derivational complexity of TRSs (cf. [126], triangular matrix interpretations) are based on matrix interpretations over the natural numbers. But these results have never been lifted to the reals. In the next section we shall see, however, how this follows from a more general result that holds for matrix interpretations over both $\mathbb{N}$ and $\mathbb{R}$, the foundations of which are laid in the present chapter.

Let $\mathcal{F}$ denote a signature. A *matrix interpretation* $\mathcal{M}$ over $\mathbb{N}$ is a well-founded monotone algebra, where the carrier M is the set $\mathbb{N}^n$ for some fixed dimension $n \in \mathbb{N} \setminus \{0\}$. The well-founded order $>_M$ on M is defined as follows:

$$(x_1, x_2, \ldots, x_n)^T >_M (y_1, y_2, \ldots, y_n)^T :\Longleftrightarrow x_1 >_{\mathbb{N}} y_1 \wedge x_2 \geqslant_{\mathbb{N}} y_2 \wedge \cdots \wedge x_n \geqslant_{\mathbb{N}} y_n$$

For each k-ary function symbol $f \in \mathcal{F}$, we choose an interpretation function

$$f_{\mathcal{M}} \colon (\mathbb{N}^n)^k \to \mathbb{N}^n, (\vec{x_1}, \ldots, \vec{x_k}) \mapsto F_1\vec{x_1} + \cdots + F_k\vec{x_k} + \vec{f}$$

where $\vec{f} \in \mathbb{N}^n$ and $F_1, \ldots, F_k \in \mathbb{N}^{n \times n}$. In addition, we require $(F_i)_{1,1} \geqslant_{\mathbb{N}} 1$ for all $i = 1, \ldots, k$ to achieve strict monotonicity of $f_{\mathcal{M}}$ in all arguments. Finally, a *triangular matrix interpretation* over $\mathbb{N}$ is a matrix interpretation over $\mathbb{N}$, where all matrices are upper triangular complexity matrices.

When extending matrix interpretations from $\mathbb{N}$ to $\mathbb{R}$, the main problem is the non-well-foundedness of $>_{\mathbb{R}}$. This problem is overcome by $>_{\mathbb{R},\delta}$, which is defined as follows: given some fixed positive real number δ, $x >_{\mathbb{R},\delta} y$ if and only if $x - y \geqslant_{\mathbb{R}} \delta$ for all $x, y \in \mathbb{R}$. Thus $>_{\mathbb{R},\delta}$ is well-founded on subsets of $\mathbb{R}$ that are bounded from below. Then a *matrix interpretation* $\mathcal{M}$ over $\mathbb{R}$ is a well-founded monotone algebra, where the carrier M is the set $\mathbb{R}_0^n$ for some fixed dimension $n \in \mathbb{N} \setminus \{0\}$. The well-founded order $>_M$ on M is defined as follows:

$$(x_1, x_2, \dots, x_n)^T >_M (y_1, y_2, \dots, y_n)^T :\Longleftrightarrow x_1 >_{\mathbb{R},\delta} y_1 \wedge x_2 \geqslant_{\mathbb{R}} y_2 \wedge \cdots \wedge x_n \geqslant_{\mathbb{R}} y_n$$

For each k-ary function symbol f, we choose an interpretation function

$$f_{\mathcal{M}} \colon (\mathbb{R}_0^n)^k \to \mathbb{R}_0^n, (\vec{x_1}, \dots, \vec{x_k}) \mapsto F_1\vec{x_1} + \ldots + F_k\vec{x_k} + \vec{f}$$

where $\vec{f} \in \mathbb{R}_0^n$ and $F_1, \dots, F_k$ are non-negative matrices in $\mathbb{R}^{n \times n}$ with $(F_i)_{1,1} \geqslant_{\mathbb{R}} 1$ for all $i = 1, \dots, k$ in order to achieve strict monotonicity of $f_{\mathcal{M}}$ in all arguments. Again, a *triangular matrix interpretation* over $\mathbb{R}$ is a matrix interpretation over $\mathbb{R}$, where all matrices are upper triangular complexity matrices.

Remark 6.1. *Concerning polynomial interpretations, it was recently shown in [130] that it suffices to consider the set $\mathbb{R}_{\mathsf{alg}}$ of real* algebraic[2] *numbers instead of the entire set $\mathbb{R}$ of real numbers. To be precise, it was shown that polynomial termination over $\mathbb{R}$ is equivalent to polynomial termination over $\mathbb{R}_{\mathsf{alg}}$. Observing that the technique of [130] readily applies to matrix interpretations as well, we may draw the conclusion that matrix interpretations over $\mathbb{R}$ are equivalent to matrix interpretations over $\mathbb{R}_{\mathsf{alg}}$ with respect to proving termination of TRSs.*

Matrix interpretations over $\mathbb{R}$ can be used to bound the derivational complexity of compatible TRSs.[3] Let $\mathcal{M}$ be a matrix interpretation over $\mathbb{R}$ that is compatible with some TRS $\mathcal{R}$. Then any rewrite sequence

$$t = t_0 \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} t_3 \to_{\mathcal{R}} t_4 \to_{\mathcal{R}} \cdots$$

gives rise to a strictly decreasing sequence of vectors of non-negative real numbers

$$[\alpha]_{\mathcal{M}}(t) >_M [\alpha]_{\mathcal{M}}(t_1) >_M [\alpha]_{\mathcal{M}}(t_2) >_M [\alpha]_{\mathcal{M}}(t_3) >_M [\alpha]_{\mathcal{M}}(t_4) >_M \cdots$$

[2]A real number is said to be algebraic if it is a root of a non-zero polynomial in one variable with integer coefficients.

[3]The reasoning presented in the sequel readily includes matrix interpretations over $\mathbb{N}$ as a special case (by letting $\delta = 1$ and observing that $x >_{\mathbb{N}} y$ if and only if $x \geqslant_{\mathbb{N}} y + 1$).

for all variable assignments α. In particular, by definition of $>_M$, the first components of these vectors form a sequence of non-negative real numbers that is strictly decreasing with respect to the order $>_{\mathbb{R},\delta}$, and every rewrite step causes a decrease of at least δ. Hence, the first component of the vector $\frac{1}{\delta} \cdot [\alpha]_{\mathcal{M}}(t)$ gives an upper bound on $\mathsf{dh}(t, \rightarrow_{\mathcal{R}})$. So if we manage to bound (the first component of) this vector for all terms t up to a given (but arbitrary) size k, then we have actually established an upper bound on the derivational complexity of $\mathcal{R}$. Moreover, as we are only interested in the asymptotic growth of $\frac{1}{\delta} \cdot [\alpha]_{\mathcal{M}}(t)$ with respect to the size of t, we may neglect the multiplicative factor $\frac{1}{\delta}$ because δ is a constant. As already observed in [126], this problem essentially reduces to bounding the entries of finite matrix products of the form $M_1 \cdot M_2 \cdot \ldots \cdot M_k$, $M_i \in \mathcal{M}$. Such products arise naturally when evaluating terms in a matrix interpretation; e.g., if $t := \mathsf{f}(\mathsf{g}(\mathsf{a}, \mathsf{b}), \mathsf{c})$ then $[\alpha]_{\mathcal{M}}(t) = F_1 G_1 \vec{a} + F_1 G_2 \vec{b} + F_1 \vec{g} + F_2 \vec{c} + \vec{f}$. As in [126], we reduce this problem to the analysis of the growth of the powers of a single matrix. To this end, we note that for all $1 \leqslant i, j \leqslant n$, $(M_1 \cdot M_2 \cdot \ldots \cdot M_k)_{i,j} \leqslant (A^k)_{i,j}$, where the matrix A is the component-wise maximum of all matrices occurring in $\mathcal{M}$; i.e., $A_{i,j} := \max\{B_{i,j} \mid B \in \mathcal{M}\}$ for all $1 \leqslant i, j \leqslant n$. If $|t| \leqslant k$ then the length of each product is at most $\mathrm{depth}(t)$ ($\leqslant k$) and the number of products equals the number of subterms of t, which is also bounded by k. Thus any lemma stating that the entries of the matrix A^k are polynomially bounded in k of degree $d - 1$ can readily be used as the basis of a corresponding theorem that establishes a polynomial upper bound of degree d on the derivational complexity of all TRSs that are compatible with the matrix interpretation $\mathcal{M}$. In [126], for example, this is achieved by restricting the shape of the matrices to upper triangular form.

Lemma 6.1 ([126, Lemma 5]). *Let $A \in \mathbb{N}^{n \times n}$ be an upper triangular complexity matrix and $k \in \mathbb{N}$. Then $(A^k)_{i,j} \in O(k^{n-1})$ for all $1 \leqslant i, j \leqslant n$.* □

Theorem 6.2 ([126, Theorem 6]). *If a TRS $\mathcal{R}$ is compatible with a triangular matrix interpretation of dimension n, then $\mathsf{dc}_{\mathcal{R}}(k) \in O(k^n)$.* □

However, we claim that Lemma 6.1 only gives a rough estimate of the growth of the entries of the matrix A^k, i.e., the degree of the polynomial bound can be lowered in many cases. To this end, we provide a more concise analysis of the growth of A^k in the next section, obtaining a replacement for Lemma 6.1, which allows us to tighten the bounds established by Theorem 6.2. In particular, our refinement holds for matrix interpretations over both $\mathbb{N}$ and $\mathbb{R}$. Moreover, we remark that the restriction of the shape of the matrices is another source for improvement. Clearly, there are also non-triangular matrices that exhibit polynomial growth, but in general non-triangular matrix interpretations do not induce polynomial (but rather exponential) upper bounds on the derivational complexity of compatible TRSs.

So in order to be useful in (automated) complexity analysis of term rewriting, a characterization of polynomially bounded matrices is required such that, when searching for a compatible matrix interpretation for a given TRS, it is guaranteed beforehand that the search process only considers such matrices. This is the main goal of the following sections.

6.4. Main Result

In this section we elaborate on how to lift the restriction to upper triangular matrices. To this end, we leverage the Cayley-Hamilton theorem and the theory of linear homogeneous recurrence relations to completely characterize the growth of the powers of real square matrices (independently of the shape of the matrices). In particular, we show that the key point with respect to polynomial boundedness of such matrices is the nature of their eigenvalues. According to the discussion in Section 6.3, our results apply to matrix interpretations over $\mathbb{N}$ and $\mathbb{R}$ alike.

Lemma 6.3. *Let $A \in \mathbb{R}_0^{n\times n}$. Then $\rho(A) \leqslant 1$ if and only if all entries of A^k ($k \in \mathbb{N}$) are asymptotically bounded by a polynomial in k of degree d, where $d := \max_\lambda(0, m_\lambda - 1)$ and λ are the eigenvalues with absolute value exactly one.*

Proof. First, let us assume that $\rho(A) > 1$, i.e., A has an eigenvalue λ of absolute value strictly greater than one. For any eigenvector x associated to λ, we have $Ax = \lambda x$ and hence $A^k x = \lambda^k x$. Since x is non-zero by definition and $|\lambda| > 1$, there is at least one component of $\lambda^k x$ whose absolute value grows exponentially in k. But this can only be the case if at least one entry of A^k grows exponentially in k as well. Conversely, if $\rho(A) \leqslant 1$, we have to show that the entries of A^k are polynomially bounded. Since A is a real $n \times n$ matrix, its characteristic polynomial $\chi_A(\lambda)$ is a monic polynomial of degree n with real coefficients. Without loss of generality, it can be written as $\chi_A(\lambda) = \lambda^t \cdot p(\lambda)$, $0 \leqslant t \leqslant n$, where t is maximal and p is a monic polynomial of degree $n - t$. By the Cayley-Hamilton theorem, A satisfies its own characteristic equation, that is, $\chi_A(A) = 0$. Clearly, if $t = n$ then $A^k = 0$ for all $k \geqslant n$ and $d = 0$, such that the claim follows trivially. If $t < n$ we rearrange the equation $\chi_A(A) = 0$ into the form $A^n = c_1A^{n-1}+c_2A^{n-2}+\cdots+c_{n-t}A^t$ with coefficients $c_1, \dots, c_{n-t}$, readily obtaining a recursive equation for the powers of A, namely, for all $k \geqslant n \in \mathbb{N}$ $A^k = c_1A^{k-1} + c_2A^{k-2} + \cdots + c_{n-t}A^{k-(n-t)}$. Thus we establish the following recurrence relation

$$A_k = c_1A_{k-1} + c_2A_{k-2} + \cdots + c_{n-t}A_{k-(n-t)} \tag{6.1}$$

and note that the sequence $(A_j)_{j\geqslant t}$ where $A_j := A^j$ satisfies it by construction. This is a linear homogeneous recurrence relation with constant coefficients and characteristic polynomial $\chi(\lambda) = p(\lambda)$. Since the coefficients of $\chi(\lambda)$ are real numbers, the non-real roots (eigenvalues) always come in conjugate pairs; i.e., if $\lambda_j := r_j(\cos(\phi_j) + i\sin(\phi_j))$ is a root of $\chi(\lambda)$, then so is its complex conjugate $\lambda_j^* := r_j(\cos(\phi_j) - i\sin(\phi_j))$. Thus the general solution of (6.1) can be written as

$$\begin{aligned} A_k = & \sum_i (C_{i,0} + C_{i,1}k + \cdots + C_{i,m_i-1}k^{m_i-1})\lambda_i^k \\ & + \sum_j (D_{j,0} + D_{j,1}k + \cdots + D_{j,m_j-1}k^{m_j-1}) r_j^k \cos(k\phi_j) \\ & + \sum_j (D'_{j,0} + D'_{j,1}k + \cdots + D'_{j,m_j-1}k^{m_j-1}) r_j^k \sin(k\phi_j) \end{aligned} \tag{6.2}$$

where the λ_i's are the distinct real roots of $\chi(\lambda)$, each having multiplicity m_i, and the λ_j's, $\lambda_j := r_j(\cos(\phi_j) + i\sin(\phi_j))$, the distinct complex roots (modulo conjugates), each having multiplicity m_j. By assumption, the absolute values of all eigenvalues are at most one; hence, $|\lambda_i| \leqslant 1$ and $r_j \leqslant 1$ in (6.2), such that the asymptotic growth of the entries of the matrix A^k is polynomial rather than exponential. In particular, the degree d of the polynomial bound is at most $m-1$, where m is the largest of the multiplicities of the eigenvalues with absolute value exactly one. If there are no such eigenvalues, then $\rho(A) < 1$ and $\lim_{k\to\infty} A^k = 0$, such that $d = 0$. $\square$

Example 6.4. Consider the 4×4 matrix $A := (A_{i,j})_{1\leqslant i,j\leqslant 4}$ with all entries zero except $A_{1,1} = A_{2,4} = A_{3,2} = A_{4,3} = 1$. It has one real eigenvalue $\lambda_1 = 1$ of multiplicity two and a pair of complex conjugate eigenvalues $\lambda_2 = \frac{1}{2}(-1 + i\sqrt{3})$ and $\lambda_2^* = \frac{1}{2}(-1 - i\sqrt{3})$ of multiplicity one, all of which have absolute value exactly one. Hence, the spectral radius $\rho(A)$ of A is also one. According to Lemma 6.3, the entries of the matrix A^k, $k \in \mathbb{N}$, are bounded by a linear polynomial in k. The actual bound, however, is even lower since $A^4 = A$, such that the powers of A are trivially bounded by a constant, and we can use the method outlined in the proof of Lemma 6.3 to show this. To this end, we note that the characteristic polynomial of A is $\chi_A(\lambda) = \lambda^4 - \lambda^3 - \lambda + 1$. Thus, by the Cayley-Hamilton theorem, we obtain the recursive equation $A^k = A^{k-1} + A^{k-3} - A^{k-4}$ for all $k \geqslant 4 \in \mathbb{N}$, the general solution of which can be written as

$$A^k = (C_0 + C_1 k)\lambda_1^k + D\, r^k \cos(k\phi) + D'\, r^k \sin(k\phi) \tag{6.3}$$

where $r(\cos(\phi) + i\sin(\phi)) = \lambda_2$, that is, $r = 1$ and $\phi = \frac{2\pi}{3}$. In the next step, the exact values of the four constants C_0, C_1, D and D' can be determined, for example, by letting $k = 4, 5, 6, 7$ in (6.3) and solving the resulting systems of linear equations. In doing so, one learns that C_1 is zero, which means that the linear summand in (6.3) vanishes. Further, we obtain $A^k = C_0 + D\cos(k\phi) + D'\sin(k\phi)$,

$$C_0 := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix} \quad D := \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ 0 & -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \\ 0 & -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \end{pmatrix} \quad D' := \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{\sqrt{3}}{3} & \frac{\sqrt{3}}{3} \\ 0 & \frac{\sqrt{3}}{3} & 0 & -\frac{\sqrt{3}}{3} \\ 0 & -\frac{\sqrt{3}}{3} & \frac{\sqrt{3}}{3} & 0 \end{pmatrix}$$

which explains why the powers of A are bounded by a constant. In particular, the periodic nature of the sequence $(A^k)_{k\in\mathbb{N}}$ becomes evident.

On the basis of Lemma 6.3, we now establish the following theorem concerning complexity analysis of TRSs that holds for matrix interpretations over $\mathbb{N}$ and $\mathbb{R}$.

Theorem 6.5. *Let $\mathcal{R}$ be a TRS and $\mathcal{M}$ a compatible matrix interpretation of dimension n. Further, let A denote the component-wise maximum of all matrices occurring in $\mathcal{M}$. If the spectral radius of A is at most one, then $\mathsf{dc}_{\mathcal{R}}(k) \in O(k^{d+1})$, where $d := \max_\lambda(0, m_\lambda - 1)$ and λ are the eigenvalues of A with absolute value exactly one.* □

Remark 6.2. *Actually the d in Theorem 6.5 can be strengthened to* $\max_\lambda(0, m_\lambda)-1$ *because the pathological case $\rho(A) < 1$ implies $\mathsf{dc}_{\mathcal{R}}(k) \in O(k^0)$.*

The next example shows why triangular matrices may fail. Similar (but larger) systems are contained in TPDB [183], e.g., TRS/Cime_04/dpqs.xml.

Example 6.6. Consider the TRS $\mathcal{R} = \{\mathsf{f}(\mathsf{f}(x)) \to \mathsf{f}(\mathsf{c}(\mathsf{f}(x))), \mathsf{c}(\mathsf{c}(x)) \to x\}$, which is compatible with the matrix interpretation

$$\mathsf{f}_{\mathcal{M}}(\vec{x}) = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \vec{x} + \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \mathsf{c}_{\mathcal{M}}(\vec{x}) = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \vec{x} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

The eigenvalues of the component-wise maximum matrix are -1, 1 and 1; hence, Theorem 6.5 deduces a quadratic upper bound on the derivational complexity of $\mathcal{R}$. There cannot exist a *triangular* matrix interpretation compatible with $\mathcal{R}$ since the second rule demands that all diagonal entries in $\mathsf{c}_{\mathcal{M}}$ are non-zero, but then the first rule can no longer be oriented.

Next we specialize Theorem 6.5 to triangular matrix interpretations. In such interpretations all matrices are upper triangular complexity matrices whose diagonal entries are restricted to the closed interval $[0,1]$ and whose top-left entry is always one. Hence, this is also true for the component-wise maximum matrix A. Since the diagonal entries of a triangular matrix give the multiset of its eigenvalues, the matrix A is therefore guaranteed to have spectral radius one.

Theorem 6.7. *Let $\mathcal{R}$ be a TRS and $\mathcal{M}$ a compatible triangular matrix interpretation over $\mathbb{N}$ or $\mathbb{R}$ of dimension n. Further, let A denote the component-wise maximum of all matrices occurring in $\mathcal{M}$, and let d denote the number of ones occurring along the diagonal of A. Then $\mathsf{dc}_{\mathcal{R}}(k) \in O(k^d)$.*[4] $\square$

Note that the bound established by Theorem 6.7 for matrix interpretations over $\mathbb{N}$ is at least as tight as the one of Theorem 6.2 since $d \leqslant n$.

Example 6.8. The TRS $\mathcal{R} = \{\mathsf{a}(\mathsf{b}(\mathsf{a}(x))) \to \mathsf{a}(\mathsf{b}(\mathsf{b}(\mathsf{a}(x)))), \mathsf{b}(\mathsf{b}(\mathsf{b}(x))) \to \mathsf{b}(\mathsf{b}(x))\}$[5] is compatible with the triangular matrix interpretation

$$\mathsf{a}_{\mathcal{M}}(\vec{x}) = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \vec{x} + \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \qquad \mathsf{b}_{\mathcal{M}}(\vec{x}) = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \vec{x} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

The diagonal of the component-wise maximum of the two matrices has the shape $(1,0,0)$. Hence, $\mathcal{R}$ has (at most) linear derivational complexity by Theorem 6.7, whereas the bound established by Theorem 6.2 is cubic. Incidentally, the bound inferred from Theorem 6.7 is even optimal since it is easy to see that the derivational complexity of $\mathcal{R}$ is at least linear. It is easy to show that there are no triangular matrix interpretations of dimension one and two compatible with $\mathcal{R}$.

The final example shows the benefit of matrix interpretations over $\mathbb{R}$.

Example 6.9. Consider the TRS $\mathcal{R}$.[6] There exists a matrix interpretation (see website in Footnote 8) compatible with $\mathcal{R}$ such that the diagonal of the component-wise maximum matrix has the shape $(1, \frac{1}{2}, 0)$. Due to Theorem 6.7, the derivational complexity of $\mathcal{R}$ is at most linear. Our implementation could find a triangular matrix interpretation of the same dimension over $\mathbb{N}$ compatible with $\mathcal{R}$ establishing a quadratic but not a linear bound.

6.5. Implementation Issues

In Theorem 6.5, we consider some TRS together with a compatible matrix interpretation and demand that the component-wise maximum matrix A has spectral

[4]Independently in [187, Proposition 7.6] the same result has been established for $\mathbb{N}$.
[5]TPDB problem TRS/Zantema_04/z126.xml
[6]TPDB problem TRS/Secret_05_SRS/matchbox2.xml

radius at most one. So we have to make sure that the absolute values of all its eigenvalues (real and complex ones) are at most one. However, since A is a non-negative real square matrix, we only have to ensure this condition for all (non-negative) real eigenvalues of A. This follows directly from the Perron-Frobenius theorem ([159], weak form), which states that the spectral radius of a non-negative real square matrix is an eigenvalue of the matrix; i.e., there exists a non-negative real eigenvalue that dominates in absolute value all eigenvalues.

Concerning the automation of Theorem 6.5, the main problem that has to be dealt with is the following. Given some square matrix A with unknown entries, all of which are supposed to be non-negative real (or integer) numbers, we need a set of constraints, expressed in terms of the unknown entries, that enforce $\rho(A) \leqslant 1$; e.g., for which non-negative values of a, b, c and d has the matrix

$$A := \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

spectral radius at most one? In the sequel, we present three different approaches.

(A)

The first approach is based on the explicit calculation of the eigenvalues of A, i.e., the explicit calculation of the roots of the characteristic polynomial $\chi_A(\lambda)$. For the two-dimensional case, we have $\chi_A(\lambda) = \lambda^2 - (a + d)\lambda + ad - bc$, and by the quadratic formula we obtain the roots $\lambda_{1,2} = \frac{a+d}{2} \pm \frac{\sqrt{(a-d)^2+4bc}}{2}$, both of which are real because all matrix entries are non-negative. In particular, λ_2 ($\geqslant \lambda_1$) is non-negative, such that it suffices to require $\lambda_2 \leqslant 1$ according to the Perron-Frobenius theorem. Simplifying this condition as much as possible, we infer that the matrix A has spectral radius at most one if and only if $a + d \leqslant 2$ and $a + d \leqslant ad - bc + 1$. This explicit approach also applies to matrices of dimension three and four since there exist formulas for the solution of arbitrary cubic and quartic polynomial equations with symbolic coefficients (though the respective calculations are tedious). However, for equations of degree five or higher, there are no formulas that express the solutions of such equations in terms of their coefficients using only the four basic arithmetic operations and radicals (n-th roots, for some integer n).

(B)

Next we present an alternative and simpler approach for three-dimensional matrices. To this end, let A be some arbitrary three-dimensional non-negative real square matrix with entries $a, b, \ldots, i$ and characteristic polynomial $\chi_A(\lambda)$

$$\lambda^3 - (a+e+i)\lambda^2 + (ei - fh + ai - cg + ae - bd)\lambda - (aei + bfg + cdh - ceg - bdi - afh)$$

which we abbreviate by $\lambda^3 + p\lambda^2 + q\lambda + r$. By the Perron-Frobenius theorem, it suffices to constrain the real roots of $\chi_A(\lambda)$ to the closed interval $[-1, 1]$. To this end, we make use of the well-known fact that a cubic polynomial like $\chi_A(\lambda)$ either has only one real root (and two complex conjugate roots) if its *discriminant* $D := p^2q^2 - 4q^3 - 4p^3r - 27r^2 + 18pqr$ is negative or three (not necessarily distinct) real roots if $D \geqslant 0$. Visualizing the geometric shape of $\chi_A(\lambda)$, it is not hard to see that in the latter case all three roots are in $[-1, 1]$ if and only if $\chi_A(-1) \leqslant 0$, $\chi_A(1) \geqslant 0$ and $\chi'_A(\lambda) \geqslant 0$ for all $\lambda \in \mathbb{R}$ with $|\lambda| \geqslant 1$ (here χ'_A denotes the first derivative of χ_A). Thus we conclude that the matrix A has spectral radius at most one if and only if

$$\begin{aligned}&(D < 0 \wedge \chi_A(-1) \leqslant 0 \wedge \chi_A(1) \geqslant 0) \vee \\ &(\chi_A(-1) \leqslant 0 \wedge \chi_A(1) \geqslant 0 \wedge \chi'_A(\lambda) \geqslant 0 \text{ for all } |\lambda| \geqslant 1)\end{aligned}$$

These are polynomial constraints in the entries of A. In particular, the constraint $\chi'_A(\lambda) = 3\lambda^2 + 2p\lambda + q \geqslant 0$ for all $|\lambda| \geqslant 1$ can be shown to be equivalent to

$$(p^2 - 3q \leqslant 0) \vee (-3 \leqslant p \leqslant 3 \wedge -(q+3) \leqslant 2p \leqslant q + 3)$$

by means of the quadratic formula. Here the term $p^2 - 3q$ is essentially the discriminant of $\chi'_A(\lambda)$; if it is negative, then $\chi'_A(\lambda)$ has no real root, such that the constraint holds trivially, otherwise it has two real roots λ_1 and λ_2. In case $\lambda_1 = \lambda_2$, the constraint also holds because then $\chi'_A(\lambda) = 3 \cdot (\lambda - \lambda_1)^2$. Finally, if $\lambda_1 \neq \lambda_2$, then both must necessarily lie in the closed interval $[-1, 1]$ for the constraint to hold, which is ensured by the second disjunct in the above formula.

(C)

Last but not least, we present a generic method that works for matrices with unknown entries of any dimension. To this end, let A be an n-dimensional square matrix whose entries are supposed to be real numbers (not necessarily non-negative). Its characteristic polynomial is a monic polynomial of degree n, which can be written as $\chi_A(\lambda) = \lambda^n + \sum_{i=0}^{n-1} c_i\lambda^i$, where the coefficients c_i, $0 \leqslant i \leqslant n-1$, are polynomial expressions in the entries of A. Since all coefficients are supposed to be real numbers, $\chi_A(\lambda)$ can always be factored as

$$\chi_A(\lambda) = (\lambda - r)^b \cdot \prod_j (\lambda^2 + p_j\lambda + q_j)^{m_j} \tag{6.4}$$

where $b = 0$ if n is even, $b = 1$ otherwise, $m_j \geqslant 1$ ($m_j \in \mathbb{N}$) is the multiplicity of the quadratic factor $\lambda^2 + p_j\lambda + q_j$, and $r, p_j, q_j \in \mathbb{R}$. Thus the absolute values of all roots (real and complex ones) of $\chi_A(\lambda)$ are at most one if and only if $|r| \leqslant 1$ (in case $b = 1$) and the absolute values of the roots of all quadratic factors are at most one. So when does the latter condition hold for a given quadratic factor $\lambda^2 + p_j\lambda + q_j$? By the quadratic formula, we obtain the roots $\lambda_{1,2} := -\frac{p_j}{2} \pm \frac{\sqrt{p_j^2-4q_j}}{2}$. If the discriminant $p_j^2 - 4q_j$ is negative, both roots are complex, i.e., $\lambda_{1,2} := -\frac{p_j}{2} \pm i\frac{\sqrt{4q_j-p_j^2}}{2}$ and have absolute value $|\lambda_1| = |\lambda_2| = \sqrt{q_j}$. Hence, we demand $\sqrt{q_j} \leqslant 1$, or equivalently, $q_j \leqslant 1$. In the other case, if $p_j^2 - 4q_j \geqslant 0$, both roots are real, and the constraints $|\lambda_1| \leqslant 1$ and $|\lambda_2| \leqslant 1$ simplify to

$$-2 \leqslant p_j \leqslant 2 \text{ and } -(q_j + 1) \leqslant p_j \leqslant q_j + 1$$

As a consequence, the matrix $A \in \mathbb{R}^{n\times n}$ with characteristic polynomial (6.4) has spectral radius at most one if and only if $b = 1$ implies $-1 \leqslant r \leqslant 1$ and for all quadratic factors $\lambda^2 + p_j\lambda + q_j$ in (6.4),

$$(p_j^2 - 4q_j < 0 \wedge q_j \leqslant 1) \vee (p_j^2 - 4q_j \geqslant 0 \wedge -2 \leqslant p_j \leqslant 2 \wedge -(q_j + 1) \leqslant p_j \leqslant q_j + 1)$$

Non-negative Integer Matrices. If all matrix entries are non-negative integers, one can also apply a totally different approach. It is based on graph theory and the following lemma, which is an immediate consequence of [91, Corollary 1].[7]

Lemma 6.10. *Let $A \in \mathbb{N}^{n\times n}$. Then $\rho(A) > 1$ if and only if $(A^k)_{i,i} > 1$ for some $k \in \mathbb{N}$ and $i \in \{1, \dots, n\}$.* □

Viewing $A \in \mathbb{N}^{n\times n}$ as the adjacency matrix of a directed weighted graph G_A of n vertices numbered from 1 to n, such that for every positive entry $A_{i,j}$ there is an edge from vertex i to vertex j of weight $A_{i,j}$, the condition $(A^k)_{i,i} > 1$ for some $i \in \{1, \dots, n\}$ mentioned in the previous lemma holds if and only if

1. there is a cycle in G_A containing at least one edge of weight $w > 1$, or

2. there are (at least) two different paths (cycles) from some vertex to itself.

This is due to the well-known fact that the entry $(A^k)_{i,j}$ equals the sum of the weights of all distinct paths in G_A of length k from vertex i to vertex j, where the weight w of a path is the product of the weights of its edges (in particular, $w \geqslant 1$). Hence, we have $\rho(A) \leqslant 1$ if and only if neither of the two conditions holds. Since every cycle of G_A is composed of simple cycles, that is, cycles with no repeated vertices (aside from the necessary repetition of the start and end vertex), we may restrict to simple cycles for both conditions.

[7] The joint spectral radius of a singleton set $\{A\}$ of matrices coincides with $\rho(A)$.

Next we make two important observations. First, for $A \in \mathbb{N}^{n \times n}$, G_A cannot have a simple cycle containing an edge of weight greater than one if every matrix in the set $\{A, A^2, \ldots, A^n\}$ has diagonal entries less than or equal to one. Concerning the second condition, let us assume that there are two different simple cycles C_1 and C_2 of length l_1 and l_2, $1 \leqslant l_1, l_2 \leqslant n$, from some vertex i to itself. Considering all paths of length $\mathsf{lcm}(l_1, l_2)$, the least common multiple of l_1 and l_2, we clearly have $(A^{\mathsf{lcm}(l_1,l_2)})_{i,i} > 1$. In addition, we also have $(A^{l_1+l_2})_{i,i} > 1$ because there are two different cycles, each of weight at least one, from vertex i to itself of length $l_1 + l_2$, namely, the concatenation of C_1 and C_2 as well as the concatenation of C_2 and C_1. Hence, we can detect the existence of the cycles C_1 and C_2 by examining the diagonal entries of all matrices in the set $\{A, A^2, \ldots, A^m\}$, where $m := \min(l_1 + l_2, \mathsf{lcm}(l_1, l_2))$. More generally, we can detect any pair of cycles satisfying condition 2 by examining the diagonal entries of the matrices in the set $\{A, A^2, \ldots, A^{p(n)}\}$, where $p(n) := \max\{\min(l_1 + l_2, \mathsf{lcm}(l_1, l_2)) \mid 1 \leqslant l_1, l_2 \leqslant n\}$. The left part of the table below shows the values of $p(n)$ for various values of n.

n	1	2	3	4	5	6
$p(n)$	1	2	5	7	9	11

n	1	2	3	4	5	6
$q(n)$	1	2	3	5	7	9

In particular, we observe that $p(n) \geqslant n$ for $n \geqslant 1$, and we draw the following conclusion. If every matrix in the set $\{A, A^2, \ldots, A^{p(n)}\}$ has diagonal entries less than or equal to one, then neither condition 1 nor condition 2 can hold, which implies $\rho(A) \leqslant 1$. The converse is obvious.

Now let us apply this result to matrix interpretations. By definition, all matrices of a matrix interpretation $\mathcal{M}$ must have a top-left entry of at least one. Hence, this is also true for the maximum matrix A of $\mathcal{M}$. In other words, in G_A, vertex 1 has a loop (of length one) to itself. This corresponds to a dimension reduction by one for precluding all instances of condition 2. More precisely, we do not have to consider the cases $l_1 = n$ or $l_2 = n$ because then not only C_1 and C_2 but also C_1 (C_2) and the loop of vertex 1 satisfy condition 2 (for $n > 1$), and we can detect this by examining the diagonal entries of the matrix A^n, which has to be considered anyway for precluding all instances of condition 1. Therefore, if $A_{1,1} > 0$, we have $\rho(A) \leqslant 1$ if and only if every matrix in the set $\{A, A^2, \ldots, A^{q(n)}\}$ has diagonal entries less than or equal to one, where $q(n) := \max(n, p(n-1))$ for $n > 1$ and $q(1) := 1$. Some values for $q(n)$ are displayed in the right part of the above table.

6.6. Experimental Results

The criteria proposed in this paper have been implemented in the complexity tool CaT [206] and the 1172 non-duplicating TRSs in TPDB 7.0.2 have been considered.

	$O(k)$	$O(k^2)$	$O(k^3)$	$O(k^n)$
Theorem 6.2|6.7$_{\mathbb{N}}$|6.7$_{\mathbb{R}}$	46| 85| 88	158|184|185	177|202|196	203|205|199
A|B|C|Lemma 6.10	61|68| 80| 64	158|176|185|175	–|182|191|180	–|–|193|190
row 1|row 2|row 1+2	88| 80| 88	191|185|200	205|191|209	208|196|212
CaT (2009)|CaT (2010)	208|214	299|309	310|321	328|329

Table 6.1.: Polynomial bounds for 1172 systems.

All tests have been performed on a server equipped with 64 GB of main memory and eight dual-core AMD Opteron® 885 processors running at a clock rate of 2.6 GHz with a time limit of 60 seconds per system.[8]

We searched for matrix interpretations of dimension $d \in \{1, \ldots, 5\}$ by encoding the constraints as an SMT problem (quantifier-free non-linear arithmetic), which is solved by bit-blasting. We used $\max(2, 6-d)$ $(7-d)$ bits to represent coefficients (intermediate results). The numerators of rational numbers are represented with the bit-width mentioned above while all denominators are 2. CaT found compatible matrix interpretations (not necessarily polynomially bounded) for 287 TRSs, giving an upper bound on the number of systems our results can apply to (if used stand-alone).

Table 6.1 indicates the number of systems where the labeled approach yields polynomial upper bounds on the derivational complexity. The first row shows that the theorems proposed in this paper allow to infer tighter upper bounds from triangular matrices than [126]; e.g., the number of linear (quadratic) upper bounds increases by 84% (16%) if one compares Theorems 6.2 and 6.7$_{\mathbb{N}}$. The results for (possibly) non-triangular matrix interpretations are reported in the second row. The generic method based on factoring the characteristic polynomial (C) is implemented by comparing the coefficients from the characteristic polynomial with the coefficients of equation (6.4). Note that only this non-triangular approach allows to add upper bounds on the multiplicity of eigenvalues to the matrix encoding, which explains the high score for linear bounds. Since encoding A (B) is becoming harder for larger dimensions, we implemented it for dimensions one and two (and three) only (explaining the – in the table). Row three relates the approaches based on triangular and non-triangular matrices. Here row 1 corresponds to the accumulated power of Theorems 6.2 and 6.7 and row 2 to A, B, C, and Lemma 6.10, respectively. The impact of the methods proposed in this paper when integrated into the 2009 competition version of CaT is shown in row four. CaT was the strongest (derivational) complexity prover in 2008, 2009, and 2010. Since most parts of this paper aim at tightening bounds, it is not surprising that the total number of polynomial bounds did not increase significantly.

[8]For full details see `http://cl-informatik.uibk.ac.at/software/cat/polymatrix`.

6.7. Conclusion, Related and Future Work

We have presented a characterization of matrix interpretations that induce polynomial upper bounds on the derivational complexity of compatible TRSs. Contrary to previous approaches, our method applies to matrix interpretations over $\mathbb{N}$ and $\mathbb{R}$ alike and does not restrict the shape of the matrices. At the core of our method is the analysis of the growth of finite products of matrices. In particular, we estimate the growth of a product of the form $M_1 \cdot M_2 \cdot \ldots \cdot M_k$ by the growth of a (suitably chosen) matrix A^k, which is determined by its spectral radius. For future work, the investigation of joint spectral radius theory [91] looks promising since the joint spectral radius is a measure of the maximal growth of products of matrices taken from a set and has been the subject of intense research.

Concerning related work, very recently (and independently) Waldmann [187] provides a characterization of polynomially bounded matrix interpretations over $\mathbb{N}$, which extends triangular matrix interpretations. In [187] matrices are viewed as weighted (word) automata and the derivational complexity of TRSs is bounded by the growth of the weight function computed by such automata. We believe that the method is at least as powerful as our approach for matrix interpretations over $\mathbb{N}$. In contrast to our approach, it can handle the TRS in [187, Example 7.5], probably because it is not based on the maximum matrix. In practice, the method based on automata is much harder to implement (cf. [187, Section 8]). Unlike our approach, it only applies to matrix interpretations over $\mathbb{N}$; the extension to $\mathbb{R}$ ($\mathbb{Q}$) raises non-trivial issues (cf. [187, Section 10]).

Part IV

Termination

7. Beyond Polynomials and Peano Arithmetic – Automation of Elementary and Ordinal Interpretations

Publication Details

Harald Zankl, Sarah Winkler, and Aart Middeldorp, Beyond Polynomials and Peano Arithmetic – Automation of Elementary and Ordinal Interpretations, Journal of Symbolic Computation, accepted for publication, to appear.
doi: 10.1016/j.jsc.2014.09.033

Sarah Winkler, Harald Zankl, and Aart Middeldorp, Beyond Peano Arithmetic – Automatically Proving Termination of the Goodstein Sequence, In Proceedings of the 24th International Conference on Rewriting Techniques and Applications (RTA 2013), Leibniz International Proceedings in Informatics 21, pp. 335–351, Schloss Dagstuhl, 2013.
doi: 10.4230/LIPIcs.RTA.2013.335

Abstract

Kirby and Paris (1982) proved in a celebrated paper that a theorem of Goodstein (1944) cannot be established in Peano arithmetic. We present an encoding of Goodstein's theorem as a termination problem of a finite rewrite system. Using a novel implementation of algebras based on ordinal interpretations, we are able to automatically prove termination of this system, resulting in the first automatic termination proof for a system whose derivational complexity is not multiple recursive. Our method can also cope with the encoding by Touzet (1998) of the battle of Hercules and Hydra as well as a (corrected) encoding by Beklemishev (2006) of the Worm battle, two further systems which have been out of reach for automatic tools, until now. Based on our ideas of implementing ordinal algebras we also present a new approach for the automation of elementary interpretations for termination analysis.

7.1. Introduction

Since the beginning of the millennium there has been much progress regarding automated termination tools for rewrite systems.[1] Despite the many different techniques that have been developed, it seems that (terminating) TRSs which admit very long derivations are out of reach even for the most powerful tools. This is not surprising since many base methods induce rather small upper bounds on the derivational complexity, which is a function that bounds the length of the longest possible derivation (rewrite sequence) by the size of its starting term. Hofbauer and Lautemann [77] have shown that polynomial interpretations are limited to double exponential derivational complexity. They further showed that the derivational complexity of a rewrite system compatible with the Knuth-Bendix order (KBO) cannot be bounded by a primitive recursive function. Later, Lepper [114] established the Ackermann function as an upper bound for KBO, whereas Weiermann [190] proved a multiple recursive upper bound for the lexicographic path order (LPO). More recently, Moser and Schnabl [125, 155] have studied upper bounds on the complexity when using these base methods in the dependency pair framework. Although dependency pairs significantly increase termination proving power, from the viewpoint of derivational complexity the limit is still multiple recursive. This has led to the conjecture [155, Conjecture 6.99] that for any system whose termination can be proved automatically by modern tools the length of its derivations can be bounded by a multiple recursive function (in the size of the starting terms).

Ordinals have been used in termination arguments for many decades (e.g., [54, 184]). In fact ordinals are essential to prove termination of the battle of Hercules and Hydra (also due to [96]), or the sequences associated with Goodstein's theorem since these derivations cannot be bounded by a multiple recursive function (Cichon [31]). Although TRS encodings of the Hydra battle are known for many years (e.g., by Touzet [178]), they could so far not be handled by automatic termination tools, witnessing Schnabl's conjecture. Indeed a successful implementation of ordinals for automatic termination proofs is still lacking. Very recently, Urban and Miné [185] presented an approach to conclude termination of imperative programs by inferring ordinal-valued ranking functions. Here ordinals are essential to handle nondeterminism, though only ordinals below ω^{ω^ω} are involved and hence the ranking functions are still multiple recursive. The theorem prover Vampire uses ordinal numbers (see [108, Section 7]) in its implementation of KBO but only for weights of predicate symbols. As these symbols occur only at the root of atomic expressions no ordinal arithmetic is needed but comparisons suffice.

[1] http://www.termination-portal.org/

In this article we first encode the computation of Goodstein sequences (see Theorem 7.9) as a rewrite system $\mathcal{G}$ such that termination of $\mathcal{G}$ implies Goodstein's theorem. Since these sequences cannot be bounded by a multiple recursive function, this also holds for the derivational complexity of $\mathcal{G}$. After presenting this motivating example, we discuss automation of a termination criterion based on ordinal interpretations which is capable of proving $\mathcal{G}$ terminating, thereby overcoming the limitations alleged by the above conjecture. Our implementation can also cope with Touzet's encoding [178] of the battle of Hercules and Hydra, as well as a (corrected) encoding of the Worm battle [21].

Automation of ordinal interpretations is challenging since ordinal arithmetic does, e.g., not satisfy commutativity. Hence in contrast to polynomial interpretations terms do not evaluate to expressions of a canonical shape. We tackle this deficiency by introducing approximations which yield expressions of a special shape. Approximations (albeit less involved) have already been used for polynomial interpretations with negative [49, 72] or irrational [208] coefficients. In preliminary work [194, 203] already used ordinal domains to increase automatic termination proving power. However, in [203] the focus is on string rewriting and the interpretation functions have a very limited shape to avoid ordinal arithmetic. As a consequence the method is limited to systems with at most multiple *exponential* derivational complexity. Similarly, [194] use ordinal domains for generalized KBO, again for string rewriting only. In the respective implementation, function symbol weights are moreover below ω^ω. We anticipate that our treatment of arithmetic for ordinals up to ϵ_0 could improve some of the results from [108, 185, 194].

Lescanne [115] proposed elementary functions for proving (AC-)termination but his implementation is limited to checking the orientation of rules for *given* interpretations. Lucas [117] considers so-called linear elementary interpretations (LEIs) of the shape $A(\overline{x}) + B(\overline{x})^{C(\overline{x})}$ where $A(\overline{x})$, $B(\overline{x})$, and $C(\overline{x})$ are linear polynomials. Furthermore, he proposes an approach based on rewriting, constraint logic programming (CLP), and constraint satisfaction problems (CSPs) to also *find* suitable interpretation functions. He leaves an actual implementation of his method as future work and mentions the need for heuristics to achieve an efficient implementation. In this article we propose a different shape of interpretation functions because LEIs are neither closed under (scalar) multiplication, addition, nor composition. Furthermore, the motivating example in [117] (which is a simplified version of the leading example in [115]), uses a non-linear (elementary) interpretation for multiplication. We show that also an implementation of algebras with elementary interpretations can take advantage from an approximation-based approach. These findings are related to Problem #28 in the RTA List of Open

Problems,[2] which asks to "develop effective methods to decide whether a system decreases with respect to some exponential interpretation". Our contribution is restricted to a subclass of elementary interpretations but also admits the search for suitable interpretations.

This article is organized as follows. In the next section we recall ordinal arithmetic and weakly monotone algebras for termination proofs. In Section 7.3 we present our encoding of Goodstein's theorem and prove its correctness. Section 7.4 discusses how ordinal algebras can be automated and applies the approach to several rewrite systems (some of them encoding the Hydra battle), where also the limitations of our method become apparent. Likewise, Section 7.5 adapts the approach to elementary interpretations. Experimental results are the topic of Section 7.6. We conclude in Section 7.7.

This article is an updated and extended version of [193]. In particular, the extension to elementary interpretations (Section 7.5) and the experimental evaluation (Section 7.6) are new. Furthermore, in Section 7.4 the approximation $+_\mu$ has been refined (to succeed on the Worm battle) while tiny flaws in the approximations $+_\nu$ and $\oplus_\nu$ have been corrected (cf. Definition 7.22).

7.2. Preliminaries

We recall some preliminaries about ordinal numbers. Ordinals are transitive sets well-ordered with respect to $\in$. Hence $\alpha < \beta$ if and only if $\alpha \in \beta$. By identifying $\varnothing$, $\{\varnothing\}$, $\{\varnothing, \{\varnothing\}\}$, $\ldots$ with 0, 1, 2, $\ldots$, the natural numbers are embedded in the ordinals. If α is an ordinal then the ordinal $\alpha \cup \{\alpha\}$ is its successor, denoted by $\alpha + 1$. An ordinal β constitutes a successor ordinal if there is some α such that $\beta = \alpha + 1$, otherwise β is called a limit ordinal. For instance 1, 2, 3, $\ldots$ are successor ordinals, whereas 0 and the smallest infinite ordinal ω are limit ordinals. The latter is equivalent to the set of all natural numbers. The following ordinal arithmetic operations constitute extensions of the respective operations on natural numbers (see [92] for details).

Definition 7.1. For ordinals α and β their *sum* $\alpha + \beta$ is defined by recursion over β as (a) $\alpha + 0 = \alpha$, (b) $\alpha + \beta = (\alpha + \gamma) + 1$ if $\beta = \gamma + 1$, and (c) $\alpha + \beta = \bigcup_{\gamma < \beta} \alpha + \gamma$ if β is a non-zero limit ordinal.

Addition satisfies associativity $\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$ but is not commutative, e.g., $1 + \omega = \omega \neq \omega + 1$.

[2]`http://www.cs.tau.ac.il/~nachum/rtaloop/`

Definition 7.2. For ordinals α and β their *product* $\alpha \cdot \beta$ is defined by recursion over β as (a) $\alpha \cdot 0 = 0$, (b) $\alpha \cdot \beta = \alpha \cdot \gamma + \alpha$ if $\beta = \gamma + 1$, and (c) $\alpha \cdot \beta = \bigcup_{\gamma<\beta} \alpha \cdot \gamma$ if β is a non-zero limit ordinal.

Since $2 \cdot \omega = \omega \neq \omega \cdot 2$ multiplication is not commutative, and as $(\omega + 1) \cdot 2 = (\omega+1)+(\omega+1) = \omega+\omega+1 = \omega \cdot 2+1$ also not right-distributive, but associativity $\alpha \cdot (\beta \cdot \gamma) = (\alpha \cdot \beta) \cdot \gamma$ and left-distributivity $\alpha \cdot (\beta + \gamma) = (\alpha \cdot \beta) + (\alpha \cdot \gamma)$ hold. We mostly write αa for $\alpha \cdot a$ whenever α is an ordinal and a a finite ordinal, i.e., $a < \omega$.

Definition 7.3. For ordinals α and β, recursion over β allows to define *exponentiation* α^β as follows: (a) $\alpha^0 = 1$, (b) $\alpha^\beta = \alpha^\gamma \cdot \alpha$ if $\beta = \gamma+1$, and (c) $\alpha^\beta = \bigcup_{\gamma<\beta} \alpha^\gamma$ if β is a non-zero limit ordinal.

Examples of infinite ordinals include $\omega^1 = \omega$, $\omega\, 3 = \omega + \omega + \omega$, $\omega^2 = \omega \cdot \omega$, $\omega^{\omega+1}$, and ω^{ω^ω}. The ordinal ϵ_0 is the smallest ordinal α which satisfies $\alpha^\omega = \alpha$. Let $\mathbb{O}$ denote the class of ordinal numbers smaller than ϵ_0, $\mathbb{N}$ the ordinal numbers smaller than ω (the natural numbers), $>$ the standard order on ordinals, and $\geqslant$ its reflexive closure.

Recall that every ordinal $\alpha < \epsilon_0$ can be represented in Cantor normal form (CNF), i.e.,

$$\alpha = \omega^{\alpha_1} a_1 + \cdots + \omega^{\alpha_n} a_n \tag{7.1}$$

such that $\alpha_1 > \cdots > \alpha_n$ are in CNF as well and $a_1, \ldots, a_n \in \mathbb{N}_{>0}$. The ordinal 0 is represented as the empty sum.

Definition 7.4. Let $\alpha = \omega^{\alpha_1} a_1 + \cdots + \omega^{\alpha_n} a_n$ and $\beta = \omega^{\beta_1} b_1 + \cdots + \omega^{\beta_m} b_m$ be ordinals in CNF, and $\{\gamma_1, \ldots, \gamma_k\} = \{\alpha_1, \ldots, \alpha_n\} \cup \{\beta_1, \ldots, \beta_m\}$ such that $\gamma_1 > \cdots > \gamma_k$. The *natural sum* of α and β is defined as follows: $\alpha \oplus \beta = \omega^{\gamma_1}(a'_1 + b'_1) + \cdots + \omega^{\gamma_k}(a'_k + b'_k)$ where $a'_i = a_j$ ($b'_i = b_j$) if $\gamma_i = \alpha_j$ ($\gamma_i = \beta_j$) for some j, and $a'_i = 0$ ($b'_i = 0$) otherwise.

In contrast to standard addition, natural addition on ordinals enjoys all properties known from addition on natural numbers, e.g., $2 \oplus \omega = \omega \oplus 2 = \omega + 2$. For ordinal algebras as considered later in this article we rely critically on the fact that addition, natural addition, multiplication, and exponentiation are weakly monotone in both arguments.

We assume familiarity with term rewriting and termination [17, 172].

The derivation height of a term t with respect to a well-founded and finitely branching rewrite relation $\to_\mathcal{R}$ is defined as $\mathrm{dh}_\mathcal{R}(t) = \max\{m \mid \exists u\; t \to^m_\mathcal{R} u\}$. The derivational complexity of $\mathcal{R}$ computes the maximal derivation height of all terms up to size n and is defined as $\mathrm{dc}_{\mathcal{R}}(n) = \max\{\mathrm{dh}_{\mathcal{R}}(t) \mid |t| \leqslant n\}$.

A relative TRS $\mathcal{R}/\mathcal{S}$ is a pair of TRSs $\mathcal{R}$ and $\mathcal{S}$ with the induced rewrite relation $\rightarrow_{\mathcal{R}/\mathcal{S}} = \rightarrow_{\mathcal{S}}^{*} \cdot \rightarrow_{\mathcal{R}} \cdot \rightarrow_{\mathcal{S}}^{*}$.

We consider well-founded algebras $\mathcal{A}$ with interpretation functions $f_{\mathcal{A}}$. An interpretation function $f_{\mathcal{A}}$ is *simple* if $f_{\mathcal{A}}(a_1, \ldots, a_n) \geqslant a_i$ for all $1 \leqslant i \leqslant n$. It is *monotone* if $a > b$ implies $f_{\mathcal{A}}(\ldots, a_{i-1}, a, a_{i+1}, \ldots) > f_{\mathcal{A}}(\ldots, a_{i-1}, b, a_{i+1}, \ldots)$ and *weakly monotone* if $a > b$ implies $f_{\mathcal{A}}(\ldots, a_{i-1}, a, a_{i+1}, \ldots) \geqslant f_{\mathcal{A}}(\ldots, a_{i-1}, b, a_{i+1}, \ldots)$. An algebra is *simple/monotone/weakly monotone* if all its interpretation functions are simple/monotone/weakly monotone. An assignment α maps variables to values in the carrier of $\mathcal{A}$. By $[\alpha]_{\mathcal{A}}(t)$ we denote the interpretation of the term t based on the assignment α. A TRS $\mathcal{R}$ is *compatible* with an algebra $\mathcal{A}$ if $[\alpha]_{\mathcal{A}}(\ell) > [\alpha]_{\mathcal{A}}(r)$ for every $\ell \rightarrow r \in \mathcal{R}$ and assignment α (also written $\mathcal{R} \subseteq >_{\mathcal{A}}$). Algebras may yield termination proofs.

Theorem 7.5. *A TRS is terminating if and only if it is compatible with a well-founded monotone algebra.* □

Theorem 7.6 ([178, 209])**.** *A TRS is terminating if it is compatible with a well-founded weakly monotone simple algebra.* □

7.3. The Goodstein Sequence

In this section we present a TRS for the Goodstein sequence, whose definition requires the following key notion. Given $n > 1$, a natural number α is in *hereditary base n representation*, which we indicate by writing $(\alpha)_n$, if

$$(\alpha)_n = n^{(\alpha_k)_n} \cdot a_k + n^{(\alpha_{k-1})_n} \cdot a_{k-1} + \cdots + n^{(\alpha_0)_n} \cdot a_0 \tag{7.2}$$

such that all $(\alpha_k)_n > \cdots > (\alpha_0)_n$ are in hereditary base n representation and $0 < a_i < n$ for all $0 \leqslant i \leqslant k$. For $m > n$ we denote by $(\alpha)_n^m$ the result of replacing n by m in $(\alpha)_n$, so $(\alpha)_n^m = m^{(\alpha_k)_n^m} \cdot a_k + m^{(\alpha_{k-1})_n^m} \cdot a_{k-1} + \cdots + m^{(\alpha_1)_n^m} \cdot a_1 + a_0$ is in hereditary base m representation.

For instance, $(1)_2 = 2^0 \cdot 1$, where we drop the coefficient 1 and simply write $(1)_2 = 2^0$. Moreover, $(2)_2 = 2^1 = 2^{2^0}$ and $(5)_2 = 2^2 + 1 = 2^{2^{2^0}} + 2^0$, whereas $(5)_2^3 = 3^{3^{3^0}} + 3^0 = 28$.

Definition 7.7. The *Goodstein sequence* g_α with starting value α is defined by $g_\alpha(0) = \alpha$ and $g_\alpha(i+1) = (g_\alpha(i))_{i+2}^{i+3} - 1$ for all $i \geqslant 0$.

Example 7.8. For $\alpha = 2$ the Goodstein sequence yields

$$\begin{aligned}
g_2(0) &= 2 \\
g_2(1) &= (2)_2^3 - 1 = (2^1)_2^3 - 1 = 3^1 - 1 = 2 \\
g_2(2) &= (2)_3^4 - 1 = 2 - 1 = 1 \\
g_2(3) &= (1)_4^5 - 1 = 1 - 1 = 0
\end{aligned}$$

while for $\alpha = 5$ we obtain

$$\begin{aligned}
g_5(0) &= 5 \\
g_5(1) &= (5)_2^3 - 1 = (2^2 + 2^0)_2^3 - 1 = 3^3 + 3^0 - 1 = 27 \\
g_5(2) &= (27)_3^4 - 1 = (3^3)_3^4 - 1 = 4^4 - 1 = 255 \\
g_5(3) &= (255)_4^5 - 1 = (4^3 \cdot 3 + 4^2 \cdot 3 + 4 \cdot 3 + 3)_4^5 - 1 = 5^3 \cdot 3 + 5^2 \cdot 3 + 5 \cdot 3 + 2 \\
&= 467
\end{aligned}$$

Theorem 7.9 (Goodstein [63])**.** *For all α there exists a k such that $g_\alpha(k) = 0$.* $\square$

By $G(\alpha)$ we denote the smallest number k with this property. Totality of this function is not provable in Peano arithmetic, as shown by Kirby and Paris [96]. Cichon [31] presented a very short proof using results concerning recursion theoretic hierarchies of functions. In particular, he showed that the growth rate of G cannot be bounded by any H_α such that $\alpha < \epsilon_0$.[3]

Definition 7.10. For all $n > 1$ we define a mapping $[\cdot]_n$ to represent natural numbers in (hereditary) base n as ground terms over $\{\mathsf{c}, \mathsf{0}\}$, where c is a binary function symbol and $\mathsf{0}$ a constant. Let $(\alpha)_n$ be a natural number in hereditary base n representation as in (7.2). The term $\mathsf{c}(x, \mathsf{c}(x, \cdots \mathsf{c}(x, y) \cdots))$ containing $k \geqslant 0$ occurrences of c is denoted $\mathsf{c}^k(x, y)$. In particular, $\mathsf{c}^0(x, y) = y$. Then $[\cdot]_n$ is recursively defined such that $[0]_n = \mathsf{0}$ and

$$[\alpha]_n = \mathsf{c}^{a_0}([\alpha_0]_n, \ldots \mathsf{c}^{a_{k-1}}([\alpha_{k-1}]_n, \mathsf{c}^{a_k}([\alpha_k]_n, \mathsf{0})) \ldots)$$

Intuitively, given base n, the term $\mathsf{c}([\alpha]_n, [\beta]_n)$ represents the number $n^\alpha + \beta$, and terms contributing to the base n representation of a number are combined in increasing order. This is in contrast to (7.2), where terms are sorted in a decreasing way.

[3] Here H is the Hardy function: $H_0(n) = n + 1$, $H_{\alpha+1}(n) = H_\alpha(n + 1)$, and $H_\lambda(n) = H_{\lambda_n}(n)$ for a limit ordinal λ which is the supremum of an ordinal sequence $(\lambda_n)_{n \in \mathbb{N}}$.

Example 7.11. For $(1)_2 = 2^0$ we have $[1]_2 = \mathsf{c}(\mathsf{0},\mathsf{0})$, for $(2)_2 = 2^{2^0}$ we have $[2]_2 = \mathsf{c}(\mathsf{c}(\mathsf{0},\mathsf{0}),\mathsf{0})$, for $(7)_2 = 2^{2^{2^0}} + 2^{2^0} + 2^0$ we have $[7]_2 = \mathsf{c}(\mathsf{0},\mathsf{c}(\mathsf{c}(\mathsf{0},\mathsf{0}),\mathsf{c}(\mathsf{c}(\mathsf{c}(\mathsf{0},\mathsf{0}),\mathsf{0}))))$, and for $(7)_3 = 3^{3^0} \cdot 2 + 3^0$ we have $[7]_3 = \mathsf{c}(\mathsf{0},\mathsf{c}(\mathsf{c}(\mathsf{0},\mathsf{0}),\mathsf{c}(\mathsf{c}(\mathsf{0},\mathsf{0}),\mathsf{0})))$. Note that different numbers over different bases might be represented by the same term, for instance $(2)_2 = (3)_3 = \mathsf{c}(\mathsf{c}(\mathsf{0},\mathsf{0}),\mathsf{0})$.

The following TRS $\mathcal{G}$ works on inputs of the form $[\cdot]_n$ to model g_α. Its definition is inspired by Touzet's encoding of the Hydra battle [178] (see Example 7.30).

Definition 7.12. Consider the following TRS $\mathcal{G}$ over a signature consisting of unary function symbols $\bullet$, $[]$, $\circ$ and binary function symbols f, h, in addition to $\mathsf{0}$ and c:

$$\begin{aligned}
[]\circ x &\to \circ\, []\, x && \text{(A1)}\\
\bullet\, []\, x &\to []\bullet\bullet\, x && \text{(A2)}\\
\circ\, x &\to \bullet\, []\, x && \text{(A3)}\\
\mathsf{c}(\mathsf{0},x) &\to \circ\, x && \text{(B1)}\\
\bullet\,\mathsf{c}(\mathsf{c}(x,y),z) &\to \bullet\,\mathsf{f}(\mathsf{c}(x,y),z) && \text{(B2)}\\
\bullet\,\mathsf{f}(\mathsf{0},x) &\to \circ\, x && \text{(C1)}\\
\bullet\,\mathsf{f}(\mathsf{c}(x,y),z) &\to \mathsf{h}(\bullet\,\mathsf{f}(x,y),\bullet\bullet\,\mathsf{f}(\mathsf{f}(x,y),z)) && \text{(C2)}\\
\bullet\,\mathsf{h}(x,y) &\to \mathsf{h}(\bullet\, x,\bullet\bullet\,\mathsf{c}(x,y)) && \text{(D1)}\\
\mathsf{h}(x,y) &\to \circ\, y && \text{(D2)}\\
\bullet\,\mathsf{f}(x,y) &\to \mathsf{f}(\bullet\, x,y) && \text{(E1)}\\
\bullet\,\mathsf{c}(x,y) &\to \mathsf{c}(\bullet\, x,\bullet\, y) && \text{(E2)}\\
\bullet\, x &\to x && \text{(E3)}\\
\circ\, x &\to x && \text{(E4)}
\end{aligned}$$

The basic idea of the encoding is to perform a step in the Goodstein sequence as follows. The current base n and sequence element α are encoded as $\bullet\, []^n\, [\alpha]_n$. Using rule (A2), the symbol $\bullet$ is repeatedly duplicated while moving over the $[]$'s, until a term of the form $[]^n\, \bullet^{2^n}\, [\alpha]_n$ is reached. The copies of $\bullet$ can move to places (using rules (E1) and (E2)) in $[\alpha]_n$ where changes are required to turn $[\alpha]_n$ into $[\beta]_{n+1}$ for $\beta = (\alpha)_n^{n+1} - 1$. This is achieved using rules (B1) – (D2) and the symbols f and h for auxiliary purposes, and produces at least one $\circ$ symbol which can then travel back up the $[]$'s using (A1). Finally, the base is increased by (A3), which yields a term $\bullet\, []^{n+1}\, [\beta]_{n+1}$. Whenever there are too many $\bullet$ or $\circ$ symbols, they are removed with rules (E3) and (E4).

This idea is made precise in the following theorem, according to which $\mathcal{G}$ simulates for any starting value the computation of the Goodstein sequence. In particular, termination of $\mathcal{G}$ (Theorem 7.15) enforces for any $\alpha \in \mathbb{N}$ the existence of a k with $\bullet\,[\!]^2\,[\alpha]_2 \to_{\mathcal{G}}^{*} \bullet\,[\!]^k\,[0]_k$, and thus implies Theorem 7.9.

Theorem 7.13. *Let $\alpha, n \in \mathbb{N}$ such that $\alpha > 0$ and $n > 1$. Then it follows that $\bullet\,[\!]^n\,[\alpha]_n \to_{\mathcal{G}}^{+} \bullet\,[\!]^{n+1}\,[\beta]_{n+1}$ where $\beta = (\alpha)_n^{n+1} - 1$.*

The proof of this result requires some auxiliary facts about $\mathcal{G}$.

Lemma 7.14.

(a) *$\bullet^n\,\mathsf{h}(s,t) \to_{\mathcal{G}}^{+} \circ\,\mathsf{c}^n(\bullet^n\,s, \bullet^{2n}\,t)$ for all terms s and t.*

(b) *Let $\alpha, \beta \in \mathbb{N}$ and $n \in \mathbb{N}$ such that $n > 1$, $\beta + n^\alpha$ is positive, $s = [\alpha]_n$ and $t = [\beta]_n$. Then $\bullet^n\,\mathsf{f}(s,t) \to_{\mathcal{G}}^{+} \circ\,u$ where $u = [\beta + n^\alpha - 1]_n$.*

Proof.

(a) By induction on n. If $n = 0$ then $\mathsf{h}(s,t) \to_{\mathcal{G}} \circ\,t$ in a single step using (D2). If $n > 0$ then

$$\begin{aligned}
\bullet^{n+1}\,\mathsf{h}(s,t) &\to_{\mathcal{G}} \bullet^n\,\mathsf{h}(\bullet\,s, \bullet\bullet\,\mathsf{c}(s,t)) && \text{(D1)}\\
&\to_{\mathcal{G}}^{+} \circ\,\mathsf{c}^n(\bullet^{n+1}\,s, \bullet^{2(n+1)}\,\mathsf{c}(s,t)) && (\star)\\
&\to_{\mathcal{G}}^{+} \circ\,\mathsf{c}^n(\bullet^{n+1}\,s, \mathsf{c}(\bullet^{2(n+1)}\,s, \bullet^{2(n+1)}\,t)) && \text{(E2)}\\
&\to_{\mathcal{G}}^{+} \circ\,\mathsf{c}^n(\bullet^{n+1}\,s, \mathsf{c}(\bullet^{n+1}\,s, \bullet^{2(n+1)}\,t)) && \text{(E3)}\\
&= \circ\,\mathsf{c}^{n+1}(\bullet^{n+1}\,s, \bullet^{2(n+1)}\,t)
\end{aligned}$$

where $(\star)$ applies the induction hypothesis.

(b) By induction on α. If $\alpha = 0$ then $[\alpha]_n = \mathsf{0}$ and $\bullet^n\,\mathsf{f}(\mathsf{0},t) \to_{\mathcal{G}} \bullet^{n-1} \circ\,t \to_{\mathcal{G}}^{*} \circ\,t$ using rules (C1) and (E3). Since $\beta + n^0 - 1 = \beta$ and $t = [\beta]_n$ the claim holds. If $\alpha > 0$ then $[\alpha]_n = \mathsf{c}(s',t')$ and $s' = [\gamma]_n$ and $t' = [\delta]_n$ for some $\gamma, \delta \in \mathbb{N}$, so $\alpha = \delta + n^\gamma$. We have

$$\begin{aligned}
\bullet^n\,\mathsf{f}(\mathsf{c}(s',t'),t) &\to_{\mathcal{G}} \bullet^{n-1}\,\mathsf{h}(\bullet\mathsf{f}(s',t'), \bullet\bullet\,\mathsf{f}(\mathsf{f}(s',t'),t)) && \text{(C2)}\\
&\to_{\mathcal{G}}^{+} \circ\,\mathsf{c}^{n-1}(\bullet^n\,\mathsf{f}(s',t'), \bullet^{2n}\,\mathsf{f}(\mathsf{f}(s',t'),t)) && \text{(a)}\\
&\to_{\mathcal{G}}^{*} \circ\,\mathsf{c}^{n-1}(\bullet^n\,\mathsf{f}(s',t'), \bullet^n\,\mathsf{f}(\bullet^n\,\mathsf{f}(s',t'),t)) && \text{(E1)}\\
&\to_{\mathcal{G}}^{+} \circ\,\mathsf{c}^{n-1}(\circ\,w, \bullet^n\,\mathsf{f}(\circ\,w,t)) && (\star)\\
&\to_{\mathcal{G}}^{+} \circ\,\mathsf{c}^{n-1}(w, \bullet^n\,\mathsf{f}(w,t)) && \text{(E4)}\\
&\to_{\mathcal{G}}^{+} \circ\,\mathsf{c}^{n-1}(w, \circ\,w') && (\star\star)\\
&\to_{\mathcal{G}} \circ\,\mathsf{c}^{n-1}(w, w') && \text{(E4)}
\end{aligned}$$

where in ($\star$) we apply the induction hypothesis since $\gamma < \alpha$ and so we obtain a term $w = [\delta + n^\gamma - 1]_n$. Since $\delta + n^\gamma - 1 < \alpha$, we can apply the induction hypothesis again in ($\star\star$), which yields a term w' such that $w' = \left[\beta + n^{\delta+n^\gamma-1} - 1\right]_n$. Let $\nu = \delta + n^\gamma - 1$. For the term $v = \mathsf{c}^{n-1}(w, w')$ we thus have

$$v = [\beta + n^\nu \cdot (n-1) + n^\nu - 1]_n = \left[\beta + n^{\nu+1} - 1\right]_n = [\beta + n^\alpha - 1]_n . \quad \square$$

Proof of Theorem 7.13. Since $\alpha > 0$, we have $[\alpha]_n = \mathsf{c}(s,t)$ for some terms s and t. We apply case analysis on s. If $s = \mathsf{0}$ then $t = [\alpha - 1]_n$ and we have

$$\begin{aligned}
\bullet\, [\,]^n\, \mathsf{c}(\mathsf{0}, t) &\to_{\mathcal{G}} [\,]^n\, \mathsf{c}(\mathsf{0}, t) && \text{(E3)}\\
&\to_{\mathcal{G}} [\,]^n \circ t && \text{(B1)}\\
&\to_{\mathcal{G}}^+ \circ [\,]^n\, t && \text{(A1)}\\
&\to_{\mathcal{G}} \bullet [\,]^{n+1}\, t && \text{(A3)}
\end{aligned}$$

Otherwise, $s = \mathsf{c}(u,v)$ so let $\mathsf{c}(u,v) = [\gamma]_n$ and $t = [\delta]_n$ for some $\gamma, \delta \in \mathbb{N}$. There is the following rewrite sequence:

$$\begin{aligned}
\bullet\, [\,]^n\, \mathsf{c}(\mathsf{c}(u,v), t) &\to_{\mathcal{G}}^+ [\,]^n \bullet^{2^n} \mathsf{c}(\mathsf{c}(u,v), t) && \text{(A2)}\\
&\to_{\mathcal{G}}^* [\,]^n \bullet^{n+1} \mathsf{c}(\mathsf{c}(u,v), t) && \text{(E3)}\\
&\to_{\mathcal{G}}^* [\,]^n \bullet^{n+1} \mathsf{f}(\mathsf{c}(u,v), t) && \text{(B2)}\\
&\to_{\mathcal{G}}^+ [\,]^n \circ w && (\star)\\
&\to_{\mathcal{G}}^+ \circ [\,]^n\, w && \text{(A1)}\\
&\to_{\mathcal{G}} \bullet [\,]^{n+1}\, w && \text{(A3)}
\end{aligned}$$

where ($\star$) applies Lemma 7.14(b), according to which $w = [\delta + (n+1)^\gamma - 1]_{n+1}$. $\square$

Theorem 7.15. *The TRS $\mathcal{G}$ is terminating.*

Proof. We show termination of $\mathcal{G}$ by employing Theorem 7.6. Consider the following algebra $\mathcal{A}$ over the well-founded domain $\mathbb{O} \times \mathbb{N} \times \mathbb{N}$:

$$\begin{aligned}
\mathsf{0}_{\mathcal{A}} &= (0,0,0) & [\,]_{\mathcal{A}}(x,m,n) &= (x, 2m+2, n)\\
\mathsf{c}_{\mathcal{A}}((x,m,n),(y,k,l)) &= (\omega^x \oplus y + 1, 0, 0) & \circ_{\mathcal{A}}(x,m,n) &= (x, 2m+3, n)\\
\mathsf{f}_{\mathcal{A}}((x,m,n),(y,k,l)) &= (\omega^x \oplus y, 0, 0) & \bullet_{\mathcal{A}}(x,m,n) &= (x, m, n+m+1)\\
\mathsf{h}_{\mathcal{A}}((x,m,n),(y,k,l)) &= (y + \omega^{x+1}, 0, 0)
\end{aligned}$$

Note that $\mathcal{A}$ is simple and weakly monotone, and it strictly orients all rules of $\mathcal{G}$:

$$\begin{aligned}
(x, 4m + 8, n) &> (x, 4m + 7, n) && \text{(A1)}\\
(x, 2m + 2, 2m + n + 3) &> (x, 2m + 2, 2m + n + 2) && \text{(A2)}\\
(x, 2m + 3, n) &> (x, 2m + 2, n + 2m + 3) && \text{(A3)}\\
(x + 2, 0, 0) &> (x, 2m + 3, n) && \text{(B1)}\\
(\omega^{\omega^x \oplus y+1} \oplus z + 1, 0, 1) &> (\omega^{\omega^x \oplus y+1} \oplus z, 0, 1) && \text{(B2)}\\
(x + 1, 0, 1) &> (x, 2m + 3, n) && \text{(C1)}\\
(\omega^{\omega^x \oplus y+1} \oplus z, 0, 1) &> (z + \omega^{\omega^x \oplus y+1}, 0, 0) && \text{(C2)}\\
(y + \omega^{x+1}, 0, 1) &> (y + \omega^{x+1}, 0, 0) && \text{(D1)}\\
(y + \omega^{x+1}, 0, 0) &> (y, 2k + 3, l) && \text{(D2)}\\
(\omega^x \oplus y, 0, 1) &> (\omega^x \oplus y, 0, 0) && \text{(E1)}\\
(\omega^x \oplus y + 1, 0, 1) &> (\omega^x \oplus y + 1, 0, 0) && \text{(E2)}\\
(x, m, n + m + 1) &> (x, m, n) && \text{(E3)}\\
(x, 2m + 3, n) &> (x, m, n) && \text{(E4)}
\end{aligned}$$

Hence $\mathcal{G}$ is terminating. Note that rule (C2) has a weak decrease in its first component since ordinal addition might consume its left argument but natural addition does not, i.e., $\alpha \oplus \beta = \beta \oplus \alpha \geqslant \beta + \alpha$ for all ordinals α and β in CNF. $\square$

The proof of Theorem 7.15 (again inspired by the termination proof in [178]) lexicographically combines ordinal with linear polynomial interpretations. However, we remark that weak monotonicity of the lexicographic product does not follow from weak monotonicity of the single interpretations (cf. Example 7.26). Still, the search for suitable interpretation functions can be automated (see Section 7.4.2).

7.4. Automation of Ordinal Algebras

In order to automate the search for suitable ordinal interpretations, we restrict ourselves to interpretation functions of a certain shape (see Definition 7.16). In Section 7.4.1 we show how for a given algebra with interpretation functions of this shape one can encode whether the interpretation of one term is larger than that of another term. In contrast to other termination criteria, ordinal arithmetic (non-commutative, expressions may be consumed) significantly complicates the encoding. Section 7.4.2 elaborates on implementation issues needed for a successful automation, where we also explain how to find suitable coefficients for the interpretation functions. Section 7.4.3 considers different encodings of Hydra battles where also the limitations of the approach are discussed.

In the sequel we consider ordinal expressions of the following shape. By $\overline{x}$ we abbreviate $x_1, \ldots, x_n$.

Definition 7.16. A *restricted ordinal expression* (*ROE*) over variables $\overline{x}$ is either 0 or[4]

$$\sum_{1 \leqslant i \leqslant n} x_i f_i + \omega^{f'(\overline{x})} f_\omega \oplus \bigoplus_{1 \leqslant i \leqslant n} x_i \hat{f}_i \oplus f_0 \tag{7.3}$$

where $f_0, f_1, \ldots, f_n, \hat{f}_1, \ldots, \hat{f}_n, f_\omega$ are natural numbers and $f'(\overline{x})$ is an ROE over $\overline{x}$. The *depth* of an ROE is the height of the tower of ω's. An *ROE algebra* is an algebra $\mathcal{O}$ where for every n-ary function symbol f the interpretation function $f_\mathcal{O}$ is an ROE over $\overline{x}$.

7.4.1. Encodings

Let $f(\overline{x})$ and $g(\overline{x})$ be ROEs of the form

$$\begin{aligned} f(\overline{x}) &= \sum_{1 \leqslant i \leqslant n} x_i f_i + \omega^{f'(\overline{x})} f_\omega \oplus \bigoplus_{1 \leqslant i \leqslant n} x_i \hat{f}_i \oplus f_0 \\ g(\overline{x}) &= \sum_{1 \leqslant i \leqslant n} x_i g_i + \omega^{g'(\overline{x})} g_\omega \oplus \bigoplus_{1 \leqslant i \leqslant n} x_i \hat{g}_i \oplus g_0 \end{aligned} \tag{7.4}$$

We assume that these expressions depend on the same variables $\overline{x}$ (otherwise the respective coefficients can be set to 0), and that variables appear in the same order. We first encode some auxiliary properties of ROEs.

Useful Abbreviations

Let $\text{zero}(f(\overline{x}))$ be true if and only if $f(\overline{x}) = 0$ or all of f_0, f_i, $\hat{f}_i$ and f_ω are 0. Let $c_i = \max(f_i, g_i)$ for all $i \in \{0, \ldots, n, \omega\}$. An upper bound $\text{omax}(f, g)(\overline{x})$ is then given by $\text{omax}(f, 0)(\overline{x}) = \text{omax}(0, f)(\overline{x}) = f(\overline{x})$ and

$$\text{omax}(f, g)(\overline{x}) = \sum_{1 \leqslant i \leqslant n} x_i c_i + \omega^{\text{omax}(f', g')(\overline{x})} c_\omega \oplus \bigoplus_{1 \leqslant i \leqslant n} x_i \max(\hat{f}_i, \hat{g}_i) \oplus c_0$$

otherwise. For instance, if $f(\overline{x}) = x_1 + \omega^{x_2+1} \oplus x_3$ and $g(\overline{x}) = \omega^{x_1} 2 \oplus x_2 + 1$ then $\text{omax}(f, g)(\overline{x}) = x_1 + \omega^{x_1+x_2+1} 2 \oplus x_2 \oplus x_3 + 1$. Clearly, for all assignments α we have that $[\alpha](f(\overline{x})) \leqslant [\alpha](\text{omax}(f, g)(\overline{x}))$ and $[\alpha](g(\overline{x})) \leqslant [\alpha](\text{omax}(f, g)(\overline{x}))$.

[4]To enhance readability we drop parentheses in expressions of the form $x + y \oplus z$, which are to be read as $(x + y) \oplus z$ rather than $x + (y \oplus z)$. Note that these expressions are in general not equivalent, e.g., $(1 + 0) \oplus \omega = \omega + 1$ but $1 + (0 \oplus \omega) = \omega$.

Whether a variable x_i *contributes* to the value of $f(\overline{x})$ can be recursively encoded as follows:

$$\mathrm{con}_i(f(\overline{x})) = \begin{cases} \bot & \text{if } f(\overline{x}) = 0 \\ f_i > 0 \vee \hat{f}_i > 0 \vee (\mathrm{con}_i(f'(\overline{x})) \wedge f_\omega > 0) & \text{otherwise} \end{cases}$$

If $f(\overline{x})$ and $g(\overline{x})$ are defined as above then $\mathrm{con}_i(f(\overline{x})) = \mathrm{con}_j(g(\overline{x})) = \top$ for all $1 \leqslant i \leqslant 3$ and $1 \leqslant j \leqslant 2$, but $\mathrm{con}_3(g(\overline{x})) = \bot$.

Comparisons

Consider ROEs $f(\overline{x})$ and $g(\overline{x})$ as in (7.4). We want to derive sufficient (checkable) conditions such that $[\alpha](f(\overline{x})) > [\alpha](g(\overline{x}))$ for all assignments α. The following example shows that whether one ROE is larger than another one significantly depends on the assignment.

Example 7.17. Consider $x_1 + x_2$ and $x_2 + x_1$. Let α be an assignment such that $\alpha(x_1) = \omega$ and $\alpha(x_2) = 1$. Then $[\alpha](x_1 + x_2) = \omega + 1 > \omega = 1 + \omega = [\alpha](x_2 + x_1)$. Conversely we have $[\beta](x_1 + x_2) = 1 + \omega = \omega < \omega + 1 = [\beta](x_2 + x_1)$ when $\beta(x_1) = 1$ and $\beta(x_2) = \omega$.

We use the following underapproximation to check whether $[\alpha](f(\overline{x})) > [\alpha](g(\overline{x}))$ for all assignments α, which is a tradeoff between accuracy and efficiency.

Definition 7.18. Let $f(\overline{x})$ and $g(\overline{x})$ be ROEs as in (7.4).

$$\begin{aligned} [f(\overline{x}) \geqslant g(\overline{x})] &= [f(\overline{x}) \geqslant_0 g(\overline{x})] \wedge \bigwedge_{1 \leqslant i \leqslant n} [f(\overline{x}) \geqslant_i g(\overline{x})] \\ [f(\overline{x}) \geqslant_0 g(\overline{x})] &= ([f'(\overline{x}) >_0 g'(\overline{x})] \wedge f_\omega > 0) \vee \\ &\quad\ ([f'(\overline{x}) \geqslant_0 g'(\overline{x})] \wedge f_\omega \geqslant g_\omega \wedge f_0 \geqslant g_0) \vee \\ &\quad\ (g_\omega = 0 \wedge f_0 \geqslant g_0) \end{aligned}$$

$$
\begin{aligned}
[f(\overline{x}) \geqslant_i g(\overline{x})] = {} & \neg\mathrm{con}_i(g(\overline{x})) \vee && \text{(a)}\\
& ([f'(\overline{x}) \geqslant_i g'(\overline{x})] \wedge f_\omega \geqslant g_\omega \wedge g_i = 0 \wedge \hat{g}_i = 0) \vee && \text{(b)}\\
& (\mathrm{con}_i(\omega^{f'(\overline{x})} f_\omega) \wedge \neg\mathrm{con}_i(\omega^{g'(\overline{x})} g_\omega)) \vee && \text{(c)}\\
& (\mathrm{con}_i(\omega^{f'(\overline{x})} f_\omega) \wedge [f'(\overline{x}) \geqslant_i g'(\overline{x})] \wedge f_\omega > g_\omega) \vee && \text{(d)}\\
& (\mathrm{con}_i(\omega^{f'(\overline{x})} f_\omega) \wedge [f'(\overline{x}) \geqslant_i g'(\overline{x})] \wedge f_\omega = g_\omega \wedge \hat{f}_i \geqslant \hat{g}_i) \vee && \text{(e)}\\
& (\neg\mathrm{con}_i(\omega^{g'(\overline{x})} g_\omega) \wedge \hat{f}_i \geqslant \hat{g}_i \wedge f_i + \hat{f}_i \geqslant g_i + \hat{g}_i) \vee && \text{(f)}\\
& ((\mathrm{zero}(g'(\overline{x})) \vee g_\omega = 0) \wedge f_i + \hat{f}_i \geqslant g_i + \hat{g}_i) && \text{(g)}\\
[f(\overline{x}) > g(\overline{x})] = {} & [f(\overline{x}) \geqslant g(\overline{x})] \wedge [f(\overline{x}) >_0 g(\overline{x})]\\
[f(\overline{x}) >_0 g(\overline{x})] = {} & ([f'(\overline{x}) >_0 g'(\overline{x})] \wedge f_\omega > 0) \vee\\
& ([f'(\overline{x}) \geqslant_0 g'(\overline{x})] \wedge f_\omega \geqslant g_\omega \wedge f_0 > g_0) \vee\\
& (g_\omega = 0 \wedge f_0 > g_0)
\end{aligned}
$$

Here $[f(\overline{x}) >_0 g(\overline{x})]$ ($[f(\overline{x}) \geqslant_0 g(\overline{x})]$) encodes that the constant part in $f(\overline{x})$ is greater (or equal) than the constant part in $g(\overline{x})$, whereas $[f(\overline{x}) \geqslant_i g(\overline{x})]$ encodes that the coefficients of the variable x_i in $f(\overline{x})$ are greater than or equal to the respective coefficients in $g(\overline{x})$. The last disjunct in the definition of $[f(\overline{x}) >_0 g(\overline{x})]$ was added to the earlier version of our encoding [193]; it is essential to handle the last rule of the TRS $\mathcal{W}_3'$ in Example 7.31. Our comparisons are (much) more involved than the absolute positiveness approach [83] for polynomials because of ordinal arithmetic. We illustrate the different cases in the encoding of $\geqslant_i$ in the following example.

Example 7.19. Case (a) yields $[\omega^{x_1+x_2} \geqslant_1 \omega^{x_2}]$ while (b) admits $[\omega^{x_1 2}3 \geqslant_1 \omega^{x_1}3]$. From (c) validity of $[\omega^{x_1}2 \geqslant_1 x_1 3]$ is obtained and $[\omega^{x_1}2 \geqslant_1 \omega^{x_1}1 \oplus x_1 5]$ is due to (d). Case (e) obviously allows $[\omega^{x_1}2 \oplus x_1 2 \geqslant_1 \omega^{x_1}2 \oplus x_1 1]$ but also $[\omega^{x_1} \geqslant_1 x_1 10 + \omega^{x_1}]$. Case (f) implies $[x_1 2 + \omega^{x_2} \oplus x_1 3 \geqslant_1 x_1 3 + \omega^{x_2} \oplus x_1 2]$. Finally, $[x_1 4 + \omega^{x_2} \oplus x_1 1 \geqslant_1 x_1 2 \oplus x_1 3]$ is ensured by (g). It is not hard to check that for all these example ROEs satisfying $[f(x_1,x_2) \geqslant_1 g(x_1,x_2)]$ we indeed have $[\alpha](f(x_1,x_2)) \geqslant [\alpha](g(x_1,x_2))$ for any assignment α (though additional constraints are required to ensure this). In the example for case (f), the test $\hat{f}_1 \geqslant \hat{g}_1$ is required if $\omega^{\alpha(x_2)}$ consumes the preceding $\alpha(x_1)2$ (and hence $\alpha(x_1)3$) for some assignment α. Otherwise the test $f_1 + \hat{f}_1 \geqslant g_1 + \hat{g}_1$ is required. For case (g), if for some α the term $\omega^{\alpha(x_2)}$ consumes $\alpha(x_1)4$ then it also dominates $\alpha(x_1)2$. Otherwise we need the test $f_1 + \hat{f}_1 \geqslant g_1 + \hat{g}_1$.

Clearly, the encoding of $\geqslant$ is only an approximation. E.g., $[\omega^{x_1+1} \geqslant_1 \omega^{x_1}2]$ is not valid, despite the fact that $\omega^{\alpha(x_1)+1} > \omega^{\alpha(x_1)}2$ for any α. While it is straightforward to extend Definition 7.18(b) accordingly for this particular case, we do not strive for a precise encoding, which seems out of reach for practical applications.

The encodings of comparisons are sound.

Lemma 7.20. *Let $f(\overline{x})$ and $g(\overline{x})$ be ROEs as in* (7.4).

(a) *If $[f(\overline{x}) > g(\overline{x})]$ then $[\alpha](f(\overline{x})) > [\alpha](g(\overline{x}))$ for all assignments α.*

(b) *If $[f(\overline{x}) \geqslant g(\overline{x})]$ then $[\alpha](f(\overline{x})) \geqslant [\alpha](g(\overline{x}))$ for all assignments α.*

Proof. Each of the disjunctions (a)–(g) in Definition 7.18 is a sound criterion for the comparison $[\alpha](f(\overline{x})) \geqslant_i [\alpha](g(\overline{x}))$ for all $1 \leqslant i \leqslant n$. □

Composition

In contrast to e.g. polynomial interpretations, ROEs are not closed under scalar multiplication and standard/natural addition (cf. Example 7.21), and thus also not under composition. Hence we cannot compute an ROE corresponding to the interpretation of a term t with respect to an ROE algebra $\mathcal{O}$. Instead, we define ROEs $\mu(t)$ and $\nu(t)$ to under- and overapproximate $t_{\mathcal{O}}$. To this end we present in Definition 7.22 bounds for the results of ordinal arithmetic operations (based on the algorithms given in [119] for ordinals in CNF) and demonstrate them in Example 7.23 before Lemma 7.24 shows their soundness.

Example 7.21.

(a) Consider the ROEs $x+1$ and 2. If $\alpha(x) < \omega$ then $[\alpha]((x+1)\cdot 2) = [\alpha](x2+2)$ but $[\alpha]((x+1)\cdot 2) = [\alpha](x2+1)$ otherwise.

(b) Consider the ROEs ω^2 and ω^3. There is no ROE for $\omega^2 \oplus \omega^3$.

(c) Consider the ROEs $x\oplus 1$ and y. If $\alpha(y) < \omega$ then $[\alpha]((x\oplus 1)+y) = [\alpha](x+y+1)$ but $[\alpha]((x \oplus 1) + y) = [\alpha](x + y)$ otherwise.

Definition 7.22. Let $f(\overline{x})$ and $g(\overline{x})$ be ROEs as in (7.4).

(a) For $a \in \mathbb{N}$, let $(f \cdot_\mu a)(\overline{x}) = (f \cdot_\nu a)(\overline{x}) = 0$ if $a = 0$ or $f(\overline{x}) = 0$, and otherwise

$$(f \cdot_\mu a)(\overline{x}) = \sum_{1\leqslant i\leqslant n} x_i f_i + \omega^{f'(\overline{x})}(f_\omega \cdot a) \oplus \bigoplus_{1\leqslant i\leqslant n} x_i(\hat{f}_i \cdot a) \oplus (f_0 \cdot a)$$

$$(f \cdot_\nu a)(\overline{x}) = \sum_{1\leqslant i\leqslant n} x_i (f_i \cdot a) + \omega^{f'(\overline{x})}(f_\omega \cdot a) \oplus \bigoplus_{1\leqslant i\leqslant n} x_i(\hat{f}_i \cdot a) \oplus (f_0 \cdot a)$$

(b) Let $(f \oplus_\mu g)(\overline{x}) = (f \oplus_\nu g)(\overline{x}) = g(\overline{x})$ if $f(\overline{x}) = 0$ and similarly we set $(f \oplus_\mu g)(\overline{x}) = (f \oplus_\nu g)(\overline{x}) = f(\overline{x})$ if $g(\overline{x}) = 0$. Otherwise, let s_i and t_i abbreviate $\mathrm{con}_i(\omega^{f'(\overline{x})} f_\omega)\ ?\ 0 : 1$ and $\mathrm{con}_i(\omega^{g'(\overline{x})} g_\omega)\ ?\ 0 : 1$, where $b\ ?\ t : e$ encodes "if b then t else e". Let

$$(h(\overline{x}), h_\omega) = \begin{cases} (f'(\overline{x}), f_\omega + 1) & \text{if } [\omega^{f'(\overline{x})} f_\omega > \omega^{g'(\overline{x})} g_\omega] \\ (g'(\overline{x}), g_\omega + 1) & \text{if } [\omega^{g'(\overline{x})} g_\omega > \omega^{f'(\overline{x})} f_\omega] \\ (\mathrm{omax}(f', g')(\overline{x}), f_\omega + g_\omega) & \text{otherwise} \end{cases}$$

and $(k(\overline{x}), k_\omega) = [\omega^{f'(\overline{x})} f_\omega > \omega^{g'(\overline{x})} g_\omega]\ ?\ (f'(\overline{x}), f_\omega) : (g'(\overline{x}), g_\omega)$. Then

$$(f \oplus_\mu g)(\overline{x}) = \sum_{1 \leqslant i \leqslant n} x_i \max(f_i s_i, g_i t_i) + \omega^{k(\overline{x})} k_\omega \oplus \bigoplus_{1 \leqslant i \leqslant n} x_i(\hat{f}_i + \hat{g}_i) \oplus (f_0 + g_0)$$

$$\begin{aligned}(f \oplus_\nu g)(\overline{x}) = \mathrm{nat}(g(\overline{x}))\ ?\ & \sum_{1 \leqslant i \leqslant n} x_i f_i + \omega^{f'(\overline{x})} f_\omega \oplus \bigoplus_{1 \leqslant i \leqslant n} x_i(\hat{f}_i + \hat{g}_i) \oplus (f_0 + g_0) : \\ \mathrm{nat}(f(\overline{x}))\ ?\ & \sum_{1 \leqslant i \leqslant n} x_i g_i + \omega^{g'(\overline{x})} g_\omega \oplus \bigoplus_{1 \leqslant i \leqslant n} x_i(\hat{f}_i + \hat{g}_i) \oplus (f_0 + g_0) : \\ & \omega^{h(\overline{x})} h_\omega \oplus \bigoplus_{1 \leqslant i \leqslant n} x_i(\hat{f}_i + \hat{g}_i + g_i t_i + f_i s_i) \oplus (f_0 + g_0)\end{aligned}$$

Here $\mathrm{nat}(f(\overline{x}))$ abbreviates $f_1 = 0 \wedge \cdots \wedge f_n = 0 \wedge f_\omega = 0$, and similarly for $g(\overline{x})$. This definition of $(f \oplus_\nu g)(\overline{x})$ allows for a more precise encoding compared to the version in [193]. In particular, a tighter upper bound is obtained for the cases where $f(\overline{x})$ or $g(\overline{x})$ are just linear polynomials (i.e., where $\mathrm{nat}(f(\overline{x}))$ or $\mathrm{nat}(g(\overline{x}))$ is true).

(c) Let $(f +_\mu g)(\overline{x}) = (f +_\nu g)(\overline{x}) = g(\overline{x})$ if $f(\overline{x}) = 0$ and similarly $(f +_\mu g)(\overline{x}) = (f +_\nu g)(\overline{x}) = f(\overline{x})$ if $g(\overline{x}) = 0$. Otherwise, we define lower and upper bounds for $f(\overline{x}) + g(\overline{x})$ by distinguishing different cases using if-then-else expressions:

$$(f +_\mu g)(\overline{x}) = [\omega^{f'(\overline{x})} f_\omega > \omega^{g'(\overline{x})} g_\omega]\ ?\ f(\overline{x}) : \Bigg(\sum_{1 \leqslant i \leqslant n} (g_i = 0\ ?\ x_i f_i : 0) + g(\overline{x})\Bigg)$$

$$\begin{aligned}(f +_\nu g)(\overline{x}) = [\omega^{g'(\overline{x})} g_\omega > \omega^{f'(\overline{x})} f_\omega]\ ?\ \phi_1 :& \\ \big([\omega^{f'(\overline{x})} f_\omega > \omega^{g'(\overline{x})} g_\omega]\ ?\ \phi_2 : (f \oplus_\nu g)(\overline{x})\big)&\end{aligned}$$

where

$$\phi_1 = \sum_{1 \leqslant i \leqslant n} x_i(f_i s_i t_i + \hat{f}_i t_i u + g_i t_i) + \omega^{g'(\overline{x})} g_\omega \oplus \bigoplus_{1 \leqslant i \leqslant n} x_i(\hat{f}_i t_i (1 - u) + \hat{g}_i) \oplus c_0$$

$$\phi_2 = \sum_{1 \leqslant i \leqslant n} x_i f_i s_i + \omega^{f'(\overline{x})} (f_\omega + 1) \oplus \bigoplus_{1 \leqslant i \leqslant n} x_i(\hat{f}_i t_i + g_i t_i + \hat{g}_i) \oplus c_0$$

with $c_0 = ([g'(\overline{x}) > 0] \wedge g_\omega > 0)\ ?\ g_0 : f_0 + g_0$ and u is 1 if all $f_i s_i t_i$ are zero and at most one of $\hat{f}_i t_i$ is greater zero and 0 otherwise.

(d) Definitions (a)–(c) can be used to inductively set lower and upper bounds for the composition $f(\overline{g})(\overline{x}) = f(g_1(\overline{x}), \ldots, g_n(\overline{x}))$. We write $\sum^{\mu}_{1 \leqslant i \leqslant n} h_i$ to abbreviate $h_1 +_\mu \cdots +_\mu h_n$, and use similar shorthands for $\oplus$ and ν. We set

$$f(\overline{g})_\mu(\overline{x}) = \sum^{\mu}_{1 \leqslant i \leqslant n} g_i(\overline{x}) \cdot_\mu f_i +_\mu \omega^{f'(\overline{g})_\mu(\overline{x})} f_\omega \oplus_\mu \bigoplus^{\mu}_{1 \leqslant i \leqslant n} g_i(\overline{x}) \cdot_\mu \hat{f}_i \oplus_\mu f_0$$

$$f(\overline{g})_\nu(\overline{x}) = \sum^{\nu}_{1 \leqslant i \leqslant n} g_i(\overline{x}) \cdot_\nu f_i +_\nu \omega^{f'(\overline{g})_\nu(\overline{x})} f_\omega \oplus_\nu \bigoplus^{\nu}_{1 \leqslant i \leqslant n} g_i(\overline{x}) \cdot_\nu \hat{f}_i \oplus_\nu f_0$$

(e) Let t be a term, and $\mathcal{O}$ be an ROE algebra. By induction on the term structure we define ROEs $\mu_{\mathcal{O}}(t)$ and $\nu_{\mathcal{O}}(t)$ such that

$$\mu_{\mathcal{O}}(t) = \begin{cases} t & \text{if } t \in \mathcal{V} \\ f_{\mathcal{O}}(\mu_{\mathcal{O}}(t_1), \ldots, \mu_{\mathcal{O}}(t_n))_\mu & \text{otherwise} \end{cases}$$

$$\nu_{\mathcal{O}}(t) = \begin{cases} t & \text{if } t \in \mathcal{V} \\ f_{\mathcal{O}}(\nu_{\mathcal{O}}(t_1), \ldots, \nu_{\mathcal{O}}(t_n))_\nu & \text{otherwise} \end{cases}$$

The following example illustrates these definitions of upper and lower bounds for ROE arithmetic.

Example 7.23.

(a) Consider the ROE $f(\overline{x}) = x_1 + x_2$. Then $(f \cdot_\mu 2)(\overline{x}) = x_1 + x_2$ and $(f \cdot_\nu 2)(\overline{x}) = x_1 2 + x_2 2$. We clearly have $x_1 + x_2 \leqslant (x_1 + x_2)2 \leqslant x_1 2 + x_2 2$ for all values of x_1 and x_2. Note that $(x_1 + x_2)2 \neq x_1 2 + x_2 2$ since $\cdot$ does not right-distribute over $+$, as shown after Definition 7.2.

(b) Consider the ROEs $f(\overline{x}) = \omega^{x_1 + x_2 + 1} \oplus x_3 + 1$ and $g(\overline{x}) = x_2 + \omega^{x_1} 2 \oplus x_3$. As $\omega^{x_1 + x_2 + 1} > \omega^{x_1} 2$ we have $(k(\overline{x}), k_\omega) = (x_1 + x_2 + 1, 1)$ and $(h(\overline{x}), h_\omega) = (x_1 + x_2 + 1, 2)$. Thus $(f \oplus_\mu g)(\overline{x}) = x_2 + \omega^{x_1 + x_2 + 1} \oplus x_3 2 + 1$ and $(f \oplus_\nu g)(\overline{x}) = \omega^{x_1 + x_2 + 1} 2 \oplus x_2 \oplus x_3 2 + 1$. It is not difficult to see that

$$x_2 + \omega^{x_1 + x_2 + 1} \oplus x_3 2 + 1 \leqslant f(\overline{x}) \oplus g(\overline{x}) \leqslant \omega^{x_1 + x_2 + 1} 2 \oplus x_2 \oplus x_3 2 + 1$$

for all values of x_1, x_2, and x_3.

(c) Consider the ROEs $f(\overline{x}) = x_3 + \omega^{x_2} \oplus x_1$ and $g(\overline{x}) = \omega^{x_1+x_2+1} + 1$. As $\omega^{x_2} \not\geqslant \omega^{x_1+x_2+1}$ we have $(f +_\mu g)(\overline{x}) = x_3 + g(\overline{x}) = x_3 + \omega^{x_1+x_2+1} + 1$. Since $x_1 + x_2 + 1 > x_2$ the first case for $+_\nu$ applies, where $u = 0$ as $f_3 s_3 t_3 = 1$. We thus have $(f +_\nu g)(\overline{x}) = \omega^{x_1+x_2+1} \oplus x_3 + 1$. Note that the term $\oplus x_1$ in $f(\overline{x})$ disappears as x_1 contributes to the exponent of $g(\overline{x})$. We have

$$x_3 + \omega^{x_1+x_2+1} + 1 \leqslant (x_3 + \omega^{x_2} \oplus x_1) + (\omega^{x_1+x_2+1} + 1) \leqslant \omega^{x_1+x_2+1} \oplus x_3 + 1$$

for all values of x_1, x_2, and x_3.

(d) For the ROEs $f(\overline{x}) = x_2 + \omega^{x_1+1}$, $g_1(\overline{x}) = \omega^{x_1} \oplus x_2$, and $g_2(\overline{x}) = \omega^{\omega^{x_1} \oplus x_2} \oplus x_3$ we obtain

$$f(\overline{g})_\mu(\overline{x}) = (\omega^{\omega^{x_1} \oplus x_2} \oplus x_3) +_\mu \omega^{\omega^{x_1} \oplus x_2 + 1} = \omega^{\omega^{x_1} \oplus x_2 + 1}$$
$$f(\overline{g})_\nu(\overline{x}) = (\omega^{\omega^{x_1} \oplus x_2} \oplus x_3) +_\nu \omega^{\omega^{x_1} \oplus x_2 + 1} = x_3 + \omega^{\omega^{x_1} \oplus x_2 + 1}$$

(e) Consider $\ell = \bullet\,\mathsf{f}(\mathsf{c}(x_1, x_2), x_3)$ and $r = \mathsf{h}(\bullet\,\mathsf{f}(x_1, x_2), \bullet\bullet\,\mathsf{f}(\mathsf{f}(x_1, x_2), x_3))$ from rule (C2) of $\mathcal{G}$. Let $\mathcal{O}$ be the ordinal part of the ROE algebra defined in the proof of Theorem 7.15 such that $\mathsf{h}_\mathcal{O}(x_1, x_2) = x_2 + \omega^{x_1+1}$, $\mathsf{c}_\mathcal{O}(x_1, x_2) = \omega^{x_1} \oplus x_2 + 1$, $\bullet_\mathcal{O}(x_1) = x_1$, and $\mathsf{f}_\mathcal{O}(x_1, x_2) = \omega^{x_1} \oplus x_2$. Then we have $\mu_\mathcal{O}(\ell) = \nu_\mathcal{O}(\ell) = \omega^{\omega^{x_1} \oplus x_2 + 1} \oplus x_3$. It is easy to see that for $r' = \mathsf{f}(\mathsf{f}(x_1, x_2), x_3)$ we get $\mu_\mathcal{O}(r') = \nu_\mathcal{O}(r') = \omega^{\omega^{x_1} \oplus x_2} \oplus x_3$. From the computation in (d) we thus obtain $\nu_\mathcal{O}(r) = x_3 + \omega^{\omega^{x_1} \oplus x_2 + 1}$. Note that $[\mu_\mathcal{O}(\ell) \geqslant \nu_\mathcal{O}(r)]$ holds: We obviously have $[\mu_\mathcal{O}(\ell) \geqslant_0 \nu_\mathcal{O}(r)]$, $[\mu_\mathcal{O}(\ell) \geqslant_1 \nu_\mathcal{O}(r)]$, and $[\mu_\mathcal{O}(\ell) \geqslant_2 \nu_\mathcal{O}(r)]$ as the two expressions are equal in the relevant parts, and $[\mu_\mathcal{O}(\ell) \geqslant_3 \nu_\mathcal{O}(r)]$.

Note that in [193, Definition 17] we approximated $(x + \omega^0 0) \oplus_\nu \omega^x$ by $x + \omega^x$ (but $[\alpha](x \oplus \omega^x) > [\alpha](x + \omega^x)$ for $\alpha(x) = 1$), and $(x \oplus y) +_\nu \omega$ by $x + y + \omega$ (whereas $[\alpha]((x \oplus y) + \omega) > [\alpha](x + y + \omega)$ for $\alpha(x) = \omega$ and $\alpha(y) = \omega^2$). Definition 7.22 corrects these flaws and sets $(x + \omega^0 0) \oplus_\nu \omega^x = \omega^x \oplus x$ and $(x \oplus y) +_\nu \omega = \omega \oplus x \oplus y$. We now show that Definition 7.22 yields valid over- and underapproximations.

Lemma 7.24. *Let $\mathcal{O}$ be an ROE algebra and t be a term. Then $[\alpha](\mu_\mathcal{O}(t)) \leqslant [\alpha]_\mathcal{O}(t) \leqslant [\alpha](\nu_\mathcal{O}(t))$ for all assignments α.*

Proof. We argue that all approximations in Definition 7.22 constitute valid lower and upper bounds. Let α be an arbitrary assignment.

(a) It is easy to see that $[\alpha](f(\overline{x}) \cdot a) \leqslant [\alpha](f \cdot_\nu a)(\overline{x})$. For any β in CNF as in (7.1) and $a \in \mathbb{N}_{>0}$ we have $\beta a = \omega^{\beta_1} a_1 a + \omega^{\beta_2} a_2 + \cdots + \omega^{\beta_n} a_n$ [119]. Since for any $1 \leqslant i \leqslant n$ we have $\omega^{\beta_1} a_1 a + \cdots + \omega^{\beta_n} a_n \geqslant \omega^{\beta_1} a_1 + \cdots + \omega^{\beta_i} a_i a + \cdots + \omega^{\beta_n} a_n$, $(f \cdot_\mu a)(\overline{x})$ constitutes a safe (though modest) lower bound for $f(\overline{x})a$.

(b) We have

$$f(\overline{x}) \oplus g(\overline{x}) = \left(\sum_{1 \leqslant i \leqslant n} x_i f_i + \omega^{f'(\overline{x})} f_\omega\right) \oplus \left(\sum_{1 \leqslant i \leqslant n} x_i g_i + \omega^{g'(\overline{x})} g_\omega\right) \qquad (7.5)$$
$$\oplus \bigoplus_{1 \leqslant i \leqslant n} x_i(\hat{f}_i + \hat{g}_i) \oplus (f_0 + g_0)$$

Note that the term $x_i f_i$ disappears in $f(\overline{x}) \oplus g(\overline{x})$ if x_i contributes to $\omega^{f'(\overline{x})}$ and $f_\omega > 0$, and the term $x_i g_i$ disappears in $f(\overline{x}) \oplus g(\overline{x})$ if x_i contributes to $\omega^{g'(\overline{x})}$ and $g_\omega > 0$. Hence we may multiply all occurrences of f_i by s_i, and occurrences of g_i by t_i. We then have $[\alpha](f \oplus_\mu g)(\overline{x}) \leqslant [\alpha](f(\overline{x}) \oplus g(\overline{x}))$ as $(f \oplus_\mu g)(\overline{x})$ underapproximates

$$\left(\sum_{1 \leqslant i \leqslant n} x_i f_i + \omega^{f'(\overline{x})} f_\omega\right) \oplus \left(\sum_{1 \leqslant i \leqslant n} x_i g_i + \omega^{g'(\overline{x})} g_\omega\right)$$

by a coefficient-wise maximum of the respective components in $f(\overline{x})$ and $g(\overline{x})$. Concerning the upper bound, the first two cases are obvious from (7.5). Otherwise, it is easy to see that $\omega^{f'(\overline{x})} f_\omega \oplus \omega^{g'(\overline{x})} g_\omega \leqslant \omega^{h(\overline{x})} h_\omega$. As the sum of $x_i f_i$ and $x_i g_i$ can be overapproximated by the natural sum of all terms $(f_i s_i + g_i t_i) x_i$ we have $[\alpha](f(\overline{x}) \oplus g(\overline{x})) \leqslant [\alpha](f \oplus_\nu g)(\overline{x})$.

(c) We clearly have $[\alpha](f +_\mu g)(\overline{x}) \leqslant [\alpha](f(\overline{x}) + g(\overline{x}))$. For the upper bound, assume for a first case $[\omega^{g'(\overline{x})} g_\omega > \omega^{f'(\overline{x})} f_\omega]$, so $\omega^{f'(\overline{x})} f_\omega + \omega^{g'(\overline{x})} g_\omega = \omega^{g'(\overline{x})} g_\omega$. Note that the term $x_i \hat{f}_i$ disappears in $f(\overline{x}) + g(\overline{x})$ if x_i is contained in $\omega^{g'(\overline{x})} g_\omega$, i.e., if x_i occurs with a positive coefficient somewhere in $g'(\overline{x})$ and $g_\omega > 0$. The term $g_i x_i$ disappears as well if x_i is contained in $\omega^{g'(\overline{x})} g_\omega$, and $f_i x_i$ disappears if x_i occurs in $\omega^{f'(\overline{x})} f_\omega$, or if x_i occurs in $\omega^{g'(\overline{x})} g_\omega$. Hence all occurrences of $\hat{f}_i$ and g_i may be multiplied by t_i, and occurrences of f_i may be multiplied by $s_i t_i$. Clearly all terms $x_i f_i s_i t_i$ and $x_i g_i t_i$ may be put in the standard addition part of $(f +_\nu g)(\overline{x})$, and $x_i \hat{g}_i$ occurs in the natural addition part. As far as the terms $x_i \hat{f}_i t_i$ are concerned, adding them to the natural addition part is obviously sound; but note that we may also put $x_i \hat{f}_i t_i$ into the standard addition part if $x_i \hat{f}_i$ is the only part of $f(\overline{x})$ that survives, which is captured by the condition $u = 1$. Now suppose $[\omega^{f'(\overline{x})} f_\omega > \omega^{g'(\overline{x})} g_\omega]$, so $\omega^{f'(\overline{x})} f_\omega + \omega^{g'(\overline{x})} g_\omega \leqslant \omega^{f'(\overline{x})}(f_\omega + 1)$. The term $\hat{f}_i x_i$ disappears in $f(\overline{x}) + g(\overline{x})$ if x_i is contained in $\omega^{g'(\overline{x})} g_\omega$, the term $g_i x_i$ disappears as well if x_i is contained in $\omega^{g'(\overline{x})} g_\omega$. Hence for any variable x_i the sum of $x_i \hat{f}_i$, $x_i g_i$, and $x_i \hat{g}_i$ can be overapproximated by $x_i(\hat{f}_i t_i + g_i t_i + \hat{g}_i)$ such that $[\alpha](f(\overline{x}) + g(\overline{x})) \leqslant [\alpha](f +_\nu g)(\overline{x})$. Finally, $f(\overline{x}) + g(\overline{x}) \leqslant f(\overline{x}) \oplus g(\overline{x}) \leqslant (f \oplus_\nu g)(\overline{x})$ holds in any case.

(d) By (a)–(c) and weak monotonicity of the ordinal operations $\cdot$, $+$, and $\oplus$.

(e) By induction on the term structure of t, using (d). $\square$

Main Theorem

Any ROE is weakly monotone and well-defined by definition. It is easy to encode a criterion for an ROE $f(\overline{x})$ to be simple:

$$\text{simple}(f(\overline{x})) = \bigwedge_{1 \leqslant i \leqslant n} \text{con}_i(f(\overline{x}))$$

Finally we obtain the main result of this section.

Theorem 7.25. *Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$ and $\mathcal{O}$ an ROE algebra on $\mathcal{F}$. If*

$$\bigwedge_{\ell \to r \in \mathcal{R}} [\mu_{\mathcal{O}}(\ell) > \nu_{\mathcal{O}}(r)] \wedge \bigwedge_{f \in \mathcal{F}} \text{simple}(f_{\mathcal{O}}(\overline{x}))$$

holds then $\mathcal{R}$ is terminating.

Proof. We already observed that any ROE is weakly monotone and well-defined. By the assumption, every $f_{\mathcal{O}}$ is simple. Hence the result follows by Theorem 7.6 in combination with Lemmata 7.20 and 7.24. $\square$

7.4.2. Implementation

In this section we discuss crucial issues for a successful implementation. Section 7.4.2 explains the search for suitable interpretations. Section 7.4.2 shows how to ensure that the lexicographic combination of partial proofs preserves weak monotonicity. Section 7.4.2 deals with the problem of a compatible variable order and Section 7.4.2 is dedicated to efficiency considerations.

Search for Interpretations

In automatic termination proofs suitable interpretation functions must be constructed. While easy heuristics can be employed for the depth of an ROE (see Section 7.4.2), the main challenge is to establish suitable coefficients. To this end we consider *parametric* ROEs which are of the shape (7.4) with the exception that now $f_0, f_1, \ldots, f_n, \hat{f}_1, \ldots, \hat{f}_n, f_\omega$ are *unknowns* over the naturals. The encodings from the previous section then allow to reduce the search for suitable coefficients to finding models in existentially quantified non-linear integer arithmetic for which suitable SMT solvers exist (see e.g. [208]).

Lexicographic Combination of Interpretations

The termination proof of the TRS $\mathcal{G}$ (Theorem 7.15) performs a lexicographic combination of algebras into a simple and weakly monotone algebra. The proof can be seen as the lexicographic product of (1) an ordinal algebra and (2) a linear (polynomial) interpretation and (3) a matrix interpretation of dimension 2 [44].[5] Regarding automation one can either encode the search for the lexicographic combination or search for (partial) proofs and combine them lexicographically. We adopted the latter, although the lexicographic combination of weakly monotone algebras need not be weakly monotone, as shown by the following example.

Example 7.26. Consider the nonterminating TRS $\mathcal{R} = \{\mathsf{f}(\mathsf{a}) \to \mathsf{f}(\mathsf{b}), \mathsf{b} \to \mathsf{a}\}$. For the weakly monotone simple interpretation $\mathsf{f}_{\mathcal{O}}(x) = x + \omega, \mathsf{b}_{\mathcal{O}} = 1, \mathsf{a}_{\mathcal{O}} = 0$ we have $[\mathsf{f}(\mathsf{a})]_{\mathcal{O}} = \omega \geqslant \omega = [\mathsf{f}(\mathsf{b})]_{\mathcal{O}}$ and $[\mathsf{b}]_{\mathcal{O}} = 1 > 0 = [\mathsf{a}]_{\mathcal{O}}$. If we removed the second rule, then the weakly monotone simple interpretation $\mathsf{f}_{\mathcal{N}}(x) = x + 1, \mathsf{a}_{\mathcal{N}} = 1, \mathsf{b}_{\mathcal{N}} = 0$ shows termination of the remaining rule $\mathsf{f}(\mathsf{a}) \to \mathsf{f}(\mathsf{b})$. Note that the lexicographic combination is no longer weakly monotone, i.e., $[\mathsf{b}]_{\mathcal{O}\times\mathcal{N}} = (1,0) >_{\mathsf{lex}} (0,1) = [\mathsf{a}]_{\mathcal{O}\times\mathcal{N}}$ but $[\mathsf{f}(\mathsf{b})]_{\mathcal{O}\times\mathcal{N}} = (\omega,1) \not\geqslant_{\mathsf{lex}} (\omega,2) = [\mathsf{f}(\mathsf{a})]_{\mathcal{O}\times\mathcal{N}}$.

However, weak monotonicity of a lexicographic interpretation can be partially recovered: If both $f_{\mathcal{O}}(\overline{x})$ and $f_{\mathcal{A}}(\overline{x})$ are weakly monotone this also holds for the lexicographic combination $(f_{\mathcal{O}}(\overline{x}), f_{\mathcal{A}}(\overline{x}))$ provided that an argument x_i is ignored in $f_{\mathcal{A}}(\overline{x})$ whenever $f_{\mathcal{O}}(\overline{x})$ is not strictly—but still weakly—monotone with respect to x_i. This fact was already exploited in the termination proof by Touzet (see Example 7.30) and is also used in the proof of Theorem 7.15.

Example 7.27. For the interpretations $\mathsf{g}_{\mathcal{O}}(x) = x + \omega$ and $\mathsf{g}_{\mathcal{N}}(x) = 1$ the lexicographic combination $\mathsf{g}_{\mathcal{O}\times\mathcal{N}}((x,k)) = (x+\omega, 1)$ is weakly monotone. Similarly, for $\mathsf{h}_{\mathcal{O}}(x,y) = y + x$ and $\mathsf{h}_{\mathcal{N}}(x,y) = x+1$ also $\mathsf{h}_{\mathcal{O}\times\mathcal{N}}((x,k),(y,m)) = (y+x, k+1)$ is weakly monotone (note that $h_{\mathcal{O}\times\mathcal{N}}$ is strictly monotone in its first argument).

To encode monotonicity of an ROE $f(\overline{x})$ in its i-th argument we set

$$\begin{aligned}\mathrm{mon}_i(f(\overline{x})) &= \big(f_i > 0 \land (\bigwedge_{i<j\leqslant n} f_j = 0) \land f_\omega = 0\big) \lor \hat{f}_i > 0 \lor (f_\omega > 0 \land \mathrm{mon}_i(f'(\overline{x})))\\ \mathrm{mon}(f(\overline{x})) &= \bigwedge_{1\leqslant i\leqslant n} \mathrm{mon}_i(f(\overline{x}))\end{aligned}$$

Then it follows that $\neg\mathrm{mon}(f(\overline{x}))$ ($\neg\mathrm{mon}_i(f(\overline{x}))$) holds whenever $f(\overline{x})$ is not strictly monotone (in its i-th argument). In the implementation we consider relative rewriting and add a rule $f'(\pi(x_1,\ldots,x_n)) \to f(x_1,\ldots,x_n)$ in the relative part

[5]Item (3) can also be seen as a lexicographic combination of two linear (polynomial) interpretations.

whenever $\mathrm{mon}(f(\overline{x}))$ is not satisfied. Here f' is a fresh function symbol and $\pi(x_1,\ldots,x_n)$ returns the variables $x_{i_1},\ldots,x_{i_m}$ for $1 \leqslant i_1 \leqslant \cdots \leqslant i_m \leqslant n$ in which $f(\overline{x})$ is strictly monotone, i.e., $\mathrm{mon}_{i_j}(f(\overline{x}))$ is satisfied. In presence of a rule $f'(\pi(x_1,\ldots,x_n)) \to f(x_1,\ldots,x_n)$, compatible interpretation functions $f_{\mathcal{A}}$ cannot depend on variables $x_j \notin \pi(x_1,\ldots,x_n)$. The idea is shown by the examples below.

Example 7.28 (Example 7.26 revisited)**.** Consider the TRS $\mathcal{R}$ from Example 7.26. After applying the first interpretation we are left with the following relative TRS $\{\mathsf{f}(\mathsf{a}) \to \mathsf{f}(\mathsf{b})\}/\{\mathsf{f}' \to \mathsf{f}(x)\}$. Although this system is terminating there is no compatible interpretation since f may not depend on its arguments due to the second rule.

Example 7.29 (Example 7.27 revisited)**.** Let $\mathsf{h}_{\mathcal{O}}(x_1,x_2) = x_2 + x_1$. Then we obtain $\mathrm{mon}_1(\mathsf{h}_{\mathcal{O}}(x_1,x_2)) = \top$ and $\mathrm{mon}_2(\mathsf{h}_{\mathcal{O}}(x_1,x_2)) = \bot$. Hence $\pi(x_1,x_2) = x_1$ and subsequent interpretations have to orient $\mathsf{h}'(x_1) \to \mathsf{h}(x_1,x_2)$ weakly and cannot depend on h's second argument while e.g., $\mathsf{h}_{\mathcal{A}}(x_1,x_2) = x_1 + 1$ is possible.

However, adding rules $f'(\pi(x_1,\ldots,x_n)) \to f(x_1,\ldots,x_n)$ is likely to disable the orientation of rules whose left-hand sides are rooted by f (in order to satisfy $[\alpha]_{\mathcal{A}}(f'(\pi(x_1,\ldots,x_n))) \geqslant [\alpha]_{\mathcal{A}}(f(x_1,\ldots,x_n))$ the interpretation of f may not depend on arguments x_i which do not occur in $\pi(x_1,\ldots,x_n)$) and consequently the termination proof might not be successful. To avoid this situation in the implementation we add constraints demanding to orient such rules only if the interpretation of f is not strictly monotone. Then rules rooted with f must be oriented before a rule $f'(\pi(x_1,\ldots,x_n)) \to f(x_1,\ldots,x_n)$ is added.

Another necessary requirement is that the (lexicographic) algebra is simple. Again we avoid an explicit lexicographic encoding. Rather, in a preprocessing step for every $f \in \mathcal{F}$ we add the embedding rules $f(x_1,\ldots,x_n) \to x_i$ (for $1 \leqslant i \leqslant n$) into the relative component of the TRS. This then ensures $[\alpha]_{\mathcal{A}}(f(x_1,\ldots,x_n)) \geqslant [\alpha]_{\mathcal{A}}(x_i)$ for each $1 \leqslant i \leqslant n$.

All in all, for a TRS $\mathcal{R}$ over a signature $\mathcal{F}$ we execute the following procedure:

```
$\mathcal{S} := \{f(x_1,\ldots,x_n) \to x_i \mid f \in \mathcal{F}$ has arity $n$ and $1 \leqslant i \leqslant n\}$
while $\mathcal{R} \neq \varnothing$ do
    find an algebra $\mathcal{A}$ satisfying $\mathcal{R} \cup \mathcal{S} \subseteq \geqslant_{\mathcal{A}}$ and $(\mathcal{R} \cup \mathcal{S}) \cap >_{\mathcal{A}} \neq \varnothing$
    $\mathsf{Nmon}_{\mathcal{A}}(\mathcal{R}) := \{f'(\pi(x_1,\ldots,x_n)) \to f(x_1,\ldots,x_n) \mid f \in \mathcal{F}$ has arity $n$,
                         $1 \leqslant i \leqslant n$, and $x_i \notin \pi(x_1,\ldots,x_n)$ if
                         $f_{\mathcal{A}}$ is not strictly monotone in $x_i\}$
    $\mathcal{R} := \mathcal{R} \setminus >_{\mathcal{A}}$ and $\mathcal{S} := (\mathcal{S} \setminus >_{\mathcal{A}}) \cup \mathsf{Nmon}_{\mathcal{A}}(\mathcal{R})$
report *terminating*
```

Instead of proving termination of $\mathcal{R}$ we try to establish termination of $\mathcal{R}$ relative to $\mathcal{S}$. This pre-processing step ensures that the algebras constructed in the body of the while loop are simple. We use SMT to find appropriate ROEs and matrix interpretations (of different dimensions), respectively. If no suitable algebra is found, the while loop is aborted and the procedure fails. Adding $\mathsf{Nmon}_{\mathcal{A}}(\mathcal{R})$ to the relative part ensures that the lexicographic combination of the employed algebras is weakly monotone.

Compatible Variable Orders

When interpreting or comparing terms we might get ROEs not having the same variable order. E.g., the rule $\mathsf{s}(\mathsf{g}(x, y)) \to \mathsf{g}(y, x)$ results in the constraint $x+y+1 > y + x$, if $\mathsf{g}_{\mathcal{O}}(x, y) = x + y$ and $\mathsf{s}_{\mathcal{O}}(x) = x + 1$. The assignment $\alpha(x) = 1$ and $\alpha(y) = \omega$ yields $1 + \omega + 1 = \omega + 1 \not> \omega + 1$ but according to Definition 7.18 the constraint $[x + y + 1 > y + x]$ is valid. The same effect also happens in arithmetic operations, e.g., the overapproximation of $+$ in Lemma 7.24(d). Taking $\mathsf{f}_{\mathcal{O}}(x, y) = \mathsf{g}_{\mathcal{O}}(x, y) = x+y$ with $\alpha(x) = 1$ and $\alpha(y) = \omega$, the term $\mathsf{f}(\mathsf{g}(x, y), \mathsf{g}(y, x))$ evaluates to $(1 + \omega) + (\omega + 1) = \omega 2 + 1$ but the overapproximation based on the variable order $[x, y]$ yields $2 + \omega 2 = \omega 2$. Clearly $\omega 2 + 1 \not\leqslant \omega 2$. Hence we have to ensure our global assumption that two ROEs have *compatible* variable orders (in the standard addition part) when comparing, composing, or adding them. Let $\sum_{1 \leqslant i \leqslant n} x_i f_i$ and $\sum_{1 \leqslant i \leqslant n} y_i g_i$ be ordinal expressions over the same variables (so $\overline{y}$ is a permutation of $\overline{x}$). Let $i < j$. Two variables x_i and x_j are not compatible if there exist i' and j' with $1 \leqslant i' < j' \leqslant n$ such that $x_i = y'_j$, $x_j = y'_i$ and $f_i, f_j, g_{i'}, g_{j'}$ are positive. In such a case we constrain one of the coefficients to be zero, i.e., $f_i = 0 \vee f_j = 0 \vee g_{i'} = 0 \vee g_{j'} = 0$. For example consider $e_1 = x_1 1 + x_2 1$, $e_2 = x_2 1 + x_1 1$, and $e_3 = x_2 1 + x_1 0$. Then e_1 and e_2 do not have compatible variable orders while e_1 and e_3 do.

Efficiency

While the implementation fixes some initial depth d for the interpretation of function symbols, this depth increases when evaluating terms (when approximating compositions $f(g_1(\overline{x}), \ldots, g_n(\overline{x}))$). Not surprisingly, for efficiency it is necessary to bound the depth of expressions occurring in evaluations of terms. Dropping parts of an interpretation is sound as an underapproximation while for the overapproximation we add constraints (to the SMT solver) that the dropped part must evaluate to zero.

7.4.3. Examples and Limitations

We have implemented the search for suitable ROEs in $\mathsf{T_TT_2}$ [107] (see Section 7.6 for the global setup). For the automatic termination proof of the TRS $\mathcal{G}$ in $\mathsf{T_TT_2}$ we (lexicographically) combine ordinal algebras with matrix interpretations [44]. Then, $\mathsf{T_TT_2}$ manages $\mathcal{G}$ within nine seconds when using depth 1 for interpreting function symbols and limiting the depth of evaluations to 2. The CNF of the underlying SAT problem has approximately 120,000 variables and 300,000 clauses.

In their influential paper Kirby and Paris [96] also presented the battle of Hercules and Hydra as a combinatorial game on trees. Generalizations of the Hydra battle are found in many papers ([48] contains a nice survey) and several different encodings of the battle into a termination problem of a specific TRS can be found in the literature [27, 40, 41, 43, 113, 178]. Not all of these TRSs faithfully model the battle, and termination of some of them is not independent of Peano arithmetic.

Example 7.30. Touzet [178] presents the following TRS $\mathcal{H}$ to describe the battle between Hercules and Hydra for starting terms corresponding to ordinals $\alpha < \omega^{\omega^\omega}$ and using the *standard strategy*:

$$
\begin{array}{rrr}
\circ\, x \to \bullet\, \Box\, x & \mathsf{H}(\mathsf{0}, x) \to \circ\, x & \bullet\, \mathsf{c}^1(x, y) \to \mathsf{c}^1(x, \mathsf{H}(x, y)) \\
\bullet\, \Box\, x \to \Box \bullet \bullet\, x & \bullet\, \mathsf{H}(\mathsf{H}(\mathsf{0}, y), z) \to \mathsf{c}^1(y, z) & \bullet\, \mathsf{c}^2(x, y, z) \to \mathsf{c}^2(x, \mathsf{H}(x, y), z) \\
\Box\, \circ\, x \to \circ\, \Box\, x & \bullet\, \mathsf{H}(\mathsf{H}(\mathsf{H}(\mathsf{0}, x), y), z) \to \mathsf{c}^2(x, y, z) & \mathsf{c}^1(y, z) \to \circ\, z \\
\bullet\, x \to x & \mathsf{c}^2(x, y, z) \to \circ\, \mathsf{H}(y, z) &
\end{array}
$$

So far all termination tools failed on this example whose derivational complexity cannot be bounded by a multiple recursive function. Its termination can be shown by the following simple and weakly monotone interpretation $\mathcal{A}$ over the domain $\mathbb{O} \times \mathbb{N} \times \mathbb{N}$, where $f(x, y) = y + \omega^{x+1}$ [178]:

$$
\begin{array}{rlrl}
\mathsf{0}_{\mathcal{A}} &= (0, 0, 0) & \Box_{\mathcal{A}}(x, m, n) &= (x, 2m + 2, n) \\
\mathsf{H}_{\mathcal{A}}((x, m, n), (y, k, l)) &= (\omega^x \oplus y, 0, 0) & \circ_{\mathcal{A}}(x, m, n) &= (x, 2m + 3, n) \\
\mathsf{c}^1_{\mathcal{A}}((x, m, n), (y, k, l)) &= (f(x, y), 0, 0) & \bullet_{\mathcal{A}}(x, m, n) &= (x, m, n + m + 1) \\
\mathsf{c}^2_{\mathcal{A}}((x, m, n), (y, k, l), (z, i, j)) &= (\omega^{f(x,y)} \oplus z, 0, 0) & &
\end{array}
$$

Compared to $\mathcal{G}$, $\mathsf{T_TT_2}$ requires more resources (initial depth 2, intermediate depth 3, 12 seconds, 160,000 variables, 410,000 clauses) to automatically prove termination of $\mathcal{H}$. This is surprising as the derivational complexity of $\mathcal{G}$ far exceeds that of the Hydra system $\mathcal{H}$, which is bounded by the Hardy function $H_{\omega^{\omega^\omega}}$.

Example 7.31. Beklemishev [21] presents two infinite TRSs and one finite TRS describing the Worm battle (corresponding to a one-dimensional version of Buchholz' Hydra battle [28], first introduced by Hamano and Okada [67]). The second infinite TRS $\mathcal{W}_2$ consists of the rules

$$(x \cdot y) \cdot z \to x \cdot (y \cdot z) \qquad \mathsf{f}(\mathsf{0}) \to \mathsf{0}^m \qquad \mathsf{f}(\mathsf{0} \cdot x) \to (\mathsf{0} \cdot \mathsf{f}(x))^m$$

for $m \geqslant 1$, where t^m abbreviates the term $t \cdot (\cdots (t \cdot (t \cdot t)) \cdots)$ with m copies of t. The ROE algebra $\mathsf{0}_{\mathcal{O}} = 1$, $f_{\mathcal{O}}(x) = \omega^x$, and $x \cdot_{\mathcal{O}} y = 2x \oplus y \oplus 1$ is weakly monotone and simple on $\mathbb{O}$ and orients $\mathcal{W}_2$:

$$\begin{aligned} 4x \oplus 2y \oplus z \oplus 3 &> 2x \oplus 2y \oplus z \oplus 2 \\ \omega &> 3m - 2 \\ \omega^{x \oplus 3} &> \omega^x(2m-1) \oplus (7m-4) \end{aligned}$$

The finite system $\mathcal{W}_3'$ to simulate the Worm sequence consists of the following rules:[6]

$$\begin{array}{rlrl}
1\colon & (x \cdot y) \cdot z \to x \cdot (y \cdot z) & 2\colon & \mathsf{f}(\mathsf{0} \cdot x) \to \mathsf{b}(\mathsf{0} \cdot \mathsf{f}(x)) \\
3\colon & \mathsf{f}(\mathsf{0}) \to \mathsf{b}(\mathsf{0}) & 4\colon & \mathsf{a}(\mathsf{f}(x)) \to \mathsf{f}(\mathsf{a}(x)) \\
5\colon & \mathsf{a}(x \cdot y) \to \mathsf{a}(x) \cdot y & 6\colon & \mathsf{a}(\mathsf{b}_1(x)) \to \mathsf{b}_1(\mathsf{a}(x)) \\
7\colon & \mathsf{f}(\mathsf{b}(x)) \to \mathsf{b}(\mathsf{f}(x)) & 8\colon & \mathsf{b}(x) \cdot y \to \mathsf{b}(x \cdot y) \\
9\colon & \mathsf{a}(\mathsf{f}(\mathsf{0} \cdot x)) \to \mathsf{b}_1(\mathsf{f}(\mathsf{0} \cdot x) \cdot (\mathsf{0} \cdot \mathsf{f}(x))) & 10\colon & \mathsf{a}(\mathsf{f}(\mathsf{0})) \to \mathsf{b}_1(\mathsf{f}(\mathsf{0}) \cdot \mathsf{0}) \\
11\colon & \mathsf{b}_1(\mathsf{b}(x)) \to \mathsf{b}(\mathsf{b}(x)) & 12\colon & \mathsf{c}(\mathsf{b}(x)) \to \mathsf{c}(\mathsf{a}(x)) \\
13\colon & \mathsf{a}(\mathsf{b}(x)) \to \mathsf{b}(\mathsf{a}(x)) & 14\colon & \mathsf{a}(\mathsf{0} \cdot x) \to \mathsf{b}(\mathsf{b}(x))
\end{array}$$

Consider the algebra $\mathcal{A}$ on $\mathbb{O}_{>0} \times \mathbb{N} \times \mathbb{N}$:

$$\begin{array}{rlrl}
\mathsf{c}_{\mathcal{A}}(x,m,n) &= (x+1, 2m+2, 2m) & \mathsf{0}_{\mathcal{A}} &= (1,0,0) \\
\mathsf{b}_{\mathcal{A}}(x,m,n) &= (x, m+1, n) & \mathsf{b_1}_{\mathcal{A}}(x,m,n) &= (x, 2m, n+1) \\
\mathsf{a}_{\mathcal{A}}(x,m,n) &= (x, m, m+2n) & \mathsf{f}_{\mathcal{A}}(x,m,n) &= (\omega^x, m, m+3) \\
(x,m,n) \cdot_{\mathcal{A}} (y,k,l) &= (y+x, m, m+n+1) & &
\end{array}$$

[6]Note that we added rule (14) to the TRS $\mathcal{W}_3$ originally presented in [21] since personal communication with Lev Beklemishev revealed that such an additional rule is in fact required to faithfully model the worm sequence. We believe the derivational complexity of $\mathcal{W}_3$ to be actually smaller than that of $\mathcal{W}_2$ and $\mathcal{W}_3'$, which is also supported by the fact that termination of $\mathcal{W}_3$ can be shown by $\mathsf{T_TT_2}$ with standard techniques.

This algebra is simple (note that $(x,m,n) \cdot_{\mathcal{A}} (y,k,l) \geqslant (y,k,l)$ since $x \neq 0$), weakly monotone, and orients all rules of $\mathcal{W}_3'$:

$$\begin{aligned}
1&: & (z+y+x,m,2m+n+2) &> (z+y+x,m,m+n+1)\\
2&: & (\omega^{x+1},0,3) &> (\omega^x+1,1,1)\\
3&: & (\omega,0,3) &> (1,1,0)\\
4&: & (\omega^x,m,3m+6) &> (\omega^x,m,m+3)\\
5&: & (y+x,m,3m+2n+2) &> (y+x,m,2m+2n+1)\\
6&: & (x,2m,2m+2n+2) &> (x,2m,m+2n+1)\\
7&: & (\omega^x,m+1,m+4) &> (\omega^x,m+1,m+3)\\
8&: & (y+x,m+1,m+n+2) &> (y+x,m+1,m+n+1)\\
9&: & (\omega^{x+1},0,6) &> (\omega^x+1+\omega^{x+1},0,5)\\
10&: & (\omega,0,6) &> (\omega,0,5)\\
11&: & (x,2m+2,n+1) &> (x,m+2,n)\\
12&: & (x+1,2m+4,2m+2) &> (x+1,2m+2,2m)\\
13&: & (x,m+1,m+2n+1) &> (x,m+1,m+2n)\\
14&: & (x+1,0,2) &> (x,m+2,n)
\end{aligned}$$

for all $x,y,z \in \mathbb{O}_{>0}$ and $m,n,k,l \in \mathbb{N}$. Thus this algebra shows termination of the system $\mathcal{W}_3'$ by Theorem 7.6.

The termination proof from the above example cannot be reproduced within TT2, since the interpretation function of $\cdot$ is simple on the carrier $\mathbb{O}_{>0} \times \mathbb{N}^2$ but not on $\mathbb{O} \times \mathbb{N}^2$, which would be used by TTT2. However, TTT2 succeeds in the dependency pair setting where it manages the crucial SCC by lexicographically combining an ROE algebra of degree 2 with a linear interpretation. The overall time is about four seconds while the CNF of the underlying SAT problem has approximately 87,000 variables and 205,000 clauses.

7.5. Automation of Elementary Algebras

Similar to ordinal algebras (Section 7.4), we give encodings of elementary interpretation functions (Section 7.5.1) before implementation aspects are addressed in Section 7.5.2 and examples (and limitations) are discussed in Section 7.5.3.

The shape of FBIs (see below) suffices to go beyond polynomial interpretations, which fail on Examples 7.33 and 7.34.[7] Furthermore, a fixed base allows to use more powerful approximations of comparisons/arithmetic.

[7]We remark that establishing termination of these systems becomes much easier when using depen-

Definition 7.32. A *fixed-base elementary interpretation function* (FBI) of depth 0 is

$$f(\overline{x}) = \sum_{1 \leqslant i \leqslant n} x_i f_i + f_0$$

and an FBI of depth $d+1$ is

$$f(\overline{x}) = \sum_{1 \leqslant i \leqslant n} x_i f_i + f_0 + b^{f'(\overline{x})} \left(\sum_{1 \leqslant i \leqslant n} x_i \hat{f}_i + \hat{f}_0 \right) \tag{7.6}$$

where $f_0, f_1, \ldots, f_n, \hat{f}_0, \hat{f}_1, \ldots, \hat{f}_n$ are naturals, $f'(\overline{x})$ is an FBI of depth d, and $b \geqslant 2$ is a fixed natural number. Throughout this section we use the following abbreviations:

$$\dot{f}(\overline{x}) = \sum_{1 \leqslant i \leqslant n} x_i f_i + f_0 \qquad \hat{f}(\overline{x}) = \sum_{1 \leqslant i \leqslant n} x_i \hat{f}_i + \hat{f}_0$$

An *FBI algebra* has $\mathbb{N}_{\geqslant 1}$ as carrier and FBIs as interpretation functions for all function symbols in the signature.

It is known that for polynomial interpretations the carriers $\mathbb{N}$ and $\mathbb{N}_{\geqslant \mu}$ admit the same termination proving power for any $\mu \in \mathbb{N}$ (see e.g. [36, 172]). However, the situation is different for FBI's. The function $2^x y$ is not monotone on $\mathbb{N}$ but it is on $\mathbb{N}_{\geqslant 1}$. The typical transformation converts $2^x y$ into the function $2^{x+1}(y+1) - 1$, but the latter does not admit an FBI representation. The following examples show the usefulness of $2^x y$, so we restrict to the carrier $\mathbb{N}_{\geqslant 1}$ in the sequel.

Example 7.33. Termination of Lescanne's factorial example [115]

$$\begin{array}{ccc} 0 + x \to x & 0 \cdot x \to 0 & \mathsf{fact}(0) \to \mathsf{s}(0) \\ \mathsf{s}(x) + y \to \mathsf{s}(x + y) & \mathsf{s}(x) \cdot y \to x \cdot y + y & \mathsf{fact}(\mathsf{s}(x)) \to \mathsf{s}(x) \cdot \mathsf{fact}(x) \\ x \cdot (y + z) \to x \cdot y + x \cdot z & & \end{array}$$

can be shown by an FBI algebra $\mathcal{A}$ of depth 2 with base $b = 2$ and interpretation functions $0_{\mathcal{A}} = 2$, $\mathsf{s}_{\mathcal{A}}(x) = x+2$, $x +_{\mathcal{A}} y = 2x+y+1$, $x \cdot_{\mathcal{A}} y = 2^x y$, and $\mathsf{fact}_{\mathcal{A}}(x) = 2^{2^x}$. We have

$$\begin{array}{ccc} x + 5 > x & 2^2 x > 2 & 2^{2^2} > 4 \\ 2x + y + 5 > 2x + y + 3 & 2^{x+2} y > 2^{x+1} y + y + 1 & 2^{2^{x+2}} > 2^{x+2+2^x} \\ 2^x(2y + z + 1) > 2^{x+1} y + 2^x z + 1 & & \end{array}$$

for all $x, y, z \geqslant 1$.

dency pairs but then totality of the order is lost, which is essential for applications such as ordered completion.

Example 7.34. Termination of Lucas' factorial example [117]

$$\begin{array}{lll} x + \mathsf{0} \to x & \mathsf{0} \cdot x \to \mathsf{0} & \mathsf{fact}(\mathsf{0}) \to \mathsf{s}(\mathsf{0}) \\ x + \mathsf{s}(y) \to \mathsf{s}(x+y) & \mathsf{s}(x) \cdot y \to x \cdot y + y & \mathsf{fact}(\mathsf{s}(x)) \to \mathsf{s}(x) \cdot \mathsf{fact}(x) \end{array}$$

can also be shown by an FBI algebra $\mathcal{A}$ of depth 2 with base $b = 2$ and interpretation functions $\mathsf{0}_\mathcal{A} = 2$, $\mathsf{s}_\mathcal{A}(x) = x + 2$, $x +_\mathcal{A} y = x + 2y + 1$, $x \cdot_\mathcal{A} y = 2^x y$, and $\mathsf{fact}_\mathcal{A}(x) = 2^{2^x}$. We have

$$\begin{array}{lll} x + 5 > x & 2^2 x > 2 & 2^{2^2} > 4 \\ x + 2y + 5 > x + 2y + 3 & 2^{x+2} y > 2^x y + 2y + 1 & 2^{2^{x+2}} > 2^{x+2} 2^{2^x} = 2^{x+2+2^x} \end{array}$$

for all $x, y \geqslant 1$.

In the sequel we sometimes treat an FBI $f(\overline{x})$ of depth 0 as $\sum_{1 \leqslant i \leqslant n} x_i f_i + f_0 + b^0 0$ to avoid case distinctions.

7.5.1. Encodings

Let $f(\overline{x})$ and $g(\overline{x})$ be FBIs of the form

$$\begin{aligned} f(\overline{x}) &= \sum_{1 \leqslant i \leqslant n} x_i f_i + f_0 + b^{f'(\overline{x})} \left(\sum_{1 \leqslant i \leqslant n} x_i \hat{f}_i + \hat{f}_0 \right) \\ g(\overline{x}) &= \sum_{1 \leqslant i \leqslant n} x_i g_i + g_0 + b^{g'(\overline{x})} \left(\sum_{1 \leqslant i \leqslant n} x_i \hat{g}_i + \hat{g}_0 \right) \end{aligned} \tag{7.7}$$

Useful Abbreviations

First we introduce lower and upper bounds for two FBIs:

$$\begin{aligned} \mathrm{fmin}(f,g)(\overline{x}) = {} & \sum_{1 \leqslant i \leqslant n} x_i \min(f_i, g_i) + \min(f_0, g_0) \\ & + b^{\mathrm{fmin}(f',g')(\overline{x})} \left(\sum_{1 \leqslant i \leqslant n} x_i \min(\hat{f}_i, \hat{g}_i) + \min(\hat{f}_0, \hat{g}_0) \right) \\ \mathrm{fmax}(f,g)(\overline{x}) = {} & \sum_{1 \leqslant i \leqslant n} x_i \max(f_i, g_i) + \max(f_0, g_0) \\ & + b^{\mathrm{fmax}(f',g')(\overline{x})} \left(\sum_{1 \leqslant i \leqslant n} x_i \max(\hat{f}_i, \hat{g}_i) + \max(\hat{f}_0, \hat{g}_0) \right) \end{aligned}$$

Again we introduce the notation of contribution of a variable x_i to $f(\overline{x})$, which we denote by $\mathrm{con}_i(f(\overline{x}))$:[8]

$$\mathrm{con}_i(f(\overline{x})) = f_i > 0 \vee \hat{f}_i > 0 \vee (\mathrm{con}_i(f'(\overline{x})) \wedge \hat{f}(\overline{x}) > 0)$$

Comparisons

The following recursive definition reduces the comparison of FBIs to the comparison of non-linear polynomials. The latter can be compared by the absolute positiveness approach, see [83]. For comparing polynomials we take the carrier $\mathbb{N}_{\geqslant 1}$ into account such that e.g. $3y \geqslant y + 1$ evaluates to true.

Definition 7.35. Let $f(\overline{x})$ and $g(\overline{x})$ be FBIs as in (7.7). Let

$$\lfloor b^{f'(\overline{x})} \rfloor = \big((\dot{f}'(\overline{x}) + \hat{f}'(\overline{x}) = 0)\ ?\ 1 : b\,(\dot{f}'(\overline{x}) + \hat{f}'(\overline{x}))\big)$$

Note that $b^{f(\overline{x})} \geqslant \lfloor b^{f(\overline{x})} \rfloor$. Furthermore let $p(\overline{x}) = \dot{f}'(\overline{x}) + \hat{f}'(\overline{x}) - \dot{g}'(\overline{x}) - \hat{g}'(\overline{x})$ and $h(\overline{x}) = \lfloor b^{p(\overline{x})} \rfloor \hat{f}(\overline{x}) - \hat{g}(\overline{x})$. We set

$$\begin{aligned}
[f(\overline{x}) \geqslant g(\overline{x})] \;=\; & (\hat{g}(\overline{x}) > 0 \to [f'(\overline{x}) \geqslant g'(\overline{x})]) \;\wedge\; \big(\\
& (\hat{f}(\overline{x}) > 0 \wedge [f'(\overline{x})\, b \geqslant g(\overline{x})]) \;\vee && \text{(a)} \\
& (\dot{f}(\overline{x}) \geqslant \dot{g}(\overline{x}) \wedge \hat{f}(\overline{x}) \geqslant \hat{g}(\overline{x})) \;\vee && \text{(b)} \\
& (h(\overline{x}) \geqslant 0 \wedge p(\overline{x}) \geqslant 0 \wedge \hat{f}'(\overline{x}) \geqslant \hat{g}'(\overline{x}) \wedge \\
& \qquad \dot{f}(\overline{x}) + \lfloor b^{g'(\overline{x})} \rfloor \lfloor b^{p(\overline{x})} \rfloor \hat{f}(\overline{x}) \geqslant \dot{g}(\overline{x}) + \lfloor b^{g'(\overline{x})} \rfloor \hat{g}(\overline{x}))\ \big) && \text{(c)} \\
[f(\overline{x}) > g(\overline{x})] \;=\; & (\hat{g}(\overline{x}) > 0 \to [f'(\overline{x}) \geqslant g'(\overline{x})]) \;\wedge\; \big(\\
& (\hat{f}(\overline{x}) > 0 \wedge [f'(\overline{x})\, b > g(\overline{x})]) \;\vee && \text{(d)} \\
& (\dot{f}(\overline{x}) \geqslant \dot{g}(\overline{x}) \wedge \hat{f}(\overline{x}) \geqslant \hat{g}(\overline{x}) \wedge \\
& \qquad ((\hat{f}(\overline{x}) > 0 \wedge [f'(\overline{x}) > g'(\overline{x})]) \vee \dot{f}(\overline{x}) > \dot{g}(\overline{x}) \vee \hat{f}(\overline{x}) > \hat{g}(\overline{x}))) \;\vee && \text{(e)} \\
& (h(\overline{x}) \geqslant 0 \wedge p(\overline{x}) \geqslant 0 \wedge \hat{f}'(\overline{x}) \geqslant \hat{g}'(\overline{x}) \wedge \\
& \qquad \dot{f}(\overline{x}) + \lfloor b^{g'(\overline{x})} \rfloor \lfloor b^{p(\overline{x})} \rfloor \hat{f}(\overline{x}) > \dot{g}(\overline{x}) + \lfloor b^{g'(\overline{x})} \rfloor \hat{g}(\overline{x}))\ \big) && \text{(f)}
\end{aligned}$$

The difference between $[f(\overline{x}) \geqslant g(\overline{x})]$ and $[f(\overline{x}) > g(\overline{x})]$ is that in the latter we demand at least one strict decrease. The following example shows that our encodings of comparisons are very accurate.

[8]Here $\hat{f}(\overline{x}) > 0$ tests the linear polynomial $\hat{f}(\overline{x})$ for positiveness.

Example 7.36. The encoding of $[2^{x+1} > 1 + 2^x]$ evaluates to true. The only interesting case is (f) where $p(\overline{x}) = 1$ and $\lfloor 2^x \rfloor \lfloor 2^1 \rfloor = 4x > 1 + 2x = 1 + \lfloor 2^x \rfloor$, i.e., $2x > 1$, which holds for all $x \in \mathbb{N}_{\geqslant 1}$.

The encodings of comparisons are sound.

Lemma 7.37. *Let $f(\overline{x})$ and $g(\overline{x})$ be FBIs as in (7.7).*

(a) *If $[f(\overline{x}) > g(\overline{x})]$ then $[\alpha](f(\overline{x})) > [\alpha](g(\overline{x}))$ for all assignments α.*

(b) *If $[f(\overline{x}) \geqslant g(\overline{x})]$ then $[\alpha](f(\overline{x})) \geqslant [\alpha](g(\overline{x}))$ for all assignments α.*

Proof. We only show (b) the argument for (a) is similar. Case (a) in Definition 7.35 approximates the situation when $b^{f'(\overline{x})}\hat{f}(\overline{x}) \geqslant g(\overline{x})$ and case (b) is obvious. Finally, case (c) follows from the argument below. Let $p(\overline{x})$ and $h(\overline{x})$ be as in Definition 7.35. For the moment assume $f'(\overline{x}) \geqslant p(\overline{x}) + g'(\overline{x})$. Then

$$\begin{aligned}
& \dot{f}(\overline{x}) + b^{f'(\overline{x})}\hat{f}(\overline{x}) \geqslant \dot{g}(\overline{x}) + b^{g'(\overline{x})}\hat{g}(\overline{x}) \\
\Longleftarrow\quad & \dot{f}(\overline{x}) + b^{p(\overline{x})+g'(\overline{x})}\hat{f}(\overline{x}) \geqslant \dot{g}(\overline{x}) + b^{g'(\overline{x})}\hat{g}(\overline{x}) \\
\Longleftrightarrow\quad & \frac{\dot{f}(\overline{x})}{b^{g'(\overline{x})}} + b^{p(\overline{x})}\hat{f}(\overline{x}) \geqslant \frac{\dot{g}(\overline{x})}{b^{g'(\overline{x})}} + \hat{g}(\overline{x}) \\
\Longleftrightarrow\quad & \frac{\dot{f}(\overline{x})}{b^{g'(\overline{x})}} + b^{p(\overline{x})}\hat{f}(\overline{x}) + h(\overline{x}) \geqslant \frac{\dot{g}(\overline{x})}{b^{g'(\overline{x})}} + \hat{g}(\overline{x}) + h(\overline{x}) \\
\Longleftarrow\quad & \frac{\dot{f}(\overline{x})}{b^{g'(\overline{x})}} + h(\overline{x}) \geqslant \frac{\dot{g}(\overline{x})}{b^{g'(\overline{x})}} \wedge b^{p(\overline{x})}\hat{f}(\overline{x}) \geqslant \hat{g}(\overline{x}) + h(\overline{x}) \\
\Longleftrightarrow\quad & \dot{f}(\overline{x}) + b^{g'(\overline{x})}h(\overline{x}) \geqslant \dot{g}(\overline{x}) \wedge b^{p(\overline{x})}\hat{f}(\overline{x}) \geqslant \hat{g}(\overline{x}) + h(\overline{x}) \\
\Longleftarrow\quad & h(\overline{x}) \geqslant 0 \wedge \dot{f}(\overline{x}) + \lfloor b^{g'(\overline{x})}\rfloor h(\overline{x}) \geqslant \dot{g}(\overline{x}) \wedge \lfloor b^{p(\overline{x})}\rfloor \hat{f}(\overline{x}) \geqslant \hat{g}(\overline{x}) + h(\overline{x}) & (\star) \\
\Longleftrightarrow\quad & h(\overline{x}) \geqslant 0 \wedge \dot{f}(\overline{x}) + \lfloor b^{g'(\overline{x})}\rfloor h(\overline{x}) \geqslant \dot{g}(\overline{x}) & (\dagger) \\
\Longleftrightarrow\quad & h(\overline{x}) \geqslant 0 \wedge \dot{f}(\overline{x}) + \lfloor b^{g'(\overline{x})}\rfloor \lfloor b^{p(\overline{x})}\rfloor \hat{f}(\overline{x}) \geqslant \dot{g}(\overline{x}) + \lfloor b^{g'(\overline{x})}\rfloor \hat{g}(\overline{x})
\end{aligned}$$

In the step $(\star)$ the non-negativity of $h(\overline{x})$ is used and the step $(\dagger)$ follows from the definition of $h(\overline{x})$. Finally we have to show that our assumption $f'(\overline{x}) \geqslant p(x)+g'(\overline{x})$ follows from the constraints. We observe

$$\begin{aligned}
& f'(\overline{x}) \geqslant p(\overline{x}) + g'(\overline{x}) \\
\Longleftrightarrow\quad & \dot{f}'(\overline{x}) + b^{f''(\overline{x})}\hat{f}'(\overline{x}) \geqslant \dot{f}'(\overline{x}) + \hat{f}'(\overline{x}) - \dot{g}'(\overline{x}) - \hat{g}'(\overline{x}) + \dot{g}'(\overline{x}) + b^{g''(\overline{x})}\hat{g}'(\overline{x}) \\
\Longleftrightarrow\quad & (b^{f''(\overline{x})} - 1)\hat{f}'(\overline{x}) \geqslant (b^{g''(\overline{x})} - 1)\hat{g}'(\overline{x}) \\
\Longleftarrow\quad & (\hat{g}'(\overline{x}) > 0 \to [f''(\overline{x}) \geqslant g''(\overline{x})]) \wedge \hat{f}'(\overline{x}) \geqslant \hat{g}'(\overline{x})
\end{aligned}$$

While the above proof does not rely on $p(\overline{x}) \geqslant 0$ this (redundant) constraint in Definition 7.35 might cut the search space. □

Composition

Similar as for ROEs, FBIs are not closed under addition and composition.

Example 7.38. The sum $2^x + 2^y$ of the FBIs 2^x and 2^y has no FBI representation. Also, substituting the FBI $2^y + 1$ for x in the FBI $2^x x$ results in $2^{2^y+1}(2^y + 1) = 2^{2^y+y+1} + 2^{2^y+1}$, which also has no equivalent FBI representation.

We thus define under- and overapproximations for addition, multiplication, and composition.

Definition 7.39. Let $f(\overline{x})$ and $g(\overline{x})$ be FBIs as in (7.7).

(a) Multiplication of an FBI by a scalar again yields an FBI, i.e.

$$f(\overline{x})\, a = \sum_{1 \leqslant i \leqslant n} x_i f_i a + f_0 a + b^{f'(\overline{x})} \Bigg(\sum_{1 \leqslant i \leqslant n} x_i \hat{f}_i a + \hat{f}_0 a \Bigg)$$

(b) For addition we use fmin (fmax) to estimate a lower (upper) bound for both $f(\overline{x})$ and $g(\overline{x})$, and introduce approximations by FBIs as follows:

$$\begin{aligned} f(\overline{x}) +_\mu g(\overline{x}) = {} & \sum_{1 \leqslant i \leqslant n} x_i (f_i + g_i) + (f_0 + g_0) \\ & + b^{e_\mu(\overline{x})} \Bigg(\sum_{1 \leqslant i \leqslant n} x_i (\hat{f}_i + \hat{g}_i) + (\hat{f}_0 + \hat{g}_0) \Bigg) \\ f(\overline{x}) +_\nu g(\overline{x}) = {} & \sum_{1 \leqslant i \leqslant n} x_i (f_i + g_i) + (f_0 + g_0) \\ & + b^{e_\nu(\overline{x})} \Bigg(\sum_{1 \leqslant i \leqslant n} x_i (\hat{f}_i + \hat{g}_i) + (\hat{f}_0 + \hat{g}_0) \Bigg) \end{aligned}$$

with $e_\mu(\overline{x})$ abbreviating $\hat{f}(\overline{x}) = 0 \ ? \ g'(\overline{x}) : \big(\hat{g}(\overline{x}) = 0 \ ? \ f'(\overline{x}) : \mathrm{fmin}(f', g')(\overline{x})\big)$ and $e_\nu(\overline{x})$ abbreviating $\hat{f}(\overline{x}) = 0 \ ? \ g'(\overline{x}) : \big(\hat{g}(\overline{x}) = 0 \ ? \ f'(\overline{x}) : \mathrm{fmax}(f', g')(\overline{x})\big)$.

(c) To approximate multiplication of an expression of the form $b^{g'(\overline{x})}$ with $f(\overline{x})$ by an FBI, we may use

$$\begin{aligned} b^{g'(\overline{x})} \cdot_\mu f(\overline{x}) &= \hat{f}(\overline{x}) > 0 \ ? \ \dot{f}(\overline{x}) + b^{f'(\overline{x}) +_\mu g'(\overline{x})} \hat{f}(\overline{x}) : b^{g'(\overline{x})} \dot{f}(\overline{x}) \\ b^{g'(\overline{x})} \cdot_\nu f(\overline{x}) &= \hat{f}(\overline{x}) > 0 \ ? \ b^{f'(\overline{x}) +_\nu g'(\overline{x})} \Bigg(\sum_{1 \leqslant i \leqslant n} x_i (\hat{f}_i + f_i) + (\hat{f}_0 + f_0) \Bigg) \\ & \qquad : b^{g'(\overline{x})} \dot{f}(\overline{x}) \end{aligned}$$

(d) Finally we can give approximations for the composition of two functions $f(\overline{g})(\overline{x}) = f(g_1(\overline{x}), \ldots, g_n(\overline{x}))$:

$$f(\overline{g})_\mu(\overline{x}) = \sum_{1 \leqslant i \leqslant n}^{\mu} g_i(\overline{x}) f_i +_\mu f_0 +_\mu b^{f'(\overline{g})_\mu(\overline{x})} \cdot_\mu \left(\sum_{1 \leqslant i \leqslant n}^{\mu} g_i(\overline{x}) \hat{f}_i +_\mu \hat{f}_0 \right)$$

$$f(\overline{g})_\nu(\overline{x}) = \sum_{1 \leqslant i \leqslant n}^{\nu} g_i(\overline{x}) f_i +_\nu f_0 +_\nu b^{f'(\overline{g})_\nu(\overline{x})} \cdot_\nu \left(\sum_{1 \leqslant i \leqslant n}^{\nu} g_i(\overline{x}) \hat{f}_i +_\nu \hat{f}_0 \right)$$

(e) Let t be a term and $\mathcal{A}$ an FBI algebra. By induction on the term structure we define FBIs $\mu_{\mathcal{A}}(t)$ and $\nu_{\mathcal{A}}(t)$ such that

$$\mu_{\mathcal{A}}(t) = \begin{cases} t & \text{if } t \in \mathcal{V} \\ f_{\mathcal{A}}(\mu_{\mathcal{A}}(t_1), \ldots, \mu_{\mathcal{A}}(t_n))_\mu & \text{otherwise} \end{cases}$$

$$\nu_{\mathcal{A}}(t) = \begin{cases} t & \text{if } t \in \mathcal{V} \\ f_{\mathcal{A}}(\nu_{\mathcal{A}}(t_1), \ldots, \nu_{\mathcal{A}}(t_n))_\nu & \text{otherwise} \end{cases}$$

The following example illustrates Definition 7.35.

Example 7.40. We consider the cases for addition and multiplication.

(b) We have $\operatorname{fmin}(x+1, x) = x$ and $\operatorname{fmax}(x+1, x) = x+1$, thus $2^{x+1}y +_\mu 2^x(z+1) = 2^x(y + z + 1)$ but $2^{x+1}y +_\nu 2^x(z + 1) = 2^{x+1}(y + z + 1)$.

In certain pathological cases the approximations of addition are not commutative. To be more precise, the resulting FBIs may be syntactically different but denote the same elementary function. For instance, $2^x \cdot 0 +_\mu 2^{x+1} \cdot 0 = 2^{x+1} \cdot 0$ while $2^{x+1} \cdot 0 +_\mu 2^x \cdot 0 = 2^x \cdot 0$. Still, we do not regard this a problem for our application as the encoding of comparisons takes these cases into account.

(c) For multiplication we have $2^{x+1} \cdot_\mu 2^{2^x} = 2^{(x+1)+_\mu 2^x} = 2^{x+1+2^x}$ and $2^{x+1} \cdot_\nu 2^{2^x} = 2^{x+1+2^x}$, the approximation is thus precise in these cases. On the other hand, as $(x+1) +_\mu 2^x = (x+1) +_\nu 2^x = x + 1 + 2^x$ we have $2^{x+1} \cdot_\mu (z + 1 + 2^{2^x}y) = z + 1 + 2^{x+1+2^x}y$, while $2^{x+1} \cdot_\nu (z + 1 + 2^{2^x}y) = 2^{x+1+2^x}(y + z + 1)$.

The following example shows that in practice our approximations are very accurate, i.e., for Examples 7.33 and 7.34 the approximations are exact.

Example 7.41. For Example 7.33 we get the following constraints

$$\begin{array}{ccc} x+5 > x & 2^2 x > 2 & 2^{2^2} > 4 \\ 2x+y+5 > 2x+y+3 & 2^{x+2}y > y+1+2^x 2y & 2^{2^{x+2}} > 2^{x+2+2^x} \\ 2^x(2y+z+1) > 1+2^x(2y+z) & & \end{array}$$

while Example 7.34 yields

$$\begin{array}{ccc} x+5 > x & 2^2 x > 2 & 2^{2^2} > 4 \\ x+2y+5 > x+2y+3 & 2^{x+2}y > 2y+1+2^x y & 2^{2^{x+2}} > 2^{x+2+2^x} \end{array}$$

We now show that Definition 7.39 yields valid over- and underapproximations.

Lemma 7.42. *Let $\mathcal{A}$ be an FBI algebra and t be a term. Then $[\alpha](\mu_{\mathcal{A}}(t)) \leqslant [\alpha]_{\mathcal{A}}(t) \leqslant [\alpha](\nu_{\mathcal{A}}(t))$ for all assignments α.*

Proof. We argue that all approximations in Definition 7.39 constitute valid lower and upper bounds. Let α be an arbitrary assignment.

(a) Since scalar multiplication is no approximation there is nothing to show.

(b) For $+_\mu$ (the reasoning for $+_\nu$ is analogous) one of the three cases applies:

$$f(\overline{x})+g(\overline{x}) \geqslant \dot f(\overline{x})+\dot g(\overline{x})+\begin{cases} b^{g'(\overline{x})}\hat g(\overline{x}) & \text{if } \hat f(\overline{x}) = 0 \\ b^{f'(\overline{x})}\hat f(\overline{x}) & \text{if } \hat g(\overline{x}) = 0 \\ b^{h'(\overline{x})}(\hat f(\overline{x})+\hat g(\overline{x})) & \text{if } [\alpha](f'(\overline{x})) \geqslant [\alpha](h'(\overline{x})) \\ & \text{and } [\alpha](g'(\overline{x})) \geqslant [\alpha](h'(\overline{x})) \end{cases}$$

(c) If $\hat f(\overline{x}) = 0$ then $b^{g'(\overline{x})} f(\overline{x}) = b^{g'(\overline{x})}\dot f(\overline{x})$ and if $\hat f(\overline{x}) > 0$ we obtain for $\cdot_\mu$

$$\begin{aligned} b^{g'(\overline{x})} \cdot f(\overline{x}) &= b^{g'(\overline{x})}\dot f(\overline{x}) + b^{f'(\overline{x})+g'(\overline{x})}\hat f(\overline{x}) \geqslant \dot f(\overline{x}) + b^{f'(\overline{x})+g'(\overline{x})}\hat f(\overline{x}) \\ &\geqslant \dot f(\overline{x}) + b^{f'(\overline{x})+_\mu g'(\overline{x})}\hat f(\overline{x}) = b^{g'(\overline{x})} \cdot_\mu f(\overline{x}) \end{aligned}$$

while $\cdot_\nu$ is justified by

$$\begin{aligned} b^{g'(\overline{x})} \cdot f(\overline{x}) &= b^{g'(\overline{x})}\dot f(\overline{x}) + b^{f'(\overline{x})+g'(\overline{x})}\hat f(\overline{x}) \leqslant b^{f'(\overline{x})+g'(\overline{x})}\dot f(\overline{x}) + b^{f'(\overline{x})+g'(\overline{x})}\hat f(\overline{x}) \\ &= b^{f'(\overline{x})+g'(\overline{x})}(\dot f(\overline{x}) + \hat f(\overline{x})) \leqslant b^{f'(\overline{x})+_\nu g'(\overline{x})}(\dot f(\overline{x}) + \hat f(\overline{x})) \\ &= b^{g'(\overline{x})} \cdot_\nu f(\overline{x}) \end{aligned}$$

(d) By (a)–(c) and weak monotonicity of addition, multiplication, and exponentiation.

(e) By induction on the term structure of t, using (d). □

Main Theorem

An FBI $f(\overline{x})$ is monotone if all variables x_i contribute to it. Monotonicity of $f(\overline{x})$ is thus expressed by

$$\operatorname{mon}(f(\overline{x})) = \bigwedge_{1 \leqslant i \leqslant n} \operatorname{con}_i(f(\overline{x}))$$

An FBI $f(\overline{x})$ is well-defined if $[f(\overline{x}) \geqslant 1]$ holds.

Finally we obtain the main result of this section.

Theorem 7.43. *Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$ and $\mathcal{A}$ be an FBI algebra on $\mathcal{F}$. If*

$$\bigwedge_{\ell \to r \in \mathcal{R}} [\mu_{\mathcal{A}}(\ell) > \nu_{\mathcal{A}}(r)] \wedge \bigwedge_{f \in \mathcal{F}} \big([f_{\mathcal{A}}(\overline{x}) \geqslant 1] \wedge \operatorname{mon}(f_{\mathcal{A}}(\overline{x}))\big)$$

holds then $\mathcal{R}$ is terminating.

Proof. By the assumption any $f_{\mathcal{A}}$ is well-defined and monotone. Hence the result follows by Theorem 7.5 in combination with Lemmata 7.37 and 7.42. □

7.5.2. Implementation

To find suitable coefficients we consider *parametric* FBIs where we let the coefficients $f_0, f_1, \ldots, f_n, \hat{f}_0, \hat{f}_1, \ldots, \hat{f}_n$ in (7.7) be *unknowns* over the naturals. Then the encodings from the previous section reduce the problem to finding models in existentially quantified non-linear integer arithmetic. For an efficient implementation the following heuristics (which are applied to interpretations of a function symbol but not enforced for FBIs occurring when evaluating terms) have been proved useful:

(d) depth: A function symbol f is interpreted by an FBI using $\max\{0, d_{\mathcal{R}}(f) - 2\}$ as depth where $d_{\varnothing}(f) = 0$ and otherwise we compute the depth by $d_{\mathcal{S}}(f) = 1 + \max\{d_{\mathcal{S} \setminus \mathcal{S}_f}(g) \mid \ell \to r \in \mathcal{S}_f \text{ and } g \text{ occurs in } r\}$. Here $\mathcal{S}_f$ denotes the rules in $\mathcal{S}$ whose left-hand sides have root f. For Examples 7.33 and 7.34 the heuristic yields depth 2 for fact, depth 1 for $\cdot$, and depth 0 for the remaining function symbols.

(1) shape: Every variable may only appear once in each FBI, i.e., either in $\dot{f}(\overline{x})$ or in $\hat{f}(\overline{x})$ or in $f'(\overline{x})$. We enforce this by adding a side constraint.

(2) shape: Note that in the motivating examples every function symbol is interpreted by an FBI $f(\overline{x})$ satisfying

$$\bigwedge_{1 \leqslant i \leqslant n} (f_i = 0 \vee \hat{f}_i = 0) \tag{7.8}$$

Heuristic (2) shares the variables for the coefficients f_i and $\hat{f}_i$. This is achieved by using fresh boolean variables b_i and interpreting a function by (here $f_i = c_i b_i$ and $\hat{f}_i = c_i(1 - b_i)$)

$$\sum_{1 \leqslant i \leqslant n} x_i b_i c_i + c_0 b_0 + b^{f'(\overline{x})} \left(\sum_{1 \leqslant i \leqslant n} x_i(1 - b_i)c_i + c_0(1 - b_0) \right)$$

Heuristic (2) does not work recursively but takes the constant part into account (in contrast to heuristic (1)).

(3) shape: At most one of $\hat{f}_i$ $(0 \leqslant i \leqslant n)$ is greater than zero.

(4) shape: If $\hat{f}(\overline{x}) = 0$ we demand all coefficients in $\dot{f}'(\overline{x})$ and $\hat{f}'(\overline{x})$ to be zero.

(5) shape: If $\hat{f}(\overline{x}) = 0$ we demand all coefficients in $f'(\overline{x})$ to be zero.

Note that (5) is more restrictive than (4); while the latter admits an interpretation of the form $b^{b^{2 \cdot 0}} \cdot 0$, this is not allowed when applying (5).

7.5.3. Examples and Limitations

It is not hard to construct TRSs where FBI termination proofs require interpretations of arbitrary depth.

Example 7.44. Let $\mathcal{R}_n$ for $n > 0$ consist of the rules

$$\begin{array}{ccc} x + \mathsf{0} \to x & x + \mathsf{s}(y) \to \mathsf{s}(x + y) & \mathsf{exp}_0(x) \to x \\ \mathsf{exp}_{i+1}(\mathsf{0}) \to \mathsf{exp}_i(\mathsf{s}(\mathsf{0})) & \mathsf{exp}_{i+1}(\mathsf{s}(x)) \to \mathsf{exp}_i(\mathsf{exp}_1(x) + \mathsf{exp}_1(x)) & \end{array}$$

for all $0 \leqslant i < n$. Termination of $\mathcal{R}_n$ can be shown by the FBI algebra $\mathcal{A}$ with base $b = 2$ and interpretations $\mathsf{0}_\mathcal{A} = 1$, $\mathsf{s}_\mathcal{A}(x) = x + 1$, $x +_\mathcal{A} y = x + 2y$, and $\mathsf{exp}_{i,\mathcal{A}}(x) = \exp_2^i(2x + 1)$ where $\exp_2^i(x)$ denotes i-fold exponentiation with base 2, i.e., $\exp_2^0(x) = x$ and $\exp_2^{i+1}(x) = 2^{\exp_2^i(x)}$:

$$\begin{array}{ccc} x + 2 > x & x + 2y + 2 > x + 2y + 1 & 2x + 1 > x \\ \exp_2^{i+1}(3) > \exp_2^i(5) & \exp_2^{i+1}(2x + 3) > \exp_2^i(2^{2x+1} + 2 \cdot 2^{2x+1}) & \end{array}$$

The last two inequalities can be verified by simple inductive arguments. It is easy to see that any FBI algebra that orients $\mathcal{R}_n$ needs to have at least depth n.

It can be shown that already $\mathcal{R}_1$ admits multiple exponential complexity. As to be expected, actually any TRS compatible with an FBI algebra is bounded by a multiple exponential function. A more precise upper bound is given by the following lemma.

Lemma 7.45. *For any TRS $\mathcal{R}$ compatible with an FBI algebra $\mathcal{A}$ having base b and maximal depth $d-1$, $\mathrm{dh}_{\mathcal{R}}(n) \in \exp_b^{d\,n}(\mathcal{O}(n))$.*

Proof. As $\mathrm{dh}_{\mathcal{R}}(t) \leqslant [t]_{\mathcal{A}}$, it suffices to find a $k \in \mathbb{N}$ such that any ground term t satisfies $[t]_{\mathcal{A}} \leqslant \exp_b^{d|t|}(k\,d\,|t|)$. Let $m-1$ be the maximal arity in $\mathcal{F}$, c the maximum of 2 and all coefficients occurring in $f_{\mathcal{A}}$ for $f \in \mathcal{F}$, and $k = 1 + \log_b(c\,m)$. We apply induction on t.

Suppose t is a constant a. In order to show $a_{\mathcal{A}} \leqslant \exp_b^d(k\,d)$ we consider a slightly more general statement. Let α be an FBI of depth e with base b and maximum coefficient smaller than or equal to c, such that α depends on no variables. We verify $\alpha \leqslant \exp_b^{e+1}(k\,(e+1))$ by induction on e, such that in particular $a_{\mathcal{A}} \leqslant \exp_b^d(k\,d)$. If $e = 0$ then $\alpha \leqslant c \leqslant c\,m = \exp_b^1(\log_b(c\,m)) \leqslant \exp_b^1(k)$. Otherwise, α has depth $e+1$ and thus α can be written as $\alpha = b^{\alpha'}c_1 + c_2$ where $c_1, c_2 \in \mathbb{N}$ and α' has depth e. By the induction hypothesis, $\alpha' \leqslant \exp_b^{e+1}(k\,(e+1))$, and hence

$$
\begin{aligned}
\alpha &\leqslant b^{\exp_b^{e+1}(k\,(e+1))}c + c = (b^{\exp_b^{e+1}(k\,(e+1))} + 1)c \leqslant b^{\exp_b^{e+1}(k\,(e+1))+1}b^{\log_b(c)} \\
&\leqslant b^{\exp_b^{e+1}(k\,(e+1))+k} \leqslant b^{\exp_b^{e+1}(k\,(e+2))} = \exp_b^{e+2}(k(e+2))
\end{aligned}
$$

Suppose $t = g(t_1, \ldots, t_n)$ is not a constant. Let $\alpha_i = [t_i]_{\mathcal{A}}$. Since $|t_i| \leqslant |t| - 1$, the induction hypothesis yields $\alpha_i \leqslant \exp_b^{d\,(|t|-1)}(k\,d\,(|t|-1))$. To verify $[t]_{\mathcal{A}} \leqslant \exp_b^{d\,|t|}(k\,d\,|t|)$ we consider a more general statement. Let $f(\overline{x})$ be an FBI of the shape (7.6), having base b, maximum coefficient at most c and depth e. We abbreviate $f(\alpha_1, \ldots, \alpha_n)$ by α and show $\alpha \leqslant \exp_b^{d\,(|t|-1)+e+1}(k\,d\,(|t|-1) + k\,(e+1))$ by induction on e.

If $e = 0$ then $f(\overline{x})$ is just a linear function and thus

$$
\begin{aligned}
\alpha &\leqslant \exp_b^{d\,(|t|-1)}(k\,d\,(|t|-1))\,c\,m \leqslant b^{\exp_b^{d\,(|t|-1)}(k\,d\,(|t|-1))+\log_b(c\,m)} \\
&\leqslant b^{\exp_b^{d\,(|t|-1)}(k\,d\,(|t|-1)+k)} = \exp_b^{d\,(|t|-1)+1}(k\,d\,(|t|-1) + k)
\end{aligned}
$$

Now suppose $f(\overline{x})$ has depth $e+1$. Thus $f'(\overline{x})$ has depth e and, by the induction hypothesis, $f'(\alpha_1,\ldots,\alpha_n) \leqslant \exp_b^{d\,(|t|-1)+e+1}(k\,d\,(|t|-1)+k\,(e+1)) = \beta$. Therefore

$$\begin{aligned}
\alpha &= \sum_{1\leqslant i\leqslant n} \alpha_i f_i + f_0 + b^{f'(\overline{\alpha})}\Biggl(\sum_{1\leqslant i\leqslant n} \alpha_i \hat{f}_i + \hat{f}_0\Biggr)\\
&\leqslant b^{\beta}\exp_b^{d\,(|t|-1)}(k\,d\,(|t|-1))\,c\,m + \exp_b^{d\,(|t|-1)}(k\,d\,(|t|-1))\,c\,m\\
&= (b^{\beta}+1)\exp_b^{d\,(|t|-1)}(k\,d\,(|t|-1))\,c\,m \leqslant b^{\beta+1}\exp_b^{d\,(|t|-1)}(k\,d\,(|t|-1))\,c\,m\\
&\leqslant b^{\beta+1} b^{\exp_b^{d\,(|t|-1)}(k\,d\,(|t|-1))} b^{\log_b(c\,m)} = b^{\beta+\exp_b^{d\,(|t|-1)}(k\,d\,(|t|-1))+k}\\
&= b^{\exp_b^{d\,(|t|-1)+e+1}(k\,d\,(|t|-1)+k\,(e+1))+\exp_b^{d\,(|t|-1)}(k\,d\,(|t|-1))+k}\\
&\leqslant b^{\exp_b^{d\,(|t|-1)+e+1}(k\,d\,(|t|-1)+k\,(e+2))} = \exp_b^{d\,(|t|-1)+e+2}(k\,d\,(|t|-1)+k\,(e+2))
\end{aligned}$$

In particular, $g_{\mathcal{A}}(\overline{\alpha}) \leqslant \exp_b^{d\,(|t|-1)+d}(k\,d\,(|t|-1)+k\,d) = \exp_b^{d\,|t|}(k\,d\,|t|)$ as the depth of $g_{\mathcal{A}}$ is smaller than d. $\square$

The next example shows that the lack of multiplication is a weakness of FBIs.

Example 7.46. The following TRS (from [115, Fig. 2]) cannot be oriented by FBIs:

$$\begin{array}{ccc}
\mathsf{0} + x \to x & \mathsf{s}(x) + y \to \mathsf{s}(x+y) & x\cdot(y+z) \to x\cdot y + x\cdot z\\
\mathsf{0}\cdot x \to \mathsf{0} & \mathsf{s}(x)\cdot y \to x\cdot y + y & x\uparrow(y+z) \to (x\uparrow y)\cdot(x\uparrow z)\\
x\uparrow \mathsf{0} \to \mathsf{s}(\mathsf{0}) & x\uparrow \mathsf{s}(y) \to x\cdot(x\uparrow y) & (x\cdot y)\uparrow z \to (x\uparrow z)\cdot(y\uparrow z)\\
 & & (x\uparrow y)\uparrow z \to x\uparrow(y\cdot z)
\end{array}$$

This is because for any linear function $x +_{\mathcal{A}} y$ the interpretation $x \cdot_{\mathcal{A}} y$ has to involve exponentiation (as in Examples 7.33 and 7.34). As a consequence the rule $x\uparrow(y+z) \to (x\uparrow y)\cdot(x\uparrow z)$ is no longer orientable since the maximal power of b occurring in the (approximated) interpretation of the right-hand side exceeds the maximal power for the left-hand side. In contrast, elementary interpretations with non-fixed base succeed (cf. [115, Fig. 2]).

7.6. Experimental Results

We implemented the algebras from Sections 7.4 and 7.5 in the termination tool TT$_2$ [107]. In version 1.15, which is available from the tool's website,[9] ordinal algebras can be used by executing `./ttt2 -s HYDRA <file>` and FBI algebras by `./ttt2 -s FBI <file>`, respectively. Furthermore, the web interface has been updated accordingly.

[9]`http://cl-informatik.uibk.ac.at/software/ttt2/`

method	YES	avg. time	$\mathcal{G}$ (Def. 7.12)	$\mathcal{H}$ (Ex. 7.30)	$\mathcal{W}_3'$ (Ex. 7.31)
ROE algebras	329	2.1	8.2	11.4	4.0

Table 7.1.: Experimental results for ROE algebras.

For experiments[10] we considered the 1463 TRSs in the Standard TRS category of the Termination Problems Data Base (TPDB 8.0.7)[11] and the examples from the paper. The experiments have been performed using a single node of a machine equipped with 12 quad-core AMD Opteron™ processors 6174 running at a clock rate of 2.2 GHz and 330 GB of main memory. If a TRS could not be handled within 60 seconds, the execution of $\mathsf{T_TT_2}$ was aborted.

For the evaluation in Table 7.1 the following setup is used. ROE algebras (according to the description in Section 7.4.2) with interpretation functions of initial depth two are used in combination with weakly monotone matrix interpretations of dimension two. The coefficients are represented with up to four bits. The method is applied directly (establishing simple termination, if successful) and in the DP setting in combination with dependency graphs, SCC analysis and the subterm criterion. The left part of Table 7.1 shows the performance of this ROE algebra-based strategy on TPDB while the relevant examples from the paper are considered in the right part of the table where the numbers indicate the execution time in seconds.

Table 7.2 compares the power of FBIs (of depth at most 2) with linear polynomial interpretations when used in direct termination proofs (orient all rules by a single interpretation). For numbers in parentheses $\mathsf{T_TT_2}$ was not successful. The numbers in brackets indicate which heuristics have been used. FBIs as well as linear interpretations use two bits to encode coefficients and six bits for arithmetic evaluations.

Our experiments show the need for a heuristic concerning the depth of the FBIs. The other heuristics are much less important, i.e., they either slightly decrease the execution time or increase the number of systems shown terminating. We remark that any proper subset of the heuristics $\{1, 2, 3, 4, 5\}$ has only tiny effects on the execution speed of the examples in the right part of Table 7.2 while the whole set admits significant gains. The systems where FBIs succeed but linear polynomials fail often require interpretation functions of non-linear shape.

While FBIs are successful on the examples from the right part of Table 7.2, $\mathsf{T_TT_2}$ cannot establish termination using ROE algebras. On the other hand, FBIs cannot cope with the examples from the right part of Table 7.1 due to their derivational complexity.

[10]Details available from `http://cl-informatik.uibk.ac.at/ttt2/ordinals`
[11]Available from `http://termcomp.uibk.ac.at`.

method	YES	avg. time	Ex. 7.33	Ex. 7.34	[115, Fig. 1]
poly	125	0.3	(0.2)	(0.4)	(0.3)
fbi	41	29.7	*1443.4*	*731.0*	*13540.5*
fbi[d]	170	4.7	16.1	10.0	27.8
fbi[d12345]	174	4.2	8.9	7.9	24.1

Table 7.2.: Experimental results for FBI algebras.

7.7. Conclusion

We have encoded Goodstein's sequence as a TRS and discussed automation of a termination criterion which can cope with this system. Furthermore our implementation is also successful on an encoding of the battle of Hercules and Hydra, for which a (sound) automatic termination proof has been lacking so far. While preliminary experiments on the termination problems database TPDB did not yield proofs for previously unknown problems, we regard the main attraction of our method that it allows to go beyond multiple recursive derivation length. As shown in the article, automation of lexicographic combinations of termination proofs with respect to Theorem 7.6 is more challenging than with respect to Theorem 7.5.

Needless to say, there will always be TRSs whose termination is out of reach of automatic tools. Lepper [113] presented an infinite sequence $(\mathcal{R}_k)_{k \geqslant 1}$ of TRSs that simulate Hydra battles. Each of these TRSs is simply terminating, but the derivational complexity of $\mathcal{R}_k$ cannot be bounded by an α-recursive function such that $\alpha < \Delta_k$, where Δ_k approaches the small Veblen ordinal $\vartheta(\Omega^\omega)$ when k tends to infinity. With ROE algebras one can only prove termination of TRSs whose derivational complexity is ϵ_0-recursive.

The very first encoding of the Hydra battle in [41] still defeats $\mathsf{T_TT_2}$. A (difficult) termination proof of this TRS can be found in [123].

$\mathsf{T_TT_2}$ also fails on the Hydra encoding of Buchholz [27], which is not simply terminating although it admits a comparatively concise termination argument.

Furthermore, we have also shown how elementary interpretations can be automated using similar means, a challenge formulated as Problem #28 in the RTA List of Open Problems. Somehow surprisingly, FBIs require further heuristics to admit an efficient implementation. We believe that ordinal arithmetic is *easier* for the underlying SMT solver since expressions might be consumed while this is not the case for elementary arithmetic.

Concerning future work we mention that the approximation of term interpretations could partially be made more precise. As an example, we discuss scalar multiplication for ordinals. Since the approximations must be correct for all values of $\overline{x}$, the overapproximation $(f \cdot_\nu a)(\overline{x})$ is already optimal. To see this consider $(x + y) \cdot_\nu 2$ for natural values of x and y. Inspecting the proof of Lemma 7.24(a), instead of the current underapproximation $(f \cdot_\mu a)(\overline{x})$ we could also use (when $a > 0$)

$$(f \cdot_{\mu'} a)(\overline{x}) = \sum_{1 \leqslant i \leqslant n} x_i(f_i \cdot e_i) + \omega^{f'(\overline{x})}(f_\omega \cdot a) \oplus \bigoplus_{1 \leqslant i \leqslant n} x_i(\hat{f}_i \cdot a) \oplus (f_0 \cdot a)$$

where exactly one of e_i is a and all others are one. The underlying SMT solver can then choose an appropriate summand to be multiplied with a such that subsequent operations (addition, comparison, etc.) benefit. Refining the approximations for other operations (addition/comparison) is more involved and it is unclear if the additional precision prevails the increasing difficulty of the resulting SMT problems. Moreover, currently we do not know of any other TRSs with high derivational complexity that are within reach of our technique and could benefit from such improvements.

It is non-trivial to decide whether, given two ROEs $f(\overline{x})$ and $g(\overline{x})$ with given coefficients, $[\alpha](f(\overline{x})) > [\alpha](g(\overline{x}))$ holds for all assignments α. Though this problem is undecidable for polynomials, note that in the case of ROEs only linear constraints are involved. Further investigation of this issue might also lead to a better approximation of the encoding $[f(\overline{x}) > g(\overline{x})]$.

Generalizing elementary interpretations to a non-fixed base is an obvious choice for future work. However, we anticipate that suitable approximations will neither give further deep insights nor significantly improve termination proving power and hence we propose a different line of research. Since they are elementary interpretations, FBIs yield a totalizable order on ground terms. This holds despite the fact that our implementation relies on approximations when evaluating or comparing terms (see Example 7.36). Hence our implementation cannot be used to decide ordered rewriting, for instance to decide word problems using ground-convergent systems. However, since unfailing completion procedures never rely on the fact that $s >_{\mathcal{A}} t$ does *not* hold, FBIs can be used for ordered completion as in [191].

It is easy to enforce AC compatibility of algebras based on elementary and ordinal interpretation functions. For the case of FBI algebras, any AC symbol f must be interpreted by an FBI of the shape

$$(x_1 + x_2)f_1 + f_0 + b^{f'(x_1,x_2)}\left((x_1 + x_2)\hat{f}_1 + \hat{f}_0\right)$$

where $f'(x_1, x_2)$ is AC compatible as well. For instance, if $+$ is considered an AC symbol then AC termination of all TRSs in Examples 7.33, 7.34, and 7.44 can be shown with AC compatible FBI algebras (by picking $x +_{\mathcal{A}} y = 2x + 2y + c$ for a suitable constant c, and adapting the interpretations of other symbols accordingly). An ROE algebra is AC compatible if any AC symbol f is interpreted by an ROE

$$\omega^{f'(x_1,x_2)} f_\omega \oplus x_1 \hat{f}_1 \oplus x_2 \hat{f}_1 \oplus f_0 \tag{7.9}$$

where $f'(x_1, x_2)$ is again AC compatible. However, note that if a TRS $\mathcal{R}$ can be oriented with an ROE algebra where all interpretations match the shape (7.9) then a similar argument as used in [194, Theorem 13] shows that $\mathcal{R}$ is also compatible with an FBI algebra with sufficiently large base b.

Since non-linear polynomials give rise to an exponential size SMT encoding, such interpretations are hardly used within termination tools. We anticipate that suitable approximations could improve the performance of these implementations.

Formalizing our approximations in a theorem prover would extend the contributions from [119] and enable certification of such termination proofs.

Acknowledgments

We thank Lev Beklemishev, Georg Moser, and René Thiemann for helpful discussion and the reviewers for their detailed and constructive comments.

8. Uncurrying for Termination and Complexity

Publication Details

Nao Hirokawa, Aart Middeldorp, and Harald Zankl, Uncurrying for Termination and Complexity, Journal of Automated Reasoning 43(2), 279–315, 2013.
doi: 10.1007/s10817-012-9248-3

Harald Zankl, Nao Hirokawa, and Aart Middeldorp, Uncurrying for Innermost Termination and Derivational Complexity, In Proceedings of the 5th International Conference on Higher-Order Rewriting (HOR 2010), Electronic Proceedings in Theoretical Computer Science 49, pp. 46–57, 2011.
doi: 10.4204/EPTCS.49.4

Nao Hirokawa, Aart Middeldorp, and Harald Zankl, Uncurrying for Termination, In Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-15), Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence) 5330, pp. 667–681, Springer, 2008.
doi: 10.1007/978-3-540-89439-1_46

Abstract

First-order applicative rewrite systems provide a natural framework for modeling higher-order aspects. In this article we present a transformation from untyped applicative term rewrite systems to functional term rewrite systems that preserves and reflects termination. Our transformation is less restrictive than other approaches. In particular, head variables in right-hand sides of rewrite rules can be handled. To further increase the applicability of our transformation, we study the method for innermost rewriting and derivational complexity, and present a version for dependency pairs.

8.1. Introduction

In this article we are concerned with proving (innermost) termination of first-order applicative term rewrite systems. These systems provide a natural framework for modeling higher-order aspects found in functional programming languages. A prominent example of a first-order applicative system is combinatory logic. The signature of an applicative term rewrite system consists of constants and a single binary function symbol called application which is denoted by the infix and left-associative symbol $\star$. In term rewriting, properties such as termination and innermost termination are of particular interest since they are essential for many rewriting techniques including equational reasoning and confluence analysis (cf. [172]). Moreover, innermost termination has received a renewed interest in termination analysis of functional programs [57].

Proving termination of applicative term rewrite systems is challenging because the rewrite rules lack sufficient structure. As a consequence, simplification orders are not effective as $\star$ is the only function symbol of non-zero arity. Moreover, the dependency pair method is of little help as $\star$ is the only defined non-constant symbol. To remedy this issue two solutions have been suggested. The first line of research is based on types [8, 9, 10, 23, 89, 111, 182] and allows to study properties like termination or strong computability directly. The second approach [59, 74] aims for transformations that recover the structure of applicative rewrite rules to enable methods that do not rely on types. The benefit of the first approach is that the type information may make proving termination properties easier. However, most of those studies are within the realm of simply typed systems and hence miss polymorphism, which is used in many functional programming languages. Moreover, in contrast to untyped rewriting, no powerful automated tools are (yet) available. This is in sharp contrast to the untyped first-order setting where powerful tools exist as witnessed by the annual competition of termination tools.[1]

The main contribution of this article is a new transformation that recovers the structure in applicative rewrite rules, thereby enabling traditional methods for proving termination and innermost termination. Our transformation can deal with partial applications as well as head variables in right-hand sides of rewrite rules. The key ingredients are the η-saturation of rewrite rules (Definition 8.15) and the addition of sufficiently many uncurrying rules to the transformed system. These rules are also crucial for a smooth transition into the dependency pair framework. Unlike the transformation of applicative dependency pair problems presented in [59, 174], our uncurrying processor preserves minimality (cf. Section 8.6), which means that it can be used at any node in a modular (non-)termination proof.

[1] `http://termcomp.uibk.ac.at`

We remark that our results for proving innermost termination with the help of uncurrying are directly applicable for functional programming languages that adopt an eager evaluation strategy. Surprisingly, these results are also helpful in the case of lazy evaluation. Since applicative term rewrite systems modeling functional programs are left-linear and non-overlapping, termination and innermost termination coincide (see [65] for a more general result). Hence instead of establishing full termination for lazy languages we investigate innermost termination, which is equivalent in this case but typically easier to establish. In this context it is worth noting the recent work of Giesl *et al.* In [57] they present a two-stage transformation method from (functions in) Haskell programs to applicative dependency pair problems such that termination of the former is concluded from innermost finiteness of the latter. This approach is capable of automatically proving the termination of most functions in standard Haskell libraries.

The remainder of this article is organized as follows. After recalling preliminaries in Section 8.2, we present a new uncurrying transformation and prove that it preserves and reflects termination for full rewriting in Section 8.3. Results for innermost termination and derivational complexity are studied in Sections 8.4 and 8.5, respectively. Two extensions to the dependency pair framework are presented in Section 8.6. How these extensions behave for full termination is the topic of Section 8.6.1 while Section 8.6.2 is concerned with their properties concerning innermost termination. Our results are empirically evaluated in Section 8.7 and we conclude with a discussion of related work in Section 8.8.

A preliminary version of this article appeared in [74]. Several of the results on innermost rewriting and derivational complexity have been published in [202]. Theorems 8.33 and 8.52 are new contributions. So is the material in Section 8.6.2. Moreover, we close a non-trivial gap in the proof of [74, Theorem 33]. Some of the new contributions go beyond the scope of uncurrying, e.g., Theorem 8.52 gives a condition when the length of reductions is the same for innermost and full rewriting, which has an immediate impact on (automated) complexity analysis. Another fundamental new result deals with signature extensions, which do not affect termination [120, 136] but surprisingly may destroy finiteness of DP problems [165]. Finally, Lemma 8.70 shows that innermost finiteness is not affected by signature extensions.

8.2. Preliminaries

In this section we fix preliminaries on rewriting, complexity, dependency pairs, and currying.

8.2.1. Term Rewriting

We assume familiarity with term rewriting [17] in general and termination [210] in particular. Let $\mathcal{F}$ be a signature and $\mathcal{V}$ be a set of variables disjoint from $\mathcal{F}$. By $\mathcal{T}(\mathcal{F},\mathcal{V})$ we denote the set of terms over $\mathcal{F}$ and $\mathcal{V}$. The *size* of a term t is denoted $|t|$ and the root symbol of t is denoted $\mathrm{root}(t)$. A *rewrite rule* is a pair of terms (ℓ, r), written $\ell \to r$ such that ℓ is not a variable and all variables in r are contained in ℓ. A *term rewrite system* (TRS for short) is a set of rewrite rules. A TRS $\mathcal{R}$ is said to be *duplicating* if there exist a rewrite rule $\ell \to r \in \mathcal{R}$ and a variable x that occurs more often in r than in ℓ. By $\mathcal{F}\mathsf{un}(\mathcal{R})$ we denote the set of function symbols that occur in a TRS $\mathcal{R}$.

Contexts are terms over the signature $\mathcal{F} \cup \{\Box\}$ with exactly one occurrence of the fresh constant $\Box$ (called *hole*). The expression $C[t]$ denotes the result of replacing the hole in C by the term t. A *substitution* σ is a mapping from variables to terms and $t\sigma$ denotes the result of replacing the variables in t according to σ. Substitutions may change only finitely many variables (and are thus written as $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$). The *set of positions* of a term t is defined as $\mathcal{P}\mathsf{os}(t) = \{\epsilon\}$ if t is a variable and as $\mathcal{P}\mathsf{os}(t) = \{\epsilon\} \cup \{iq \mid q \in \mathcal{P}\mathsf{os}(t_i)\}$ if $t = f(t_1, \ldots, t_n)$. Positions are used to address occurrences of subterms. The *subterm of t at position* $p \in \mathcal{P}\mathsf{os}(t)$ is defined as $t|_p = t$ if $p = \epsilon$ and as $t|_p = t_i|_q$ if $p = iq$. We say a position p is *to the right of* a position q if $p = p_1ip_2$ and $q = q_1jq_2$ with $p_1 = q_1$ and $i > j$. For a term t and positions $p, q \in \mathcal{P}\mathsf{os}(t)$ we say $t|_p$ is to the right of $t|_q$ if p is to the right of q.

A *rewrite relation* is a binary relation on terms that is closed under contexts and substitutions. For a TRS $\mathcal{R}$ we define $\to_{\mathcal{R}}$ to be the smallest rewrite relation that contains $\mathcal{R}$. We call $s \to_{\mathcal{R}} t$ a *rewrite step* if there exist a context C, a rewrite rule $\ell \to r \in \mathcal{R}$, and a substitution σ such that $s = C[\ell\sigma]$ and $t = C[r\sigma]$. In this case we call $\ell\sigma$ a *redex* and say that $\ell\sigma$ has been *contracted.* A *root rewrite step*, denoted by $s \to_{\mathcal{R}}^{\epsilon} t$, has the shape $s = \ell\sigma \to_{\mathcal{R}} r\sigma = t$ for some $\ell \to r \in \mathcal{R}$. A *rewrite sequence* is a sequence of rewrite steps. The *set of normal forms* of a TRS $\mathcal{R}$ is defined as $\mathit{NF}(\mathcal{R}) = \{t \in \mathcal{T}(\mathcal{F},\mathcal{V}) \mid t \text{ contains no redexes}\}$. A redex $\ell\sigma$ in a term t is called *innermost* if proper subterms of $\ell\sigma$ are normal forms, and *rightmost innermost* if in addition $\ell\sigma$ is to the right of any other innermost redex in t. A rewrite step is called *innermost* (*rightmost innermost*) if an innermost (rightmost innermost) redex is contracted, written $\xrightarrow{\mathsf{i}}$ and $\xrightarrow{\mathsf{ri}}$, respectively.

If the TRS $\mathcal{R}$ is not essential or clear from the context the subscript $_{\mathcal{R}}$ is omitted in $\to_{\mathcal{R}}$ and its derivatives. As usual, $\to^+$ ($\to^*$) denotes the transitive (reflexive and transitive) closure of $\to$ and $\to^m$ its m-th iterate. A TRS is *terminating* (*innermost terminating*) if $\to^+$ ($\xrightarrow{\mathsf{i}}^+$) is well-founded. If $s \to_{\mathcal{R}}^* t$ and $t \in \mathit{NF}(\mathcal{R})$ then we write $s \to_{\mathcal{R}}^! t$.

An overlap $(\ell_1 \to r_1, p, \ell_2 \to r_2)_\mu$ of a TRS $\mathcal{R}$ consists of variants $\ell_1 \to r_1$ and $\ell_2 \to r_2$ of rules of $\mathcal{R}$ without common variables, a non-variable position $p \in \mathcal{P}\mathsf{os}(\ell_2)$, and a most general unifier μ of ℓ_1 and $\ell_2|_p$. If $p = \epsilon$ then we require that $\ell_1 \to r_1$ and $\ell_2 \to r_2$ are not variants of the same rewrite rule. A TRS without overlaps is called *non-overlapping*. An *overlay system* is a TRS whose overlaps emerge at root positions only.

Let $\mathcal{P}$ be a property of TRSs and let Φ be a transformation on TRSs with $\Phi(\mathcal{R}) = \mathcal{R}'$. We say Φ *preserves* $\mathcal{P}$ if $\mathcal{P}(\mathcal{R})$ implies $\mathcal{P}(\mathcal{R}')$ and Φ *reflects* $\mathcal{P}$ if $\mathcal{P}(\mathcal{R}')$ implies $\mathcal{P}(\mathcal{R})$. Sometimes we call Φ $\mathcal{P}$ preserving if Φ preserves $\mathcal{P}$ and $\mathcal{P}$ reflecting if Φ reflects $\mathcal{P}$, respectively.

8.2.2. Derivational Complexity

For complexity analysis we assume TRSs to be finite and (innermost) terminating. Hofbauer and Lautemann [77] introduced the concept of derivational complexity for terminating TRSs. The idea is to measure the maximal length of rewrite sequences (derivations) depending on the size of the starting term. Formally, the *derivation height* of a term t (with respect to a finitely branching and well-founded relation $\to$) is defined on natural numbers as $\mathrm{dh}(t, \to) = \max\{m \in \mathbb{N} \mid t \to^m u \text{ for some } u\}$. The *derivational complexity* $\mathrm{dc}_\mathcal{R}(n)$ of a TRS $\mathcal{R}$ is then defined as

$$\mathrm{dc}_\mathcal{R}(n) = \max\{\mathrm{dh}(t, \to_\mathcal{R}) \mid |t| \leqslant n\}$$

Similarly we define the *innermost* derivational complexity as

$$\mathrm{idc}_\mathcal{R}(n) = \max\{\mathrm{dh}(t, \xrightarrow{\mathsf{i}}_\mathcal{R}) \mid |t| \leqslant n\}$$

Since we regard finite TRSs only, these functions are well-defined if $\mathcal{R}$ is (innermost) terminating. If $\mathrm{dc}_\mathcal{R}(n)$ is bounded from above by a linear, quadratic, cubic, ... function or polynomial, $\mathcal{R}$ is said to have linear, quadratic, cubic, ... or polynomial derivational complexity. A similar convention applies to $\mathrm{idc}_\mathcal{R}(n)$.

For functions $f, g \colon \mathbb{N} \to \mathbb{N}$ we write $f(n) \in \mathcal{O}(g(n))$ if there are constants $c, N \in \mathbb{N}$ such that $f(n) \leqslant c \cdot g(n)$ for all $n \geqslant N$.

One popular method to prove polynomial upper bounds on the derivational complexity is via triangular matrix interpretations [126], which are a special instance of monotone algebras. For a signature $\mathcal{F}$, an *$\mathcal{F}$-algebra* $\mathcal{A}$ consists of a non-empty carrier A and a set of interpretations $f_\mathcal{A}$ for every $f \in \mathcal{F}$. By $[\alpha]_\mathcal{A}(\cdot)$ we denote the usual evaluation function of $\mathcal{A}$ according to an assignment α which maps variables to values in A. An $\mathcal{F}$-algebra $\mathcal{A}$ together with a well-founded order $\succ$ on A is called a *monotone algebra* if every $f_\mathcal{A}$ is monotone with respect to $\succ$. Any monotone algebra $(\mathcal{A}, \succ)$ induces a well-founded order on terms: $s \succ_\mathcal{A} t$ if for any assignment α the condition $[\alpha]_{\mathcal{A}}(s) \succ [\alpha]_{\mathcal{A}}(t)$ holds. A TRS $\mathcal{R}$ is *compatible* with a monotone algebra $(\mathcal{A}, \succ_{\mathcal{A}})$ if $\ell \succ_{\mathcal{A}} r$ for every $\ell \to r \in \mathcal{R}$.

Matrix interpretations $(\mathcal{M}, \succ)$ (often just denoted $\mathcal{M}$) are a special form of monotone algebras. Here the carrier is $\mathbb{N}^d$ for some fixed dimension $d \in \mathbb{N} \setminus \{0\}$. The order $\succ$ is defined on $\mathbb{N}^d$ as $(u_1, \ldots, u_d)^{\mathrm{T}} \succ (v_1, \ldots, v_d)^{\mathrm{T}}$ if $u_1 >_{\mathbb{N}} v_1$ and $u_i \geqslant_{\mathbb{N}} v_i$ for all $2 \leqslant i \leqslant d$. If every $f \in \mathcal{F}$ of arity n is interpreted as

$$f_{\mathcal{M}}(\overrightarrow{x_1}, \ldots, \overrightarrow{x_n}) = F_1\overrightarrow{x_1} + \cdots + F_n\overrightarrow{x_n} + \overrightarrow{f}$$

where $F_i \in \mathbb{N}^{d \times d}$ for all $1 \leqslant i \leqslant n$ and $\overrightarrow{f} \in \mathbb{N}^d$ then monotonicity of $\succ$ is achieved by demanding that the top left entry of every matrix F_i is non-zero. Such interpretations have been introduced in [44].

A square matrix A of dimension d is of *upper triangular* shape if $A_{(i,i)} \leqslant 1$ and $A_{(i,j)} = 0$ if $i > j$ for all $1 \leqslant i, j \leqslant d$. A matrix interpretation where for every $f \in \mathcal{F}$ all F_i ($1 \leqslant i \leqslant n$ where n is the arity of f) are upper triangular is called *triangular* (abbreviated by TMI). The next theorem is from [126].

Theorem 8.1. *If a TRS $\mathcal{R}$ is compatible with a TMI of dimension d then* $\mathrm{dc}_{\mathcal{R}}(n) \in \mathcal{O}(n^d)$. □

Recent generalizations of this result are reported in [122, 131, 187].

8.2.3. Dependency Pairs

Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$. The signature $\mathcal{F}$ is extended with *dependency pair symbols* $f^\sharp$ for every symbol $f \in \{\mathrm{root}(\ell) \mid \ell \to r \in \mathcal{R}\}$, where $f^\sharp$ has the same arity as f. If $\ell \to r \in \mathcal{R}$ and t is a subterm of r with a defined root symbol that is not a proper subterm of ℓ then the rule $\ell^\sharp \to t^\sharp$ is a *dependency pair* of $\mathcal{R}$. Here $\ell^\sharp$ and $t^\sharp$ are the result of replacing the root symbols in ℓ and t by the corresponding dependency pair symbols. The set of dependency pairs of $\mathcal{R}$ is denoted by $\mathsf{DP}(\mathcal{R})$. A *DP problem* is a pair of TRSs $(\mathcal{P}, \mathcal{R})$ such that the root symbols of the rules in $\mathcal{P}$ do neither occur in $\mathcal{R}$ nor in proper subterms of the left- and right-hand sides of rules in $\mathcal{P}$. The problem is said to be *finite* if there is no infinite sequence

$$s_1 \to_{\mathcal{P}}^{\epsilon} t_1 \to_{\mathcal{R}}^{*} s_2 \to_{\mathcal{P}}^{\epsilon} t_2 \to_{\mathcal{R}}^{*} \cdots$$

such that all terms $t_1, t_2, \ldots$ are terminating with respect to $\mathcal{R}$. Such an infinite sequence is said to be *minimal*. The main result underlying the dependency pair approach states that termination of a TRS $\mathcal{R}$ is equivalent to finiteness of the DP problem $(\mathsf{DP}(\mathcal{R}), \mathcal{R})$.

In order to prove a DP problem finite, a number of *DP processors* have been developed. DP processors are functions that take a DP problem as input and return a set of DP problems as output. In order to be employed to prove termination they need to be *sound*, that is, if all DP problems in a set returned by a DP processor are finite then the initial DP problem is finite. In addition, to ensure that a DP processor can be used to prove non-termination it must be *complete* which means that if one of the DP problems returned by the DP processor is not finite then the original DP problem is not finite.

A DP problem $(\mathcal{P}, \mathcal{R})$ is called *innermost finite* if there is no infinite sequence

$$s_1 \to_{\mathcal{P}}^{\epsilon} t_1 \xrightarrow{\mathsf{i}}{}^{!}_{\mathcal{R}} s_2 \to_{\mathcal{P}}^{\epsilon} t_2 \xrightarrow{\mathsf{i}}{}^{!}_{\mathcal{R}} \cdots$$

such that the term s_1 is in normal form with respect to $\mathcal{R}$. Soundness and completeness of DP problems for innermost termination are based on this altered notion of finiteness.

8.2.4. Currying

Definition 8.2. An *applicative* signature is a signature that consists of constants and a single binary function symbol called application, denoted by the infix and left-associative symbol $\star$. In examples we often use juxtaposition instead of $\star$. Based on an applicative signature we define *applicative terms (substitutions, contexts, TRSs)*. An applicative TRS is abbreviated by ATRS.

Every ordinary TRS can be transformed into an ATRS by currying.

Definition 8.3. Let $\mathcal{F}$ be a signature. The currying system $\mathcal{C}(\mathcal{F})$ consists of the rewrite rules $f_{i+1}(x_1, \ldots, x_i, y) \to f_i(x_1, \ldots, x_i) \star y$ for every n-ary function symbol $f \in \mathcal{F}$ and every $0 \leqslant i < n$. Here $f_n = f$ and, for every $0 \leqslant i < n$, f_i is a fresh function symbol of arity i.

The currying system $\mathcal{C}(\mathcal{F})$ is confluent and terminating. Hence every term t has a unique normal form $t{\downarrow_{\mathcal{C}(\mathcal{F})}}$. For instance, $\mathsf{f}(\mathsf{a}, \mathsf{b})$ is transformed into $\mathsf{f}_0\ \mathsf{a}_0\ \mathsf{b}_0$.

Definition 8.4. Let $\mathcal{R}$ be a TRS over the signature $\mathcal{F}$. The *curried* system $\mathcal{R}{\downarrow_{\mathcal{C}(\mathcal{F})}}$ is the ATRS consisting of the rules $\ell{\downarrow_{\mathcal{C}(\mathcal{F})}} \to r{\downarrow_{\mathcal{C}(\mathcal{F})}}$ for every $\ell \to r \in \mathcal{R}$. The signature of $\mathcal{R}{\downarrow_{\mathcal{C}(\mathcal{F})}}$ contains the application symbol $\star$ and a constant f_0 for every function symbol $f \in \mathcal{F}$.

In the following we write $\mathcal{R}{\downarrow_{\mathcal{C}}}$ for $\mathcal{R}{\downarrow_{\mathcal{C}(\mathcal{F})}}$ whenever $\mathcal{F}$ can be inferred from the context or is irrelevant. Moreover, we write f for f_0.

Example 8.5. The TRS $\mathcal{R}$

$$\begin{aligned} \mathsf{0} + y &\to y & \mathsf{0} \times y &\to \mathsf{0} \\ \mathsf{s}(x) + y &\to \mathsf{s}(x + y) & \mathsf{s}(x) \times y &\to (x \times y) + y \end{aligned}$$

is transformed into the ATRS $\mathcal{R}{\downarrow_\mathcal{C}}$

$$\begin{aligned} + \ \mathsf{0}\ y &\to y & \qquad \times\ \mathsf{0}\ y &\to \mathsf{0} \\ + \ (\mathsf{s}\ x)\ y &\to \mathsf{s}\ (+\ x\ y) & \qquad \times\ (\mathsf{s}\ x)\ y &\to +\ (\times\ x\ y)\ y \end{aligned}$$

Every rewrite sequence in $\mathcal{R}$ can be transformed into a sequence in $\mathcal{R}{\downarrow_\mathcal{C}}$, but the reverse does not hold. For instance, with respect to the above example, the rewrite step $+\ (\mathsf{s}\ (+\ \mathsf{0}))\ \mathsf{0} \to \mathsf{s}\ (+\ (+\ \mathsf{0})\ \mathsf{0})$ in $\mathcal{R}{\downarrow_\mathcal{C}}$ does not correspond to a rewrite step in $\mathcal{R}$. Nevertheless, termination of $\mathcal{R}$ implies termination of $\mathcal{R}{\downarrow_\mathcal{C}}$.

Theorem 8.6 (Kennaway *et al.* [95])**.** *A TRS $\mathcal{R}$ is terminating if and only if $\mathcal{R}{\downarrow_\mathcal{C}}$ is terminating.* ☐

A simple self-labeling proof can be found in [121]. As an immediate consequence we get the following transformation method for proving termination of ATRSs.

Corollary 8.7. *An ATRS $\mathcal{R}$ is terminating if and only if there exists a terminating TRS $\mathcal{S}$ such that $\mathcal{S}{\downarrow_\mathcal{C}} = \mathcal{R}$ (modulo renaming).* ☐

In [59] this method is called *transformation $\mathcal{A}$*. As can be seen from the following example, the method does not handle partially applied terms and, more seriously, head variables. Hence the method is of limited applicability as it cannot cope with the higher-order aspects modeled by ATRSs.

Example 8.8. Consider the ATRS $\mathcal{R}$ (from [9])

$$\begin{aligned} 1:&\quad \mathsf{id}\ x \to x & \qquad 4:&\quad \mathsf{map}\ f\ \mathsf{nil} \to \mathsf{nil} \\ 2:&\quad \mathsf{add}\ \mathsf{0} \to \mathsf{id} & \qquad 5:&\quad \mathsf{map}\ f\ (:\ x\ y) \to\ :\ (f\ x)\ (\mathsf{map}\ f\ y) \\ 3:&\quad \mathsf{add}\ (\mathsf{s}\ x)\ y \to \mathsf{s}\ (\mathsf{add}\ x\ y) \end{aligned}$$

Rules 1 and 4 are readily translated into functional form:

$$\mathsf{id}_1(x) \to x \qquad\qquad \mathsf{map}_2(f, \mathsf{nil}) \to \mathsf{nil}$$

However, we cannot find functional forms for rules 2 and 3 because the 'arity' of add is 1 in rule 2 and 2 in rule 3. Because of the presence of the head variable f in the subterm $f\ x$, there is no functional term t such that $t{\downarrow_\mathcal{C}} =\ :\ (f\ x)\ (\mathsf{map}\ f\ y)$. Hence also rule 5 cannot be transformed.

8.3. Full Termination

In this section we present an uncurrying transformation that can deal with ATRSs like in Example 8.8. This transformation preserves and reflects termination.

Throughout this section we assume that $\mathcal{R}$ is an ATRS over an applicative signature $\mathcal{F}$. In the sequel we restrict to TRSs where $\mathrm{aa}(f)$ (see next definition) is defined for every $f \in \mathcal{F}$. Note that $\mathrm{aa}(f)$ is defined if $\mathcal{R}$ is finite but may be undefined for infinite $\mathcal{R}$.

Definition 8.9. The *applicative arity* $\mathrm{aa}(f)$ of a constant $f \in \mathcal{F}$ is defined as the maximum n such that $f \star t_1 \star \cdots \star t_n$ is a subterm in the left- or right-hand side of a rule in $\mathcal{R}$. This notion is extended to terms as follows:

$$\mathrm{aa}(t) = \begin{cases} \mathrm{aa}(f) & \text{if } t \text{ is a constant } f \\ \mathrm{aa}(t_1) - 1 & \text{if } t = t_1 \star t_2 \end{cases}$$

Note that $\mathrm{aa}(t)$ is undefined if the head symbol of t is a variable and $\mathrm{aa}(f) = 0$ for a constant $f \in \mathcal{F}$ that does not appear in a rule in $\mathcal{R}$.

Definition 8.10. For an ATRS $\mathcal{R}$ the uncurrying system $\mathcal{U}(\mathcal{R})$ consists of the following rewrite rules $f_i(x_1, \ldots, x_i) \star y \to f_{i+1}(x_1, \ldots, x_i, y)$ for every constant $f \in \mathcal{F}$ and every $0 \leqslant i < \mathrm{aa}(f)$. Here $f_0 = f$ and, for every $i > 0$, f_i is a fresh function symbol of arity i. We say that $\mathcal{R}$ is *left head variable free* if no subterm of a left-hand side in $\mathcal{R}$ is of the form $t_1 \star t_2$ where t_1 is a variable. We write *ℓ-ATRS* to denote a left head variable free ATRS.

The uncurrying system $\mathcal{U}(\mathcal{R})$, or simply $\mathcal{U}$, is confluent and terminating. Hence every term t has a unique normal form $t{\downarrow_{\mathcal{U}}}$.

Definition 8.11. The *uncurried* system $\mathcal{R}{\downarrow_{\mathcal{U}}}$ is the TRS consisting of the rules $\ell{\downarrow_{\mathcal{U}}} \to r{\downarrow_{\mathcal{U}}}$ for every $\ell \to r \in \mathcal{R}$.

Example 8.12. The ATRS $\mathcal{R}$ of Example 8.8 is transformed into $\mathcal{R}{\downarrow_{\mathcal{U}}}$:

$$\begin{aligned} \mathsf{id}_1(x) &\to x & \mathsf{map}_2(f, \mathsf{nil}) &\to \mathsf{nil} \\ \mathsf{add}_1(\mathsf{0}) &\to \mathsf{id} & \mathsf{map}_2(f, {:}_2(x, y)) &\to {:}_2(f \star x, \mathsf{map}_2(f, y)) \\ \mathsf{add}_2(\mathsf{s}_1(x), y) &\to \mathsf{s}_1(\mathsf{add}_2(x, y)) \end{aligned}$$

The TRS $\mathcal{R}{\downarrow_{\mathcal{U}}}$ is an obvious candidate of a TRS whose termination implies termination of the original ATRS. However, as can be seen from the following example, the rules of $\mathcal{R}{\downarrow_{\mathcal{U}}}$ are not enough to simulate an arbitrary rewrite sequence in $\mathcal{R}$.

Example 8.13. The ATRS $\mathcal{R}$

$$\mathsf{f}\ x \to x\ x$$

is non-terminating since $\mathsf{f}\ \mathsf{f} \to_{\mathcal{R}} \mathsf{f}\ \mathsf{f} \to_{\mathcal{R}} \cdots$ while the transformed TRS $\mathcal{R}{\downarrow_{\mathcal{U}}}$

$$\mathsf{f}_1(x) \to x \star x$$

is terminating.

The natural solution is to add $\mathcal{U}(\mathcal{R})$. In the following we write $\mathcal{U}^{+}(\mathcal{R})$ for $\mathcal{R}{\downarrow_{\mathcal{U}(\mathcal{R})}} \cup \mathcal{U}(\mathcal{R})$. As the following example shows, we do not yet have a sound transformation.

Example 8.14. The non-terminating ATRS $\mathcal{R}$

$$\mathsf{id}\ x \to x \qquad\qquad \mathsf{f}\ x \to \mathsf{id}\ \mathsf{f}\ x$$

is transformed into the terminating TRS $\mathcal{R}{\downarrow_{\mathcal{U}}}$

$$\mathsf{id}_1(x) \to x \qquad\qquad \mathsf{f}_1(x) \to \mathsf{id}_2(\mathsf{f}, x)$$

Note that $\mathrm{aa}(\mathsf{id}) = 2$ and $\mathrm{aa}(\mathsf{f}) = 1$. The TRS $\mathcal{U}^{+}(\mathcal{R})$ consists of the following rules

$$\begin{array}{ccc} \mathsf{id}_1(x) \to x & \mathsf{id} \star x \to \mathsf{id}_1(x) & \mathsf{f} \star x \to \mathsf{f}_1(x) \\ \mathsf{f}_1(x) \to \mathsf{id}_2(\mathsf{f}, x) & \mathsf{id}_1(x) \star y \to \mathsf{id}_2(x, y) & \end{array}$$

and is easily shown to be terminating.

The ATRS $\mathcal{R}$ admits the cycle $\mathsf{f}\ x \to \mathsf{id}\ \mathsf{f}\ x \to \mathsf{f}\ x$. In $\mathcal{U}^{+}(\mathcal{R})$ we have the rule $\mathsf{f}_1(x) \to \mathsf{id}_2(\mathsf{f}, x)$ but the term $\mathsf{id}_2(\mathsf{f}, x)$ does not rewrite to $\mathsf{f}_1(x)$. It would if the rule $\mathsf{id}\ x\ y \to x\ y$ were present in $\mathcal{R}$. This inspires the following definition.

Definition 8.15. Let $\mathcal{R}$ be an ATRS. The *η-saturated* ATRS $\mathcal{R}_\eta$ is the smallest extension of $\mathcal{R}$ such that $\ell \star x \to r \star x \in \mathcal{R}_\eta$ whenever $\ell \to r \in \mathcal{R}_\eta$ and $\mathrm{aa}(\ell) > 0$. Here x is a variable that does not appear in $\ell \to r$.

The rules added during η-saturation do not affect the termination behavior of $\mathcal{R}$, according to the following lemma. Moreover, $\mathcal{R}_\eta$ is an ℓ-ATRS if and only if $\mathcal{R}$ is an ℓ-ATRS.

Lemma 8.16. *If $\mathcal{R}$ is an ℓ-ATRS then $\to_{\mathcal{R}}$ and $\to_{\mathcal{R}_\eta}$ coincide.*

Proof. The inclusion $\to_{\mathcal{R}} \subseteq \to_{\mathcal{R}_\eta}$ trivially follows from the inclusion $\mathcal{R} \subseteq \mathcal{R}_\eta$. For the reverse inclusion we show that for every rewrite step $s = C[\ell\sigma] \to_{\mathcal{R}_\eta} C[r\sigma] = t$ there exist a rule $\ell' \to r' \in \mathcal{R}$ and a context C' such that $s = C'[\ell'\sigma]$ and $t = C'[r'\sigma]$. We use induction on the derivation of $\ell \to r \in \mathcal{R}_\eta$. In the base case $\ell \to r \in \mathcal{R}$ and we simply take $\ell' \to r' = \ell \to r$ and $C' = C$. In the induction step we have $\ell \to r = \ell' \star x \to r' \star x$ for some $\ell' \to r' \in \mathcal{R}_\eta$ with $\mathrm{aa}(\ell') > 0$. Define $C' = C[\Box \star x\sigma]$. Clearly $s = C'[\ell'\sigma] \to_{\mathcal{R}_\eta} C'[r'\sigma] = t$. The induction hypothesis yields a rule $\ell'' \to r'' \in \mathcal{R}$ and a context C'' such that $s = C''[\ell''\sigma]$ and $t = C''[r''\sigma]$. $\Box$

In the following we write $\mathcal{U}_\eta^+(\mathcal{R})$ for $\mathcal{R}_\eta{\downarrow_{\mathcal{U}(\mathcal{R})}} \cup \mathcal{U}(\mathcal{R})$. We can now state the first major result of this article.

Theorem 8.17. *An ℓ-ATRS $\mathcal{R}$ is terminating if $\mathcal{U}_\eta^+(\mathcal{R})$ is terminating.*

Before we prepare for the proof of the theorem above we show another subtlety of the uncurrying transformation. While η-saturation may change applicative arities, the applicative arities used in the definition of $\mathcal{U}_\eta^+(\mathcal{R})$ always refer to those of $\mathcal{R}$ and not $\mathcal{R}_\eta$.

Example 8.18. The non-terminating ATRS $\mathcal{R}$

$$\mathsf{f} \to \mathsf{g}\ \mathsf{a} \qquad\qquad \mathsf{g} \to \mathsf{f}$$

is transformed into $\mathcal{U}_\eta^+(\mathcal{R})$

$$\mathsf{f} \to \mathsf{g}_1(\mathsf{a}) \qquad \mathsf{g} \to \mathsf{f} \qquad \mathsf{g}_1(x) \to \mathsf{f} \star x \qquad \mathsf{g} \star x \to \mathsf{g}_1(x)$$

because $\mathrm{aa}(\mathsf{f}) = 0$. The resulting TRS is non-terminating. If the applicative arities of $\mathcal{R}_\eta$ were employed, uncurrying would produce the terminating TRS

$$\mathsf{f} \to \mathsf{g}_1(\mathsf{a}) \qquad \mathsf{g} \to \mathsf{f} \qquad \mathsf{g}_1(x) \to \mathsf{f}_1(x) \qquad \mathsf{g} \star x \to \mathsf{g}_1(x) \qquad \mathsf{f} \star x \to \mathsf{f}_1(x)$$

since $\mathrm{aa}(\mathsf{f}) = 1$ for $\mathcal{R}_\eta$.

Before presenting the proof of Theorem 8.17, we revisit the running example.

Example 8.19. Consider again the ATRS $\mathcal{R}$ of Example 8.8. Proving termination of the transformed TRS $\mathcal{U}_\eta^+(\mathcal{R})$

$$\begin{array}{rcc r}
\mathsf{id}_1(x) \to x & & {:} \star x \to {:}_1(x) & \mathsf{id} \star x \to \mathsf{id}_1(x) \\
\mathsf{add}_1(0) \to \mathsf{id} & & {:}_1(x) \star y \to {:}_2(x,y) & \mathsf{add} \star x \to \mathsf{add}_1(x) \\
\mathsf{add}_2(0,y) \to \mathsf{id}_1(y) & & & \mathsf{add}_1(x) \star y \to \mathsf{add}_2(x,y) \\
\mathsf{add}_2(\mathsf{s}_1(x),y) \to \mathsf{s}_1(\mathsf{add}_2(x,y)) & & & \mathsf{s} \star x \to \mathsf{s}_1(x) \\
\mathsf{map}_2(f,\mathsf{nil}) \to \mathsf{nil} & & & \mathsf{map} \star x \to \mathsf{map}_1(x) \\
\mathsf{map}_2(f,{:}_2(x,y)) \to {:}_2(f \star x, \mathsf{map}_2(f,y)) & & & \mathsf{map}_1(x) \star y \to \mathsf{map}_2(x,y)
\end{array}$$

is possible using LPO with a quasi-precedence.

The next lemma states an easy result that is freely used in the sequel.

Lemma 8.20. *Let s be an applicative term. If $s = x \star s_1 \star \cdots \star s_n$ then $s{\downarrow_\mathcal{U}} = x \star s_1{\downarrow_\mathcal{U}} \star \cdots \star s_n{\downarrow_\mathcal{U}}$ and if $s = f \star s_1 \star \cdots \star s_n$ then $s{\downarrow_\mathcal{U}} = f_i(s_1{\downarrow_\mathcal{U}}, \ldots, s_i{\downarrow_\mathcal{U}}) \star s_{i+1}{\downarrow_\mathcal{U}} \star \cdots \star s_n{\downarrow_\mathcal{U}}$ for $i = \min\{\mathrm{aa}(f), n\}$.*

Proof. We show the second claim by induction on n (the first one is similar). In the base case $n = 0$ and $f{\downarrow_{\mathcal{U}}} = f$ concludes this case. In the inductive step

$$s \to_{\mathcal{U}}^{*} s' = f_j(s_1{\downarrow_{\mathcal{U}}}, \ldots, s_j{\downarrow_{\mathcal{U}}}) \star s_{j+1}{\downarrow_{\mathcal{U}}} \star \cdots \star s_{n+1}{\downarrow_{\mathcal{U}}}$$

for $j = \min\{\mathrm{aa}(f), n\}$ follows from the induction hypothesis and the definition of $(\cdot){\downarrow_{\mathcal{U}}}$. If $i < n + 1$ then $j = i$ and $s' = s{\downarrow_{\mathcal{U}}}$. In the case where $i = n + 1$ then $j = n$ and the result follows from

$$f_n(s_1{\downarrow_{\mathcal{U}}}, \ldots, s_n{\downarrow_{\mathcal{U}}}) \star s_{n+1}{\downarrow_{\mathcal{U}}} \to_{\mathcal{U}} f_{n+1}(s_1{\downarrow_{\mathcal{U}}}, \ldots, s_n{\downarrow_{\mathcal{U}}}, s_{n+1}{\downarrow_{\mathcal{U}}}) = s{\downarrow_{\mathcal{U}}}.$$

This concludes the proof. □

The following two lemmata state factorization properties which are used in the proof of Theorem 8.17.

Lemma 8.21. *Let $s_1, \ldots, s_n$ be applicative terms. Then $s_1{\downarrow_{\mathcal{U}}} \star \cdots \star s_n{\downarrow_{\mathcal{U}}} \to_{\mathcal{U}}^{*} (s_1 \star \cdots \star s_n){\downarrow_{\mathcal{U}}}$. If $\mathrm{aa}(s_1) \leqslant 0$ or if $\mathrm{aa}(s_1)$ is undefined then $s_1{\downarrow_{\mathcal{U}}} \star \cdots \star s_n{\downarrow_{\mathcal{U}}} = (s_1 \star \cdots \star s_n){\downarrow_{\mathcal{U}}}$.*

Proof. Observe that $s_1{\downarrow_{\mathcal{U}}} \star \cdots \star s_n{\downarrow_{\mathcal{U}}} \ {}_{\mathcal{U}}^{*}\!\!\leftarrow s_1 \star \cdots \star s_n \to_{\mathcal{U}}^{!} (s_1 \star \cdots \star s_n){\downarrow_{\mathcal{U}}}$. The first claim then follows from the confluence of $\mathcal{U}$.

For the second claim observe that if $\mathrm{aa}(s_1) \leqslant 0$ or if $\mathrm{aa}(s_1)$ is undefined then $s_1{\downarrow_{\mathcal{U}}} \star \cdots \star s_n{\downarrow_{\mathcal{U}}} \in \mathit{NF}(\mathcal{U})$ and the result follows from the confluence of $\mathcal{U}$ as in the first case. □

For an applicative substitution σ, we write $\sigma{\downarrow_{\mathcal{U}}}$ for the substitution $\{x \mapsto \sigma(x){\downarrow_{\mathcal{U}}} \mid x \in \mathcal{V}\}$.

Lemma 8.22. *Let σ be an applicative substitution. For every applicative term t, $t{\downarrow_{\mathcal{U}}}\sigma{\downarrow_{\mathcal{U}}} \to_{\mathcal{U}}^{*} (t\sigma){\downarrow_{\mathcal{U}}}$. If t is head variable free then $t{\downarrow_{\mathcal{U}}}\sigma{\downarrow_{\mathcal{U}}} = (t\sigma){\downarrow_{\mathcal{U}}}$.*

Proof. We prove the former claim by induction on the term t. The proof for the head variable free case is similar. It suffices to consider the step case.

- Consider the step case with $t = x \star t_1 \star \cdots \star t_n$. Then

$$\begin{array}{rcl} t{\downarrow_{\mathcal{U}}}\sigma{\downarrow_{\mathcal{U}}} & = & (x \star t_1{\downarrow_{\mathcal{U}}} \star \cdots \star t_n{\downarrow_{\mathcal{U}}})\sigma{\downarrow_{\mathcal{U}}} \\ & = & \sigma(x){\downarrow_{\mathcal{U}}} \star t_1{\downarrow_{\mathcal{U}}}\sigma{\downarrow_{\mathcal{U}}} \star \cdots \star t_n{\downarrow_{\mathcal{U}}}\sigma{\downarrow_{\mathcal{U}}} \\ & \to_{\mathcal{U}}^{*} & \sigma(x){\downarrow_{\mathcal{U}}} \star (t_1\sigma){\downarrow_{\mathcal{U}}} \star \cdots \star (t_n\sigma){\downarrow_{\mathcal{U}}} \\ & \to_{\mathcal{U}}^{*} & (\sigma(x) \star t_1\sigma \star \cdots \star t_n\sigma){\downarrow_{\mathcal{U}}} \ = \ (t\sigma){\downarrow_{\mathcal{U}}} \end{array}$$

  where Lemma 8.20 is applied in the first equality, the induction hypothesis in the first $\to_{\mathcal{U}}^{*}$ step, and Lemma 8.21 in the second $\to_{\mathcal{U}}^{*}$ step.

- Consider the step case with $t = f \star t_1 \star \cdots \star t_n$. Let $i = \min\{\mathrm{aa}(f), n\}$. Then

$$\begin{aligned}
t{\downarrow_{\mathcal{U}}}\sigma{\downarrow_{\mathcal{U}}} &= (f_i(t_1{\downarrow_{\mathcal{U}}}, \dots, t_i{\downarrow_{\mathcal{U}}}) \star t_{i+1}{\downarrow_{\mathcal{U}}} \star \cdots \star t_n{\downarrow_{\mathcal{U}}})\sigma{\downarrow_{\mathcal{U}}} \\
&= f_i(t_1{\downarrow_{\mathcal{U}}}\sigma{\downarrow_{\mathcal{U}}}, \dots, t_i{\downarrow_{\mathcal{U}}}\sigma{\downarrow_{\mathcal{U}}}) \star t_{i+1}{\downarrow_{\mathcal{U}}}\sigma{\downarrow_{\mathcal{U}}} \star \cdots \star t_n{\downarrow_{\mathcal{U}}}\sigma{\downarrow_{\mathcal{U}}} \\
&\to_{\mathcal{U}}^{*} f_i((t_1\sigma){\downarrow_{\mathcal{U}}}, \dots, (t_i\sigma){\downarrow_{\mathcal{U}}}) \star (t_{i+1}\sigma){\downarrow_{\mathcal{U}}} \star \cdots \star (t_n\sigma){\downarrow_{\mathcal{U}}} \\
&= f_i(t_1\sigma, \dots, t_i\sigma){\downarrow_{\mathcal{U}}} \star (t_{i+1}\sigma){\downarrow_{\mathcal{U}}} \star \cdots \star (t_n\sigma){\downarrow_{\mathcal{U}}} \\
&= (f \star t_1\sigma \star \cdots \star t_i\sigma){\downarrow_{\mathcal{U}}} \star (t_{i+1}\sigma){\downarrow_{\mathcal{U}}} \star \cdots \star (t_n\sigma){\downarrow_{\mathcal{U}}} \;=\; (t\sigma){\downarrow_{\mathcal{U}}}
\end{aligned}$$

  where Lemma 8.20 is applied in the first equality, the induction hypothesis in the first $\to_{\mathcal{U}}^{*}$ step and Lemma 8.21 in the last equality. □

Now we are ready to present the proof of Theorem 8.17.

Proof of Theorem 8.17. We show that $s{\downarrow_{\mathcal{U}}} \to_{\mathcal{U}_\eta^+(\mathcal{R})}^{+} t{\downarrow_{\mathcal{U}}}$ whenever $s \to_{\mathcal{R}_\eta} t$ for applicative terms s and t. This entails that any infinite $\mathcal{R}_\eta$ derivation is transformed into an infinite $\mathcal{U}_\eta^+(\mathcal{R})$ derivation. The theorem follows from this observation and Lemma 8.16. Let $s = C[\ell\sigma]$ and $t = C[r\sigma]$ with $\ell \to r \in \mathcal{R}_\eta$. We use induction on the size of the context C.

- If $C = \Box$ then $s{\downarrow_{\mathcal{U}}} = (\ell\sigma){\downarrow_{\mathcal{U}}} = \ell{\downarrow_{\mathcal{U}}}\sigma{\downarrow_{\mathcal{U}}}$ and $r{\downarrow_{\mathcal{U}}}\sigma{\downarrow_{\mathcal{U}}} \to_{\mathcal{U}}^{*} (r\sigma){\downarrow_{\mathcal{U}}} = t{\downarrow_{\mathcal{U}}}$ by Lemma 8.22. Hence $s{\downarrow_{\mathcal{U}}} \to_{\mathcal{U}_\eta^+(\mathcal{R})}^{+} t{\downarrow_{\mathcal{U}}}$.

- Suppose $C = \Box \star s_1 \star \cdots \star s_n$ and $n > 0$. Since $\mathcal{R}_\eta$ is left head variable free, $\mathrm{aa}(\ell)$ is defined. If $\mathrm{aa}(\ell) = 0$ then

$$\begin{aligned}
s{\downarrow_{\mathcal{U}}} &= (\ell\sigma \star s_1 \star \cdots \star s_n){\downarrow_{\mathcal{U}}} = (\ell\sigma){\downarrow_{\mathcal{U}}} \star s_1{\downarrow_{\mathcal{U}}} \star \cdots \star s_n{\downarrow_{\mathcal{U}}} \\
&= \ell{\downarrow_{\mathcal{U}}}\sigma{\downarrow_{\mathcal{U}}} \star s_1{\downarrow_{\mathcal{U}}} \star \cdots \star s_n{\downarrow_{\mathcal{U}}}
\end{aligned}$$

  and

$$\begin{aligned}
r{\downarrow_{\mathcal{U}}}\sigma{\downarrow_{\mathcal{U}}} \star s_1{\downarrow_{\mathcal{U}}} \star \cdots \star s_n{\downarrow_{\mathcal{U}}} &\to_{\mathcal{U}}^{*} (r\sigma){\downarrow_{\mathcal{U}}} \star s_1{\downarrow_{\mathcal{U}}} \star \cdots \star s_n{\downarrow_{\mathcal{U}}} \\
&\to_{\mathcal{U}}^{*} (r\sigma \star s_1 \star \cdots \star s_n){\downarrow_{\mathcal{U}}} = t{\downarrow_{\mathcal{U}}}
\end{aligned}$$

  by applications of Lemmata 8.22 and 8.21. Hence $s{\downarrow_{\mathcal{U}}} \to_{\mathcal{U}_\eta^+(\mathcal{R})}^{+} t{\downarrow_{\mathcal{U}}}$. If $\mathrm{aa}(\ell) > 0$ then $\ell \star x \to r \star x \in \mathcal{R}_\eta$ for some fresh variable x. We have $s = C'[(\ell \star x)\tau]$ and $t = C'[(r \star x)\tau]$ for the context $C' = \Box \star s_2 \star \cdots \star s_n$ and the substitution $\tau = \sigma \cup \{x \mapsto s_1\}$. Since C' is smaller than C, we can apply the induction hypothesis which yields the desired result.

- In the remaining case $C = s_1 \star C'$. The induction hypothesis yields

  $$C'[\ell\sigma]{\downarrow_{\mathcal{U}}} \to^{+}_{\mathcal{U}^{+}_{\eta}(\mathcal{R})} C'[r\sigma]{\downarrow_{\mathcal{U}}}$$

  If $\mathrm{aa}(s_1) \leqslant 0$ or if $\mathrm{aa}(s_1)$ is undefined then $s{\downarrow_{\mathcal{U}}} = s_1{\downarrow_{\mathcal{U}}} \star C'[\ell\sigma]{\downarrow_{\mathcal{U}}}$ and $t{\downarrow_{\mathcal{U}}} = s_1{\downarrow_{\mathcal{U}}} \star C'[r\sigma]{\downarrow_{\mathcal{U}}}$ by Lemma 8.21. If $\mathrm{aa}(s_1) > 0$ then $s_1{\downarrow_{\mathcal{U}}} = f_i(u_1, \ldots, u_i)$ for the head symbol f of s_1 and some terms $u_1, \ldots, u_i$. So

  $$s{\downarrow_{\mathcal{U}}} = f_{i+1}(u_1, \ldots, u_i, C'[\ell\sigma]{\downarrow_{\mathcal{U}}})$$

  and

  $$t{\downarrow_{\mathcal{U}}} = f_{i+1}(u_1, \ldots, u_i, C'[r\sigma]{\downarrow_{\mathcal{U}}}).$$

  Hence in both cases we obtain $s{\downarrow_{\mathcal{U}}} \to^{+}_{\mathcal{U}^{+}_{\eta}(\mathcal{R})} t{\downarrow_{\mathcal{U}}}$. □

The next example shows that the left head variable freeness condition cannot be weakened to the well-definedness of $\mathrm{aa}(\ell)$ for every left-hand side ℓ.

Example 8.23. Consider the non-terminating ATRS $\mathcal{R}$

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

The transformed TRS $\mathcal{U}^{+}_{\eta}(\mathcal{R})$ consists of the rules

$$\begin{array}{ccc} \mathsf{f}_1(x \star \mathsf{a}) \to \mathsf{f}_1(\mathsf{g}_1(\mathsf{b})) & \mathsf{f} \star x \to \mathsf{f}_1(x) & \mathsf{h} \star x \to \mathsf{h}_1(x) \\ \mathsf{g}_1(\mathsf{b}) \to \mathsf{h}_1(\mathsf{a}) & \mathsf{g} \star x \to \mathsf{g}_1(x) & \end{array}$$

and is terminating because its rules are oriented from left to right by the lexicographic path order with precedence $\star > \mathsf{g}_1 > \mathsf{f}_1 > \mathsf{h}_1 > \mathsf{a} > \mathsf{b}$. Note that $\mathrm{aa}(\mathsf{f}\ (x\ \mathsf{a})) = 0$.

The uncurrying transformation is not always useful.

Example 8.24. Consider the one-rule TRS $\mathcal{R}$

$$\mathsf{C}\ x\ y\ z\ u \to x\ z\ (x\ y\ z\ u)$$

from [38]. The termination of $\mathcal{R}$ is proved by the lexicographic path order with empty precedence. The transformed TRS $\mathcal{U}^{+}_{\eta}(\mathcal{R})$ consists of

$$\begin{array}{cc} \mathsf{C}_4(x, y, z, u) \to x \star z \star (x \star y \star z \star u) & \\ \mathsf{C} \star x \to \mathsf{C}_1(x) & \mathsf{C}_2(x, y) \star z \to \mathsf{C}_3(x, y, z) \\ \mathsf{C}_1(x) \star y \to \mathsf{C}_2(x, y) & \mathsf{C}_3(x, y, z) \star u \to \mathsf{C}_4(x, y, z, u) \end{array}$$

None of the tools that participated in the termination competitions between 2005 and 2010 is able to prove the termination of this TRS.

We show that the converse of Theorem 8.17 also holds. Hence the uncurrying transformation does not only reflect but also preserve termination. (This does not contradict the preceding example.) To show this result (Theorem 8.28) we transform any infinite sequence in $\mathcal{U}^+_\eta(\mathcal{R})$ into an infinite sequence of the original ℓ-ATRS $\mathcal{R}$. Since $\mathcal{U}$ is terminating, any infinite sequence in $\mathcal{U}^+_\eta(\mathcal{R})$ must have the shape

$$\to_{\mathcal{R}_\eta{\downarrow_\mathcal{U}}} \cdot \to^*_\mathcal{U} \cdot \to_{\mathcal{R}_\eta{\downarrow_\mathcal{U}}} \cdot \to^*_\mathcal{U} \cdots \tag{8.1}$$

with infinitely many $\mathcal{R}_\eta{\downarrow_\mathcal{U}}$ steps. Below we write $\mathcal{C}'$ for the TRS that is obtained from $\mathcal{U}$ by reversing all rules: $\{r \to \ell \mid \ell \to r \in \mathcal{U}(\mathcal{R})\}$ The TRS $\mathcal{C}'$ allows to mimic a rewrite step $\to_{\mathcal{R}_\eta{\downarrow_\mathcal{U}}}$ in the original ℓ-ATRS $\mathcal{R}$ (Lemma 8.26) and equates any two terms that are in the relation $\to^*_\mathcal{U}$ (Lemma 8.27). Then the sequence (8.1) can be transformed into an infinite sequence in $\mathcal{R}$.

Remark 8.1. *Note that* ${\downarrow_\mathcal{C}}$ *is not the inverse of* ${\downarrow_\mathcal{U}}$ *since for the TRS* $\mathcal{R}$ *from Example 8.14 we have* $\mathcal{R}{\downarrow_\mathcal{U}}{\downarrow_\mathcal{C}} = \{\mathsf{id}_1\ x \to x, \mathsf{f}_1\ x \to \mathsf{id}_2\ \mathsf{f}\ x\}$ *which is different from* $\mathcal{R}$. *But obviously* $\mathcal{R}{\downarrow_\mathcal{U}}{\downarrow_{\mathcal{C}'}} = \mathcal{R}$ *for any ATRS* $\mathcal{R}$.

Lemma 8.25. *Let* $\mathcal{R}$ *be an* ℓ*-ATRS. If* t, C, *and* σ *are a term, context, and substitution over the signature of* $\mathcal{U}^+_\eta(\mathcal{R})$, *then* $(C[t\sigma]){\downarrow_{\mathcal{C}'}} = C{\downarrow_{\mathcal{C}'}}[t{\downarrow_{\mathcal{C}'}}\sigma{\downarrow_{\mathcal{C}'}}]$.

Proof. We show $(t\sigma){\downarrow_{\mathcal{C}'}} = t{\downarrow_{\mathcal{C}'}}\sigma{\downarrow_{\mathcal{C}'}}$ by induction on t. If $t = x$ then the result follows since $(x\sigma){\downarrow_{\mathcal{C}'}} = \sigma(x){\downarrow_{\mathcal{C}'}} = x{\downarrow_{\mathcal{C}'}}\sigma{\downarrow_{\mathcal{C}'}}$. If $t = f_i(t_1, \ldots, t_i)$ then

$$\begin{aligned}(t\sigma){\downarrow_{\mathcal{C}'}} &= f_i(t_1\sigma, \ldots, t_i\sigma){\downarrow_{\mathcal{C}'}} = f \star (t_1\sigma){\downarrow_{\mathcal{C}'}} \star \cdots \star (t_i\sigma){\downarrow_{\mathcal{C}'}}\\ &= f \star t_1{\downarrow_{\mathcal{C}'}}\sigma{\downarrow_{\mathcal{C}'}} \star \cdots \star t_i{\downarrow_{\mathcal{C}'}}\sigma{\downarrow_{\mathcal{C}'}} = (f \star t_1{\downarrow_{\mathcal{C}'}} \star \cdots \star t_i{\downarrow_{\mathcal{C}'}})\sigma{\downarrow_{\mathcal{C}'}}\\ &= f_i(t_1, \ldots, t_i){\downarrow_{\mathcal{C}'}}\sigma{\downarrow_{\mathcal{C}'}}\end{aligned}$$

by the induction hypothesis. If $t = t_1 \star t_2$ then

$$\begin{aligned}(t\sigma){\downarrow_{\mathcal{C}'}} &= (t_1\sigma \star t_2\sigma){\downarrow_{\mathcal{C}'}} = (t_1\sigma){\downarrow_{\mathcal{C}'}} \star (t_2\sigma){\downarrow_{\mathcal{C}'}} = t_1{\downarrow_{\mathcal{C}'}}\sigma{\downarrow_{\mathcal{C}'}} \star t_2{\downarrow_{\mathcal{C}'}}\sigma{\downarrow_{\mathcal{C}'}}\\ &= (t_1{\downarrow_{\mathcal{C}'}} \star t_2{\downarrow_{\mathcal{C}'}})\sigma{\downarrow_{\mathcal{C}'}} = t{\downarrow_{\mathcal{C}'}}\sigma{\downarrow_{\mathcal{C}'}}\end{aligned}$$

by the induction hypothesis.

The proof that $(C[u]){\downarrow_{\mathcal{C}'}} = C{\downarrow_{\mathcal{C}'}}[u{\downarrow_{\mathcal{C}'}}]$ is by induction on C and similar. The lemma then follows from these two results. ☐

Lemma 8.26. *Let* $\mathcal{R}$ *be an* ℓ*-ATRS. If* s *and* t *are terms over the signature of* $\mathcal{U}^+_\eta(\mathcal{R})$ *then* $s \to_{\mathcal{R}_\eta{\downarrow_\mathcal{U}}} t$ *implies* $s{\downarrow_{\mathcal{C}'}} \to_{\mathcal{R}_\eta} t{\downarrow_{\mathcal{C}'}}$.

Proof. From $s \to_{\mathcal{R}_\eta{\downarrow_\mathcal{U}}} t$ we get $s = C[\ell\sigma]$, $t = C[r\sigma]$ for some $\ell \to r \in \mathcal{R}_\eta{\downarrow_\mathcal{U}}$. By Lemma 8.25 we obtain $s{\downarrow_{\mathcal{C}'}} = C{\downarrow_{\mathcal{C}'}}[\ell{\downarrow_{\mathcal{C}'}}\sigma{\downarrow_{\mathcal{C}'}}]$ and $t{\downarrow_{\mathcal{C}'}} = C{\downarrow_{\mathcal{C}'}}[r{\downarrow_{\mathcal{C}'}}\sigma{\downarrow_{\mathcal{C}'}}]$. The result then follows in connection with the fact that $\mathcal{R}_\eta{\downarrow_\mathcal{U}}{\downarrow_{\mathcal{C}'}} = \mathcal{R}_\eta$. ☐

Lemma 8.27. *Let $\mathcal{R}$ be an ℓ-ATRS. If s and t are terms over the signature of $\mathcal{U}_\eta^+(\mathcal{R})$ then $s \to_\mathcal{U} t$ implies $s{\downarrow_{\mathcal{C}'}} = t{\downarrow_{\mathcal{C}'}}$.*

Proof. From $s \to_\mathcal{U} t$ we get $s = C[\ell\sigma]$, $t = C[r\sigma]$ for some $\ell \to r \in \mathcal{U}$. By Lemma 8.25 we obtain $s{\downarrow_{\mathcal{C}'}} = C{\downarrow_{\mathcal{C}'}}[\ell{\downarrow_{\mathcal{C}'}}\sigma{\downarrow_{\mathcal{C}'}}]$ and $t{\downarrow_{\mathcal{C}'}} = C{\downarrow_{\mathcal{C}'}}[r{\downarrow_{\mathcal{C}'}}\sigma{\downarrow_{\mathcal{C}'}}]$. The result then follows in connection with the observation that all rules in $\mathcal{U}{\downarrow_{\mathcal{C}'}}$ have equal left- and right-hand sides. □

Theorem 8.28. *If $\mathcal{R}$ is a terminating ℓ-ATRS then $\mathcal{U}_\eta^+(\mathcal{R})$ is terminating.*

Proof. Assume that $\mathcal{U}_\eta^+(\mathcal{R})$ is non-terminating. Since $\mathcal{U}$ is terminating, any infinite rewrite sequence has the form

$$s_1 \to_{\mathcal{R}_\eta{\downarrow_\mathcal{U}}} t_1 \to_\mathcal{U}^* s_2 \to_{\mathcal{R}_\eta{\downarrow_\mathcal{U}}} t_2 \to_\mathcal{U}^* \cdots$$

Applications of Lemmata 8.26 and 8.27 transform this sequence into

$$s_1{\downarrow_{\mathcal{C}'}} \to_{\mathcal{R}_\eta} t_1{\downarrow_{\mathcal{C}'}} = s_2{\downarrow_{\mathcal{C}'}} \to_{\mathcal{R}_\eta} t_2{\downarrow_{\mathcal{C}'}} = \cdots$$

It follows that $\mathcal{R}_\eta$ is non-terminating. Since $\to_\mathcal{R} = \to_{\mathcal{R}_\eta}$ by Lemma 8.16, we conclude that $\mathcal{R}$ is non-terminating. □

Next we describe a trivial mirroring technique for TRSs. This technique can be used to eliminate some of the left head variables in an ATRS.

Definition 8.29. Let t be a term. The term t^M is defined as follows:

$$t^M = \begin{cases} t & \text{if } t \text{ is a variable} \\ f(t_n^M, \dots, t_1^M) & \text{if } t = f(t_1, \dots, t_n) \end{cases}$$

Moreover, if $\mathcal{R}$ is a TRS then $\mathcal{R}^M = \{\ell^M \to r^M \mid \ell \to r \in \mathcal{R}\}$.

We obviously have $s \to_\mathcal{R} t$ if and only if $s^M \to_{\mathcal{R}^M} t^M$. This gives the following result.

Theorem 8.30. *A TRS $\mathcal{R}$ is terminating if and only if $\mathcal{R}^M$ is terminating.* □

Example 8.31. Consider the one-rule ATRS $\mathcal{R}$

$$x\ (\mathsf{a}\ \mathsf{a}\ \mathsf{a}) \to \mathsf{a}\ (\mathsf{a}\ \mathsf{a})\ x$$

While $\mathcal{R}$ has a head variable in its left-hand side, the mirrored version $\mathcal{R}^M$

$$\mathsf{a}\ (\mathsf{a}\ \mathsf{a})\ x \to x\ (\mathsf{a}\ \mathsf{a}\ \mathsf{a})$$

is left head variable free. The transformed TRS $\mathcal{U}_\eta^+(\mathcal{R}^M)$

$$\mathsf{a}_2(\mathsf{a}_1(\mathsf{a}), x) \to x \star \mathsf{a}_2(\mathsf{a}, \mathsf{a}) \qquad \mathsf{a} \star x \to \mathsf{a}_1(x) \qquad \mathsf{a}_1(x) \star y \to \mathsf{a}_2(x, y)$$

is easily proved terminating with dependency pairs and a matrix interpretation of dimension one.

8.4. Innermost Termination

Before we prove that our transformation reflects innermost termination we show that it does not preserve innermost termination.

Example 8.32. Consider the ATRS $\mathcal{R}$

$$\mathsf{f}\ x \to \mathsf{f}\ x \qquad\qquad \mathsf{f} \to \mathsf{g}$$

In an innermost sequence the first rule is never applied and hence $\mathcal{R}$ is innermost terminating. The TRS $\mathcal{U}_\eta^+(\mathcal{R})$

$$\mathsf{f}_1(x) \to \mathsf{f}_1(x) \qquad \mathsf{f} \to \mathsf{g} \qquad \mathsf{f}_1(x) \to \mathsf{g} \star x \qquad \mathsf{f} \star x \to \mathsf{f}_1(x)$$

is not innermost terminating due to the rule $\mathsf{f}_1(x) \to \mathsf{f}_1(x)$.

The overlap between the rules of $\mathcal{R}$ in the above example is essential. This follows from a result of Gramlich [65] stating that innermost termination and termination coincide for locally confluent overlay systems. Hence for systems that satisfy the above conditions preservation of innermost termination can be recovered.

Theorem 8.33. *Let $\mathcal{R}$ be a locally confluent overlay ℓ-ATRS. If $\mathcal{R}$ is innermost terminating then $\mathcal{U}_\eta^+(\mathcal{R})$ is innermost terminating.*

Proof. Suppose $\mathcal{R}$ is innermost terminating. From [65] we know that $\mathcal{R}$ is terminating. Since uncurrying preserves termination (Theorem 8.28) also $\mathcal{U}_\eta^+(\mathcal{R})$ is terminating. In particular, $\mathcal{U}_\eta^+(\mathcal{R})$ is innermost terminating. $\square$

In the sequel we investigate if uncurrying reflects innermost termination. The next example shows that even in the innermost setting, η-saturation cannot be omitted. This is surprising since the η-rules are not innermost with respect to the original TRS but by uncurrying they become applicable at innermost redexes.

Example 8.34. The ATRS $\mathcal{R}$

$$\mathsf{h}\ x \to \mathsf{f}\ x \qquad\qquad \mathsf{f} \to \mathsf{h}$$

is not innermost terminating while $\mathcal{U}^+(\mathcal{R})$

$$\mathsf{h}_1(x) \to \mathsf{f}_1(x) \qquad \mathsf{f} \to \mathsf{h} \qquad \mathsf{h} \star x \to \mathsf{h}_1(x) \qquad \mathsf{f} \star x \to \mathsf{f}_1(x)$$

is (innermost) terminating. Note that $\mathcal{U}_\eta^+(\mathcal{R})$ is not innermost terminating because it also contains the rule $\mathsf{f}_1(x) \to \mathsf{h}_1(x)$.

The next example shows that $s \xrightarrow{\mathsf{i}}_{\mathcal{R}} t$ does not imply $s{\downarrow_{\mathcal{U}}} \xrightarrow{\mathsf{i}}{}^{+}_{\mathcal{U}_\eta^+(\mathcal{R})} t{\downarrow_{\mathcal{U}}}$. This does not contradict that uncurrying reflects innermost termination but shows that the proof of Theorem 8.17 cannot be adopted for the innermost case without further ado.

Example 8.35. Consider the ATRS $\mathcal{R}$

$$\mathsf{f} \to \mathsf{g} \qquad\qquad \mathsf{a} \to \mathsf{b} \qquad\qquad \mathsf{g}\ x \to \mathsf{h}$$

and the innermost step $s = \mathsf{f}\ \mathsf{a} \xrightarrow{\mathsf{i}}_{\mathcal{R}} \mathsf{g}\ \mathsf{a} = t$. We have $s{\downarrow_{\mathcal{U}}} = \mathsf{f} \star \mathsf{a}$ and $t{\downarrow_{\mathcal{U}}} = \mathsf{g}_1(\mathsf{a})$. In the TRS $\mathcal{U}_\eta^+(\mathcal{R})$

$$\mathsf{f} \to \mathsf{g} \qquad\qquad \mathsf{a} \to \mathsf{b} \qquad\qquad \mathsf{g}_1(x) \to \mathsf{h} \qquad\qquad \mathsf{g} \star x \to \mathsf{g}_1(x)$$

we have $s{\downarrow_{\mathcal{U}}} \xrightarrow{\mathsf{i}}_{\mathcal{U}_\eta^+(\mathcal{R})} \mathsf{g} \star \mathsf{a}$ but the step from $\mathsf{g} \star \mathsf{a}$ to $t{\downarrow_{\mathcal{U}}}$ is not innermost.

The problem in Example 8.35 is that uncurrying steps performed after the corresponding rewrite step need not be innermost. Moreover, not even an innermost variant of Lemma 8.22 holds as the next example demonstrates.

Example 8.36. Consider the ATRS $\mathcal{R}$ consisting of the rules

$$\mathsf{h} \to \mathsf{b} \qquad\qquad \mathsf{h}\ x \to \mathsf{c}$$

the term $t = x \star y$ and the substitution $\sigma = \{x \mapsto \mathsf{h}\}$. We have $t{\downarrow_{\mathcal{U}}}\sigma{\downarrow_{\mathcal{U}}} = \mathsf{h} \star y$ and $(t\sigma){\downarrow_{\mathcal{U}}} = \mathsf{h}_1(y)$. The TRS $\mathcal{U}_\eta^+(\mathcal{R})$ consists of the rules

$$\mathsf{h} \to \mathsf{b} \qquad\qquad \mathsf{h}_1(x) \to \mathsf{c} \qquad\qquad \mathsf{h}_1(x) \to \mathsf{b} \star x \qquad\qquad \mathsf{h} \star x \to \mathsf{h}_1(x)$$

Like in the previous example, the step from $t{\downarrow_{\mathcal{U}}}\sigma{\downarrow_{\mathcal{U}}}$ to $(t\sigma){\downarrow_{\mathcal{U}}}$ is not innermost.

The above problems can be solved if we consider terms that are not completely uncurried. In our proof we follow a lazy approach and postpone uncurrying as long as possible. We show that an innermost root step in an ATRS $\mathcal{R}$ can be mimicked by an innermost sequence in $\mathcal{U}_\eta^+(\mathcal{R})$ according to the following diagram (Lemma 8.39):

$$\begin{tikzcd}
\ell\sigma \arrow[rr, "\mathsf{i}\quad\epsilon", "\mathcal{R}"'] \arrow[d, "\mathcal{U}\downarrow *"'] & & r\sigma \arrow[d, dotted, "\mathcal{U}\downarrow *"'] \\
\cdot \arrow[r, dotted, "\mathsf{i}\quad *", "\mathcal{U}_\eta^+(\mathcal{R})"'] & \ell{\downarrow_{\mathcal{U}}}\sigma{\downarrow_{\mathcal{U}}} \arrow[r, dotted, "\mathsf{i}", "\mathcal{U}_\eta^+(\mathcal{R})"'] & r{\downarrow_{\mathcal{U}}}\sigma{\downarrow_{\mathcal{U}}}
\end{tikzcd}$$

To show this result we need that rewriting with $\to_{\mathcal{U}}^*$ preserves $\mathcal{R}$-normal forms (Lemma 8.37) and does not create innermost redexes (Lemma 8.38). By considering rightmost innermost rewriting (Lemma 8.41) we can then establish that uncurrying reflects innermost termination (Theorem 8.42).

Lemma 8.37. *Let $\mathcal{R}$ be an ℓ-ATRS. If s is a term over the signature of $\mathcal{R}$, $s \in NF(\mathcal{R})$, and $s \to_{\mathcal{U}}^* t$ then $t \in NF(\mathcal{R}_\eta{\downarrow}_{\mathcal{U}})$.*

Proof. From Lemma 8.27 we obtain $s{\downarrow}_{\mathcal{C}'} = t{\downarrow}_{\mathcal{C}'}$. Note that $s{\downarrow}_{\mathcal{C}'} = s$ because s is a term over the signature of $\mathcal{R}$. If $t \notin NF(\mathcal{R}_\eta{\downarrow}_{\mathcal{U}})$ then $t \to_{\mathcal{R}_\eta{\downarrow}_{\mathcal{U}}} u$ for some term u. Lemma 8.26 yields $t{\downarrow}_{\mathcal{C}'} \to_{\mathcal{R}_\eta} u{\downarrow}_{\mathcal{C}'}$ and Lemma 8.16 yields $s \to_{\mathcal{R}} u{\downarrow}_{\mathcal{C}'}$. Hence $s \notin NF(\mathcal{R})$, contradicting the assumption. □

The following lemma states that uncurrying steps followed by taking a proper subterm can be reordered into first taking a proper subterm and then perform uncurrying steps.

Lemma 8.38. *Let $\mathcal{R}$ be an ℓ-ATRS. If s is a term over the signature of $\mathcal{R}$ then $s \to_{\mathcal{U}}^* \cdot \rhd u$ implies $s \rhd \cdot \to_{\mathcal{U}}^* u$.*

Proof. Assume $s \to_{\mathcal{U}}^* t \rhd u$. We show that $s \rhd \cdot \to_{\mathcal{U}}^* u$ by induction on s. If s is a variable or a constant then there is nothing to show. So let $s = s_1 \star s_2$. We consider two cases.

- If the outermost $\star$ has not been uncurried then $t = t_1 \star t_2$ with $s_1 \to_{\mathcal{U}}^* t_1$ and $s_2 \to_{\mathcal{U}}^* t_2$. Without loss of generality assume that $t_1 \unrhd u$. If $t_1 = u$ then $s \rhd s_1 \to_{\mathcal{U}}^* t_1$. If $t_1 \rhd u$ then the induction hypothesis yields $s_1 \rhd \cdot \to_{\mathcal{U}}^* u$ and hence also $s \rhd \cdot \to_{\mathcal{U}}^* u$.
- If the outermost $\star$ has been uncurried in the sequence from s to t then the head symbol of s_1 cannot be a variable and $\mathrm{aa}(s_1) > 0$. Hence we may write $s_1 = f \star t_1 \star \cdots \star t_i$ and $t = f_{i+1}(t_1', \ldots, t_i', s_2')$ with $t_j \to_{\mathcal{U}}^* t_j'$ for all $1 \leqslant j \leqslant i$ and $s_2 \to_{\mathcal{U}}^* s_2'$. Clearly, $t_j' \unrhd u$ for some $1 \leqslant j \leqslant i$ or $s_2' \unrhd t$. In all cases the result follows with the same reasoning as in the first case. □

The next lemma states that innermost root steps in an ATRS can be simulated by a (non-empty) sequence of innermost steps in $\mathcal{U}_\eta^+(\mathcal{R})$. Note that $\xrightarrow{\mathsf{i}}_{\mathcal{U}_\eta^+(\mathcal{R})}$ means innermost reduction with respect to all rules in $\mathcal{U}_\eta^+(\mathcal{R})$.

Lemma 8.39. *Let $\mathcal{R}$ be an ℓ-ATRS. If w is a term over the signature of $\mathcal{R}$ then $s \;{}_{\mathcal{U}}^{*}{\leftarrow}\; w \xrightarrow{\mathsf{i}}{}_{\mathcal{R}}^{\epsilon} t$ implies $s \xrightarrow{\mathsf{i}}{}^{+}_{\mathcal{U}_\eta^+(\mathcal{R})} \cdot \;{}_{\mathcal{U}}^{*}{\leftarrow}\; t$.*

Proof. We prove that $s \xrightarrow{\mathsf{i}}{}^{+}_{\mathcal{U}_\eta^+(\mathcal{R})} r{\downarrow}_{\mathcal{U}}\sigma{\downarrow}_{\mathcal{U}} \;{}_{\mathcal{U}}^{*}{\leftarrow}\; r\sigma$ whenever $s \;{}_{\mathcal{U}}^{*}{\leftarrow}\; \ell\sigma \xrightarrow{\mathsf{i}}{}_{\mathcal{R}}^{\epsilon} r\sigma$ for some rewrite rule $\ell \to r$ in $\mathcal{R}$. By Lemma 8.22 and the confluence of $\mathcal{U}$,

$$s \xrightarrow{\mathsf{i}}{}_{\mathcal{U}}^{*} (\ell\sigma){\downarrow}_{\mathcal{U}} = \ell{\downarrow}_{\mathcal{U}}\sigma{\downarrow}_{\mathcal{U}} \to_{\mathcal{U}_\eta^+(\mathcal{R})} r{\downarrow}_{\mathcal{U}}\sigma{\downarrow}_{\mathcal{U}} \;{}_{\mathcal{U}}^{*}{\leftarrow}\; r\sigma.$$

It remains to show that the sequence $s \xrightarrow{\mathsf{i}}{}_{\mathcal{U}}^{*} (\ell\sigma){\downarrow}_{\mathcal{U}}$ and the step $\ell{\downarrow}_{\mathcal{U}}\sigma{\downarrow}_{\mathcal{U}} \to_{\mathcal{U}_\eta^+(\mathcal{R})} r{\downarrow}_{\mathcal{U}}\sigma{\downarrow}_{\mathcal{U}}$ are innermost with respect to $\mathcal{U}_\eta^+(\mathcal{R})$.

- For the former, let $s \xrightarrow{\mathsf{i}}{}^{*}_{\mathcal{U}} C[u] \xrightarrow{\mathsf{i}}_{\mathcal{U}} C[u'] \xrightarrow{\mathsf{i}}{}^{*}_{\mathcal{U}} (\ell\sigma){\downarrow_{\mathcal{U}}}$ with $u \xrightarrow{\mathsf{i}}{}^{\epsilon}_{\mathcal{U}} u'$ and let t be a proper subterm of u. Obviously $\ell\sigma \to^{*}_{\mathcal{U}} C[u] \rhd t$. According to Lemma 8.38, $\ell\sigma \rhd v \to^{*}_{\mathcal{U}} t$ for some term v. Since $\ell\sigma \xrightarrow{\mathsf{i}}{}^{\epsilon}_{\mathcal{R}} r\sigma$, the term v is a normal form of $\mathcal{R}$. Hence $t \in \mathit{NF}(\mathcal{R}_{\eta}{\downarrow_{\mathcal{U}}})$ by Lemma 8.37. Since $u \xrightarrow{\mathsf{i}}{}^{\epsilon}_{\mathcal{U}} u'$, t is also a normal form of $\mathcal{U}$. Hence $t \in \mathit{NF}(\mathcal{U}^{+}_{\eta}(\mathcal{R}))$ as desired.
- For the latter, let t be a proper subterm of $(\ell\sigma){\downarrow_{\mathcal{U}}}$. According to Lemma 8.38, $\ell\sigma \rhd u \to^{*}_{\mathcal{U}} t$. The term u is a normal form of $\mathcal{R}$. Hence $t \in \mathit{NF}(\mathcal{R}_{\eta}{\downarrow_{\mathcal{U}}})$ by Lemma 8.37. Obviously, $t \in \mathit{NF}(\mathcal{U})$ and thus also $t \in \mathit{NF}(\mathcal{U}^{+}_{\eta}(\mathcal{R}))$. □

The next example shows that it is not sound to replace $\xrightarrow{\mathsf{i}}{}^{\epsilon}_{\mathcal{R}}$ by $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ in Lemma 8.39.

Example 8.40. Consider the ATRS $\mathcal{R}$

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

Then $\mathsf{f}_1(\mathsf{a})\ {}_{\mathcal{U}}^{*}\!\!\leftarrow \mathsf{f} \star \mathsf{a} \xrightarrow{\mathsf{i}}_{\mathcal{R}} \mathsf{g} \star \mathsf{a}$ but $\mathsf{f}_1(\mathsf{a}) \xrightarrow{\mathsf{i}}{}^{+}_{\mathcal{U}^{+}_{\eta}(\mathcal{R})} \cdot\ {}_{\mathcal{U}}^{*}\!\!\leftarrow \mathsf{g} \star \mathsf{a}$ does not hold. To see the latter, consider the two reducts $\mathsf{g}_1(\mathsf{a})$ and $\mathsf{g} \star \mathsf{a}$ of $\mathsf{g} \star \mathsf{a}$ with respect to $\to^{*}_{\mathcal{U}}$. We have neither $\mathsf{f}_1(\mathsf{a}) \xrightarrow{\mathsf{i}}{}^{+}_{\mathcal{U}^{+}_{\eta}(\mathcal{R})} \mathsf{g}_1(\mathsf{a})$ nor $\mathsf{f}_1(\mathsf{a}) \xrightarrow{\mathsf{i}}{}^{+}_{\mathcal{U}^{+}_{\eta}(\mathcal{R})} \mathsf{g} \star \mathsf{a}$.

In order to extend Lemma 8.38 to non-root positions, we have to use *rightmost* innermost rewriting. This avoids the situation in the above example where parallel redexes become nested by uncurrying.

Lemma 8.41. *Let $\mathcal{R}$ be an ℓ-ATRS and t a term over the signature of $\mathcal{R}$. If $s\ {}_{\mathcal{U}}^{*}\!\!\leftarrow t \xrightarrow{\mathsf{ri}}_{\mathcal{R}} u$ then $s \xrightarrow{\mathsf{i}}{}^{+}_{\mathcal{U}^{+}_{\eta}(\mathcal{R})} \cdot\ {}_{\mathcal{U}}^{*}\!\!\leftarrow u$.*

Proof. Let $s\ {}_{\mathcal{U}}^{*}\!\!\leftarrow t = C[\ell\sigma] \xrightarrow{\mathsf{ri}}_{\mathcal{R}} C[r\sigma] = u$ with $\ell\sigma \xrightarrow{\mathsf{i}}{}^{\epsilon}_{\mathcal{R}} r\sigma$. We use induction on the context C. If $C = \Box$ then $s\ {}_{\mathcal{U}}^{*}\!\!\leftarrow t \xrightarrow{\mathsf{i}}{}^{\epsilon}_{\mathcal{R}} u$. Lemma 8.39 yields

$$s \xrightarrow{\mathsf{i}}{}^{+}_{\mathcal{U}^{+}_{\eta}(\mathcal{R})} \cdot\ {}_{\mathcal{U}}^{*}\!\!\leftarrow u.$$

For the induction step we consider two cases.

- Suppose $C = \Box \star s_1 \star \cdots \star s_n$ and $n > 0$. Since $\mathcal{R}$ is left head variable free, $\mathrm{aa}(\ell)$ is defined. If $\mathrm{aa}(\ell) = 0$ then

$$s = t' \star s'_1 \star \cdots \star s'_n\ {}_{\mathcal{U}}^{*}\!\!\leftarrow \ell\sigma \star s_1 \star \cdots \star s_n \xrightarrow{\mathsf{i}}_{\mathcal{R}} r\sigma \star s_1 \star \cdots \star s_n$$

  with $t'\ {}_{\mathcal{U}}^{*}\!\!\leftarrow \ell\sigma$ and $s'_j\ {}_{\mathcal{U}}^{*}\!\!\leftarrow s_j$ for $1 \leqslant j \leqslant n$. The claim follows using Lemma 8.39 and the fact that innermost rewriting is closed under contexts. If $\mathrm{aa}(\ell) > 0$ then the head symbol of ℓ cannot be a variable. We have to consider two cases. In the case where the leftmost $\star$ symbol in C has not been uncurried we proceed as when $\mathrm{aa}(\ell) = 0$. If the leftmost $\star$ symbol of C

has been uncurried, we reason as follows. We may write $\ell\sigma = f \star u_1 \star \cdots \star u_k$ where $k < \mathrm{aa}(f)$. We have $t = f \star u_1 \star \cdots \star u_k \star s_1 \star \cdots \star s_n$ and $u = r\sigma \star s_1 \star \cdots \star s_n$. There exists an i with $1 \leqslant i \leqslant \min\{\mathrm{aa}(f), k+n\}$ such that

$$s = f_i(u'_1, \dots, u'_k, s'_1, \dots, s'_{i-k}) \star s'_{i-k+1} \star \cdots \star s'_n$$

with $u'_j \;{}_{\mathcal{U}}^{*}\!\!\leftarrow u_j$ for $1 \leqslant j \leqslant k$ and $s'_j \;{}_{\mathcal{U}}^{*}\!\!\leftarrow s_j$ for $1 \leqslant j \leqslant n$. Because of rightmost innermost evaluation, the terms $u_1, \dots, u_k, s_1, \dots, s_n$ are normal forms of $\mathcal{R}$. According to Lemma 8.37 the terms $u'_1, \dots, u'_k, s'_1, \dots, s'_n$ are normal forms of $\mathcal{R}_\eta{\downarrow_{\mathcal{U}}}$. Since $i - k \leqslant \mathrm{aa}(\ell)$, $\mathcal{R}_\eta$ contains the rule

$$\ell \star x_1 \star \cdots \star x_{i-k} \to r \star x_1 \star \cdots \star x_{i-k}$$

where $x_1, \dots, x_{i-k}$ are pairwise distinct variables not occurring in $\ell \to r$. Hence the substitution $\tau = \sigma \cup \{x_1 \mapsto s_1, \dots, x_{i-k} \mapsto s_{i-k}\}$ is well-defined. We obtain

$$\begin{array}{rl} s & \xrightarrow{\mathsf{i}}{}^{*}_{\mathcal{U}_\eta^+(\mathcal{R})} \; f_i(u_1{\downarrow_{\mathcal{U}}}, \dots, u_k{\downarrow_{\mathcal{U}}}, s_1{\downarrow_{\mathcal{U}}}, \dots, s_{i-k}{\downarrow_{\mathcal{U}}}) \star s'_{i-k+1} \star \cdots \star s'_n \\ & \xrightarrow{\mathsf{i}}_{\mathcal{U}_\eta^+(\mathcal{R})} \; (r \star x_1 \star \cdots \star x_{i-k}){\downarrow_{\mathcal{U}}}\tau{\downarrow_{\mathcal{U}}} \star s'_{i-k+1} \star \cdots \star s'_n \\ & {}_{\mathcal{U}}^{*}\!\!\leftarrow \; (r \star x_1 \star \cdots \star x_{i-k})\tau \star s_{i-k+1} \star \cdots \star s_n \\ & = \; r\sigma \star s_1 \star \cdots \star s_n \; = \; u \end{array}$$

where we use the confluence of $\mathcal{U}$ in the first sequence.

- In the second case we have $C = s_1 \star C'$. Clearly $C'[\ell\sigma] \xrightarrow{\mathsf{ri}}_{\mathcal{R}} C'[r\sigma]$. If $\mathrm{aa}(s_1) \leqslant 0$ or if $\mathrm{aa}(s_1)$ is undefined or if $\mathrm{aa}(s_1) > 0$ and the outermost $\star$ has not been uncurried in the sequence from t to s then

$$s = s'_1 \star s' \;{}_{\mathcal{U}}^{*}\!\!\leftarrow s_1 \star C'[\ell\sigma] \xrightarrow{\mathsf{ri}}_{\mathcal{R}} s_1 \star C'[r\sigma] = u$$

with $s'_1 \;{}_{\mathcal{U}}^{*}\!\!\leftarrow s_1$ and $s' \;{}_{\mathcal{U}}^{*}\!\!\leftarrow C'[\ell\sigma]$. If $\mathrm{aa}(s_1) > 0$ and the outermost $\star$ has been uncurried in the sequence from t to s then we may write $s_1 = f \star u_1 \star \cdots \star u_k$ where $k < \mathrm{aa}(f)$. We have $s = f_{k+1}(u'_1, \dots, u'_k, s')$ for some term s' with $s' \;{}_{\mathcal{U}}^{*}\!\!\leftarrow C'[\ell\sigma]$ and $u'_i \;{}_{\mathcal{U}}^{*}\!\!\leftarrow u_i$ for $1 \leqslant i \leqslant k$. In both cases the induction hypothesis yields

$$s' \xrightarrow{\mathsf{i}}{}^{+}_{\mathcal{U}_\eta^+(\mathcal{R})} \cdot \;{}_{\mathcal{U}}^{*}\!\!\leftarrow C'[r\sigma]$$

and, since innermost rewriting is closed under contexts, we obtain

$$s \xrightarrow{\mathsf{i}}{}^{+}_{\mathcal{U}_\eta^+(\mathcal{R})} \cdot \;{}_{\mathcal{U}}^{*}\!\!\leftarrow u$$

as desired. □

We are now ready for the result that uncurrying reflects innermost termination.

Theorem 8.42. *An ℓ-ATRS $\mathcal{R}$ is innermost terminating if $\mathcal{U}_\eta^+(\mathcal{R})$ is innermost terminating.*

Proof. For a proof by contradiction assume an infinite sequence

$$t_1 \xrightarrow{\mathsf{ri}}_{\mathcal{R}} t_2 \xrightarrow{\mathsf{ri}}_{\mathcal{R}} t_3 \xrightarrow{\mathsf{ri}}_{\mathcal{R}} \cdots$$

Using Lemma 8.41 this sequence can be transformed into

$$t_1{\downarrow_{\mathcal{U}}} \xrightarrow{\mathsf{i}}{}^{+}_{\mathcal{U}_\eta^+(\mathcal{R})} t_2' \xrightarrow{\mathsf{i}}{}^{+}_{\mathcal{U}_\eta^+(\mathcal{R})} t_3' \xrightarrow{\mathsf{i}}{}^{+}_{\mathcal{U}_\eta^+(\mathcal{R})} \cdots$$

for terms t_2', t_3', $\ldots$ such that $t_i \to_{\mathcal{U}}^* t_i'$ for $i \geqslant 2$. The proof concludes by the fact that innermost termination is equivalent to rightmost innermost termination, a result due to Krishna Rao [109]. □

8.5. Derivational Complexity

Next we investigate how the uncurrying transformation affects derivational complexity for full (Section 8.5.1) and innermost rewriting (Section 8.5.2).

8.5.1. Full Rewriting

The next theorem explains why uncurrying can be used as a preprocessor for proving upper bounds on the derivational complexity.

Theorem 8.43. *If $\mathcal{R}$ is a terminating ℓ-ATRS then $\mathrm{dc}_{\mathcal{R}}(n) \in \mathcal{O}(\mathrm{dc}_{\mathcal{U}_\eta^+(\mathcal{R})}(n))$.*

Proof. Consider an arbitrary maximal rewrite sequence in $\mathcal{R}$ starting from t_0

$$t_0 \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t_m$$

Using the proof of Theorem 8.17, we can transform the sequence into

$$t_0{\downarrow_{\mathcal{U}}} \to^+_{\mathcal{U}_\eta^+(\mathcal{R})} t_1{\downarrow_{\mathcal{U}}} \to^+_{\mathcal{U}_\eta^+(\mathcal{R})} t_2{\downarrow_{\mathcal{U}}} \to^+_{\mathcal{U}_\eta^+(\mathcal{R})} \cdots \to^+_{\mathcal{U}_\eta^+(\mathcal{R})} t_m{\downarrow_{\mathcal{U}}}$$

Moreover, $t_0 \to^*_{\mathcal{U}_\eta^+(\mathcal{R})} t_0{\downarrow_{\mathcal{U}}}$ holds. Therefore, $\mathrm{dh}(t_0, \to_{\mathcal{R}}) \leqslant \mathrm{dh}(t_0, \to_{\mathcal{U}_\eta^+(\mathcal{R})})$. Hence $\mathrm{dc}_{\mathcal{R}}(n) \leqslant \mathrm{dc}_{\mathcal{U}_\eta^+(\mathcal{R})}(n)$ holds for all $n \in \mathbb{N}$, showing the result. □

Next we show that uncurrying preserves polynomial complexity. Since any duplicating TRS has at least exponential derivational complexity (cf. [75]), we only deal with non-duplicating TRSs. Furthermore we ignore pathological systems that yield constant derivational complexity (note that any non-empty ATRS admits at least derivations linear in the size of the starting term).

A TRS $\mathcal{R}$ is called *length-reducing* if $\mathcal{R}$ is non-duplicating and $|\ell| > |r|$ for all rules $\ell \to r \in \mathcal{R}$. The following lemma is an easy consequence of [75, Theorem 23]. Below, $\to_{\mathcal{R}/\mathcal{S}}$ denotes $\to_{\mathcal{S}}^{*} \cdot \to_{\mathcal{R}} \cdot \to_{\mathcal{S}}^{*}$.

Lemma 8.44. *Let $\mathcal{R}$ be a non-empty and non-duplicating TRS over a signature containing a function symbol of arity at least two. If a TRS $\mathcal{S}$ is length-reducing, $\mathrm{dc}_{\mathcal{R}\cup\mathcal{S}}(n) \in \mathcal{O}(\mathrm{dc}_{\mathcal{R}/\mathcal{S}}(n))$ holds whenever $\mathcal{R} \cup \mathcal{S}$ is terminating.* □

Note that the above lemma does not hold if the TRS $\mathcal{R}$ is empty.

Theorem 8.45. *Let $\mathcal{R}$ be a non-empty, non-duplicating, and terminating ℓ-ATRS. If $\mathrm{dc}_{\mathcal{R}}(n)$ is in $\mathcal{O}(n^k)$ then $\mathrm{dc}_{\mathcal{R}_\eta{\downarrow_{\mathcal{U}}}/\mathcal{U}}(n)$ and $\mathrm{dc}_{\mathcal{U}_\eta^+(\mathcal{R})}(n)$ are in $\mathcal{O}(n^k)$.*

Proof. Suppose that $\mathrm{dc}_{\mathcal{R}}(n)$ is in $\mathcal{O}(n^k)$. First consider a maximal rewrite sequence of $\to_{\mathcal{R}_\eta{\downarrow_{\mathcal{U}}}/\mathcal{U}}$ starting from a term t_0:

$$t_0 \to_{\mathcal{R}_\eta{\downarrow_{\mathcal{U}}}/\mathcal{U}} t_1 \to_{\mathcal{R}_\eta{\downarrow_{\mathcal{U}}}/\mathcal{U}} \cdots \to_{\mathcal{R}_\eta{\downarrow_{\mathcal{U}}}/\mathcal{U}} t_m$$

By Lemmata 8.27 and 8.16 we obtain the sequence

$$t_0{\downarrow_{\mathcal{C}'}} \to_{\mathcal{R}} t_1{\downarrow_{\mathcal{C}'}} \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t_m{\downarrow_{\mathcal{C}'}}.$$

Thus, $\mathrm{dh}(t_0, \to_{\mathcal{R}_\eta{\downarrow_{\mathcal{U}}}/\mathcal{U}}) \leqslant \mathrm{dh}(t_0{\downarrow_{\mathcal{C}'}}, \to_{\mathcal{R}})$. Because $|t_0{\downarrow_{\mathcal{C}'}}| \leqslant 2|t_0|$ holds, we obtain $\mathrm{dc}_{\mathcal{R}_\eta{\downarrow_{\mathcal{U}}}/\mathcal{U}}(n) \leqslant \mathrm{dc}_{\mathcal{R}}(2n)$. From the assumption the right-hand side is in $\mathcal{O}(n^k)$, Therefore, $\mathrm{dc}_{\mathcal{R}_\eta{\downarrow_{\mathcal{U}}}/\mathcal{U}}(n)$ is in $\mathcal{O}(n^k)$. Because $\mathcal{U}$ is length-reducing, $\mathrm{dc}_{\mathcal{U}_\eta^+(\mathcal{R})}(n)$ is also in $\mathcal{O}(n^k)$, by Lemma 8.44. □

In practice it is recommendable to investigate $\mathrm{dc}_{\mathcal{R}_\eta{\downarrow_{\mathcal{U}}}/\mathcal{U}}(n)$ instead of $\mathrm{dc}_{\mathcal{U}_\eta^+(\mathcal{R})}(n)$, see [206]. The next example shows that uncurrying might be useful to enable criteria for polynomial complexity.

Example 8.46. Consider the ATRS $\mathcal{R}$ consisting of the rules

$$\mathsf{add}\ x\ \mathsf{0} \to x \qquad\qquad \mathsf{add}\ x\ (\mathsf{s}\ y) \to \mathsf{s}\ (\mathsf{add}\ x\ y)$$

The system $\mathcal{U}_\eta^+(\mathcal{R})$ consists of the rules

$$\mathsf{add}_2(x, \mathsf{0}) \to x \qquad\qquad \mathsf{add}_2(x, \mathsf{s}_1(y)) \to \mathsf{s}_1(\mathsf{add}_2(x, y))$$
$$\mathsf{add}_1(x) \star y \to \mathsf{add}_2(x, y) \qquad \mathsf{add} \star x \to \mathsf{add}_1(x) \qquad \mathsf{s} \star x \to \mathsf{s}_1(x)$$

It is easy to see that the following TMI $\mathcal{M}$ of dimension 2 below orients all rules in $\mathcal{U}_\eta^+(\mathcal{R})$ strictly, inducing a quadratic bound on the derivational complexity of $\mathcal{U}_\eta^+(\mathcal{R})$ (according to Theorem 8.1):

$$\begin{aligned}
\mathsf{add}_{1\mathcal{M}}(x) = \mathsf{s}_{1\mathcal{M}}(x) &= \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} x + \begin{pmatrix}0\\1\end{pmatrix}\\
\mathsf{add}_{2\mathcal{M}}(x,y) = \star_{\mathcal{M}}(x,y) &= \begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix} x + \begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix} y\\
\mathsf{s}_{\mathcal{M}} = \mathsf{0}_{\mathcal{M}} = \mathsf{add}_{\mathcal{M}} &= \begin{pmatrix}0\\1\end{pmatrix}
\end{aligned}$$

Theorem 8.43 then establishes a quadratic bound on the derivational complexity of $\mathcal{R}$. In contrast to $\mathcal{U}_\eta^+(\mathcal{R})$, the ATRS $\mathcal{R}$ itself does not admit such an interpretation of dimension 2. To see this, we encoded the required condition as a satisfaction problem in non-linear arithmetic over the integers. MiniSmt [208] can prove this problem unsatisfiable by simplifying it into a trivially unsatisfiable constraint. Details can be inferred from the website mentioned in Footnote 4 on page 224.

8.5.2. Innermost Rewriting

Next we consider innermost derivational complexity. Let $\mathcal{R}$ be an innermost terminating TRS. From a result by Krishna Rao [109, Section 5.1] which has been generalized by van Oostrom [145, Theorems 2 and 3] we infer that

$$\mathrm{dh}(t, \xrightarrow{\mathsf{i}}_{\mathcal{R}}) = \mathrm{dh}(t, \xrightarrow{\mathsf{ri}}_{\mathcal{R}})$$

holds for all terms t.

Theorem 8.47. *Let $\mathcal{R}$ be an innermost terminating ℓ-ATRS. Then* $\mathrm{idc}_{\mathcal{R}}(n) \in \mathcal{O}(\mathrm{idc}_{\mathcal{U}_\eta^+(\mathcal{R})}(n))$.

Proof. Consider a maximal rightmost innermost rewrite sequence

$$t_0 \xrightarrow{\mathsf{ri}}_{\mathcal{R}} t_1 \xrightarrow{\mathsf{ri}}_{\mathcal{R}} t_2 \xrightarrow{\mathsf{ri}}_{\mathcal{R}} \cdots \xrightarrow{\mathsf{ri}}_{\mathcal{R}} t_m.$$

Using Lemma 8.41 the sequence can be transformed into

$$t_0 \xrightarrow{\mathsf{i}}{}^{+}_{\mathcal{U}_\eta^+(\mathcal{R})} t_1' \xrightarrow{\mathsf{i}}{}^{+}_{\mathcal{U}_\eta^+(\mathcal{R})} t_2' \xrightarrow{\mathsf{i}}{}^{+}_{\mathcal{U}_\eta^+(\mathcal{R})} \cdots \xrightarrow{\mathsf{i}}{}^{+}_{\mathcal{U}_\eta^+(\mathcal{R})} t_m'$$

for terms $t_1', t_2', \ldots, t_m'$ such that $t_i \to_{\mathcal{U}}^{*} t_i'$ for all $1 \leqslant i \leqslant m$. Thus,

$$\mathrm{dh}(t_0, \xrightarrow{\mathsf{i}}_{\mathcal{R}}) = \mathrm{dh}(t_0, \xrightarrow{\mathsf{ri}}_{\mathcal{R}}) \leqslant \mathrm{dh}(t_0, \xrightarrow{\mathsf{i}}_{\mathcal{U}_\eta^+(\mathcal{R})}).$$

Hence, we conclude $\mathrm{idc}_{\mathcal{R}}(n) \in \mathcal{O}(\mathrm{idc}_{\mathcal{U}_\eta^+(\mathcal{R})}(n))$. $\square$

As Example 8.32 on page 205 showed, uncurrying does not preserve innermost termination. Similarly, it does not preserve innermost polynomial complexity even if the original ATRS has linear derivational complexity.

Example 8.48. Consider the non-duplicating ATRS $\mathcal{R}$

$$\mathsf{f} \to \mathsf{s} \qquad\qquad \mathsf{f}\ (\mathsf{s}\ x) \to \mathsf{s}\ (\mathsf{s}\ (\mathsf{f}\ x))$$

Since the right rule is never used in innermost rewriting, $\mathrm{idc}_{\mathcal{R}}(n) \in \mathcal{O}(n)$ is shown by easy induction on n. We show that the innermost derivational complexity of $\mathcal{U}^+_\eta(\mathcal{R})$ is at least exponential. The TRS $\mathcal{U}^+_\eta(\mathcal{R})$ consists of the five rules:

$$\begin{array}{lll} \mathsf{f} \to \mathsf{s} & \mathsf{f}_1(\mathsf{s}_1(x)) \to \mathsf{s}_1(\mathsf{s}_1(\mathsf{f}_1(x))) & \mathsf{f} \star x \to \mathsf{f}_1(x) \\ \mathsf{f}_1(x) \to \mathsf{s}_1(x) & & \mathsf{s} \star x \to \mathsf{s}_1(x) \end{array}$$

One can verify

$$\mathrm{dh}(\overbrace{\mathsf{f}_1(\cdots(\mathsf{f}_1}^{n}(\mathsf{s}_1(x)))), \xrightarrow{\mathsf{i}}_{\mathcal{U}^+_\eta(\mathcal{R})}) \geqslant 2^n$$

for all $n \geqslant 1$. Hence $\mathrm{idc}_{\mathcal{U}^+_\eta(\mathcal{R})}(n+3) \geqslant 2^n$ for all $n \geqslant 0$.

Similar to Theorem 8.33, the result can be recovered for locally confluent overlay systems. In the sequel a substitution σ is called normalized (for a TRS $\mathcal{R}$) if $x\sigma \in \mathit{NF}(\mathcal{R})$ for all $x \in \mathcal{V}$. The next lemmata state useful properties that prepare for the proof. The first of these states a trivial diamond-like property of innermost rewriting. It is used in the proofs of the next lemmata to rearrange innermost rewrite sequences such that the number of steps is preserved.

Lemma 8.49. *Let $\mathcal{R}$ be a TRS. If $s \xrightarrow{\mathsf{i}} t$ and $s \xrightarrow{\mathsf{i}} u$ by rewriting innermost redexes at parallel positions then $t \xrightarrow{\mathsf{i}} v$ and $u \xrightarrow{\mathsf{i}} v$ for some term v.* □

Lemma 8.50. *If $t\sigma \xrightarrow{\mathsf{i}}{}^n u \in \mathit{NF}(\mathcal{R})$ then $t\sigma \xrightarrow{\mathsf{i}}{}^{n_1} t\tau \xrightarrow{\mathsf{i}}{}^{n_2} u$ for some normalized substitution τ and $n_1, n_2 \in \mathbb{N}$ with $n_1 + n_2 = n$.*

Proof. We use induction on n. Since the base case is trivial, we consider the inductive step. Suppose $t\sigma \xrightarrow{\mathsf{i}}{}^n u \in \mathit{NF}(\mathcal{R})$. Without loss of generality we assume $\mathcal{D}\mathsf{om}(\sigma) \subseteq \mathcal{V}\mathsf{ar}(t)$. We proceed by a case distinction. If σ is normalized then the claim follows with $n_1 = 0$, $n_2 = n$, and $\tau = \sigma$. In the other case, by Lemma 8.49 we can reorder the sequence such that the first rewrite step takes place in the substitution part. Therefore, there exists a substitution σ' with $x\sigma \xrightarrow{\mathsf{i}} x\sigma'$ for some $x \in \mathcal{D}\mathsf{om}(\sigma)$ and $y\sigma = y\sigma'$ for all $y \in \mathcal{D}\mathsf{om}(\sigma) \setminus \{x\}$. Writing k for the number of occurrences of x in t, we have $t\sigma \xrightarrow{\mathsf{i}}{}^k t\sigma' \xrightarrow{\mathsf{i}}{}^{n-k} u$. Since $k \geqslant 1$, the induction hypothesis applied to $t\sigma' \xrightarrow{\mathsf{i}}{}^{n-k} u$ yields $t\sigma' \xrightarrow{\mathsf{i}}{}^{m_1} t\tau \xrightarrow{\mathsf{i}}{}^{m_2} u$

with $m_1 + m_2 = n - k$ and normalized τ. Combining this with $t\sigma \xrightarrow{\mathsf{i}}{}^{k} t\sigma'$, $t\sigma \xrightarrow{\mathsf{i}}{}^{k+m_1} t\tau \xrightarrow{\mathsf{i}}{}^{m_2} u$ is obtained. By taking $n_1 = k + m_1$ and $n_2 = m_2$, we obtain $n_1 + n_2 = k + m_1 + m_2 = k + (n - k) = n$ which proves the claim. □

Lemma 8.51. *Let $\mathcal{R}$ be a non-duplicating overlay system. If $t \rightarrow^m u \in \mathit{NF}(\mathcal{R})$ then $t \xrightarrow{\mathsf{i}}{}^{n} u$ for some $n \geqslant m$.*

Proof. We use induction on m. Since the base case is trivial, we consider the inductive step. Suppose

$$C[\ell\sigma] \rightarrow C[r\sigma] \rightarrow^m u \in \mathit{NF}(\mathcal{R})$$

with $\ell \rightarrow r \in \mathcal{R}$. The induction hypothesis yields $C[r\sigma] \xrightarrow{\mathsf{i}}{}^{n} u$ for some $n \geqslant m$. By Lemma 8.49 this sequence can be written as $C[r\sigma] \xrightarrow{\mathsf{i}}{}^{n_1} C[u'] \xrightarrow{\mathsf{i}}{}^{n_2} u$ with $n = n_1 + n_2$ and $u' \in \mathit{NF}(\mathcal{R})$. From Lemma 8.50 we obtain a normalized τ and $m_1, m_2 \in \mathbb{N}$ with $m_1 + m_2 = n_1$ such that $r\sigma \xrightarrow{\mathsf{i}}{}^{m_1} r\tau \xrightarrow{\mathsf{i}}{}^{m_2} u'$. Because innermost rewriting is closed under contexts, $C[r\sigma] \xrightarrow{\mathsf{i}}{}^{m_1} C[r\tau] \xrightarrow{\mathsf{i}}{}^{m_2+n_2} u$. Since $\mathcal{R}$ is non-duplicating, $C[\ell\sigma] \xrightarrow{\mathsf{i}}{}^{m_0} C[\ell\tau]$ for some $m_0 \geqslant m_1$. Because $\mathcal{R}$ is an overlay system and τ is normalized $\ell\tau \xrightarrow{\mathsf{i}} r\tau$. Hence

$$C[\ell\sigma] \xrightarrow{\mathsf{i}}{}^{m_0} C[\ell\tau] \xrightarrow{\mathsf{i}} C[r\tau] \xrightarrow{\mathsf{i}}{}^{m_2+n_2} u.$$

Here $m_0 + 1 + m_2 + n_2 \geqslant m_1 + 1 + m_2 + n_2 = n_1 + 1 + n_2 = n + 1 \geqslant m + 1$. □

By the above lemma the next theorem is obtained.

Theorem 8.52. *If $\mathcal{R}$ is a non-duplicating and terminating overlay system then* $\mathrm{idc}_{\mathcal{R}}(n) = \mathrm{dc}_{\mathcal{R}}(n)$.

Proof. Since $\mathcal{R}$ is terminating, Lemma 8.51 yields $\mathrm{dh}(t, \xrightarrow{\mathsf{i}}_{\mathcal{R}}) \geqslant \mathrm{dh}(t, \rightarrow_{\mathcal{R}})$. Combining this with the obvious $\mathrm{dh}(t, \xrightarrow{\mathsf{i}}_{\mathcal{R}}) \leqslant \mathrm{dh}(t, \rightarrow_{\mathcal{R}})$ concludes the proof. □

Using Gramlich's [65] result on the equivalence of termination and innermost termination for locally confluent overlay systems we obtain the following corollary from Theorems 8.45 and 8.52.

Corollary 8.53. *Let $\mathcal{R}$ be a non-duplicating, innermost terminating, and locally confluent overlay ℓ-ATRS. If* $\mathrm{idc}_{\mathcal{R}}(n)$ *is in* $\mathcal{O}(n^k)$ *then* $\mathrm{idc}_{\mathcal{U}_\eta^+(\mathcal{R})}(n)$ *is in* $\mathcal{O}(n^k)$. □

8.6. Uncurrying with Dependency Pairs

In this section we incorporate the uncurrying transformation into the dependency pair framework [13, 58, 60, 73, 174].

In the sequel we present two DP processors that uncurry applicative DP problems, which are DP problems over signatures containing constants and two application symbols: $\star$ and $\star^\sharp$. Properties for full termination of these processors are studied in Section 8.6.1 while innermost termination is considered in Section 8.6.2.

First we define a suitable set of uncurrying rules for DP problems. Let $(\mathcal{P}, \mathcal{R})$ be an applicative DP problem. Here the applicative arities of function symbols are computed with respect to $\star$ and $\mathcal{P} \cup \mathcal{R}$. By $\mathcal{U}(\mathcal{P}, \mathcal{R})$ we denote the uncurrying rules derived from $\mathcal{U}(\mathcal{P}) \cup \mathcal{U}(\mathcal{R})$. In this section we assume that $\mathcal{F}$ is $\mathcal{F}\mathsf{un}(\mathcal{P} \cup \mathcal{R})$, write $\mathcal{U}(\mathcal{F})$ for $\mathcal{U}(\mathcal{P}, \mathcal{R})$ and let $\mathcal{U}_\eta^+(\mathcal{R}, \mathcal{F})$ denote $\mathcal{R}_\eta{\downarrow_{\mathcal{U}(\mathcal{F})}} \cup \mathcal{U}(\mathcal{F})$. If no confusion can arise, $\mathcal{F}$ is dropped in $\mathcal{U}(\mathcal{F})$ and $\mathcal{U}_\eta^+(\mathcal{R}, \mathcal{F})$. An applicative DP problem $(\mathcal{P}, \mathcal{R})$ is said to be ℓ-applicative if $\mathcal{P} \cup \mathcal{R}$ is left head variable free.

Definition 8.54. Let $(\mathcal{P}, \mathcal{R})$ be a DP problem. The DP processor $\mathcal{U}_1$ is defined as

$$(\mathcal{P}, \mathcal{R}) \mapsto \begin{cases} \{(\mathcal{P}{\downarrow_{\mathcal{U}(\mathcal{F})}}, \mathcal{U}_\eta^+(\mathcal{R}, \mathcal{F}))\} & \text{if } (\mathcal{P}, \mathcal{R}) \text{ is } \ell\text{-applicative} \\ \{(\mathcal{P}, \mathcal{R})\} & \text{otherwise} \end{cases}$$

where $\mathcal{F} = \mathcal{F}\mathsf{un}(\mathcal{P} \cup \mathcal{R})$.

Example 8.55. Consider the ℓ-applicative (note that it is left head variable free) DP problem $(\{x \ ^\sharp \ (\mathsf{a}\ \mathsf{a}) \to (\mathsf{a}\ \mathsf{a}\ \mathsf{a}) \ ^\sharp \ x\}, \varnothing)$. Processor $\mathcal{U}_1$ transforms it into the problem $(\{x \star^\sharp (\mathsf{a}_1(\mathsf{a})) \to \mathsf{a}_2(\mathsf{a}, \mathsf{a}) \star^\sharp x\}, \{\mathsf{a} \star x \to \mathsf{a}_1(x), \mathsf{a}_1(x) \star y \to \mathsf{a}_2(x, y)\})$ because the applicative arity of a is two. The latter DP problem is easily shown finite by a matrix interpretation of dimension one counting the symbols $\star$ and a_1.

A drawback of $\mathcal{U}_1$ is that dependency pair symbols are excluded from the uncurrying process. Typically, all pairs in $\mathcal{P}$ have the same root symbol $\star^\sharp$. The next example shows that uncurrying root symbols of $\mathcal{P}$ can be beneficial.

Example 8.56. Consider the ATRS consisting of the single rule $\mathsf{a}\ x\ \mathsf{a} \to \mathsf{a}\ (\mathsf{a}\ \mathsf{a})\ x$. After processing the only SCC in the dependency graph with $\mathcal{U}_1$, the rewrite rule $\mathsf{a}_1(x) \star^\sharp \mathsf{a} \to \mathsf{a}_1(\mathsf{a}_1(\mathsf{a})) \star^\sharp x$ must be oriented. This cannot be done with a matrix interpretation of dimension one nor with a reduction pair based on any other simplification order. If we transform the rule into $\mathsf{a}_2^\sharp(x, \mathsf{a}) \to \mathsf{a}_2^\sharp(\mathsf{a}_1(\mathsf{a}), x)$ this becomes trivial.

To this end we introduce a simple variant of *freezing* [196].

Definition 8.57. A *simple freeze* is a partial mapping $❄$ that assigns to a function symbol of arity $n > 0$ an argument position $i \in \{1, \ldots, n\}$. Every simple freeze $❄$ induces the following partial mapping on non-variable terms $t = f(t_1, \ldots, t_n)$, also denoted by $❄$:

- if $❄(f)$ is undefined or $n = 0$ then $❄(t) = t$,
- if $❄(f) = i$ and $t_i = g(u_1, \ldots, u_m)$ then
$$❄(t) = ❄_{fg}(t_1, \ldots, t_{i-1}, u_1, \ldots, u_m, t_{i+1}, \ldots, t_n)$$
where $❄_{fg}$ is a fresh function symbol of arity $m + n - 1$,
- if $❄(f) = i$ and t_i is a variable then $❄(t)$ is undefined.

We denote $\{❄(\ell) \to ❄(r) \mid \ell \to r \in \mathcal{R}\}$ by $❄(\mathcal{R})$.

Now uncurrying for dependency pair symbols is formulated with the simple freeze $❄(\star^\sharp) = 1$, transforming $f_n(t_1, \ldots, t_n) \star^\sharp t_{n+1}$ to $❄_{\star^\sharp f_n}(t_1, \ldots, t_n, t_{n+1})$. Writing $f^\sharp_{n+1}$ for $❄_{\star^\sharp f_n}$, we obtain the frozen term $f^\sharp_{n+1}(t_1, \ldots, t_n, t_{n+1})$. In Example 8.56 we have
$$❄(\{\mathsf{a}_1(x) \star^\sharp \mathsf{a} \to \mathsf{a}_1(\mathsf{a}_1(\mathsf{a})) \star^\sharp x\}) = \{\mathsf{a}^\sharp_2(x, \mathsf{a}) \to \mathsf{a}^\sharp_2(\mathsf{a}_1(\mathsf{a}), x)\}$$

The next definition introduces a condition that remedies that freezing is not sound in general (cf. Example 8.65).

Definition 8.58. A term t is *strongly root stable* with respect to a TRS $\mathcal{R}$ if $t\sigma \to^*_{\mathcal{R}} \cdot \to^\epsilon_{\mathcal{R}} u$ does not hold for any substitution σ and term u. Let $❄$ be a simple freeze. A DP problem $(\mathcal{P}, \mathcal{R})$ is $❄$*-stable* if $❄(\mathcal{P})$ is well-defined and t_i is strongly root stable for $\mathcal{R}$ whenever $s \to f(t_1, \ldots, t_n) \in \mathcal{P}$ and $❄(f) = i$.

Definition 8.59. Let $(\mathcal{P}, \mathcal{R})$ be a DP problem and $❄$ a simple freeze. The DP processor $❄$ is defined as
$$(\mathcal{P}, \mathcal{R}) \mapsto \begin{cases} \{(❄(\mathcal{P}), \mathcal{R})\} & \text{if } (\mathcal{P}, \mathcal{R}) \text{ is } ❄\text{-stable} \\ \{(\mathcal{P}, \mathcal{R})\} & \text{otherwise} \end{cases}$$
Furthermore, the DP processor $\mathcal{U}_2$ is defined as
$$(\mathcal{P}, \mathcal{R}) \mapsto \{❄((\mathcal{P}', \mathcal{R}')) \mid (\mathcal{P}', \mathcal{R}') \in \mathcal{U}_1((\mathcal{P}, \mathcal{R}))\}$$
where $❄(\star^\sharp) = 1$.

The DP processor $❄$ exploits the fact that a root step in $\mathcal{P}$ gives rise to a root step in $❄(\mathcal{P})$ and vice versa. This follows from the root stability of the left argument of left-hand sides rooted by $\star^\sharp$, which is a consequence of the $❄$-stability of $(\mathcal{P}, \mathcal{R})$. Moreover, $t \to^*_{\mathcal{R}} u$ if and only if $❄(t) \to^*_{❄(\mathcal{R})} ❄(u)$ because $\star^\sharp$ does not occur in the rules of $\mathcal{R}$.

8.6.1. Full Termination

Recently it has been observed by Sternagel and Thiemann [165] that the signature influences whether a DP problem is finite or not. In particular, restricting the signature of a non-finite DP problem $(\mathcal{P}, \mathcal{R})$ to the function symbols that occur in $\mathcal{P} \cup \mathcal{R}$ may make it finite. This is in sharp contrast to (innermost) termination of TRSs [120] (and *innermost non-finiteness* of DP problems, cf. Lemma 8.70).

We first show that for ℓ-applicative DP problems this cannot happen.[2]

Lemma 8.60. *Let $(\mathcal{P}, \mathcal{R})$ be an ℓ-applicative DP problem over the signature $\mathcal{G}$. If $(\mathcal{P}, \mathcal{R})$ is finite over $\mathcal{F}$ and $\star \in \mathcal{F}$ then $(\mathcal{P}, \mathcal{R})$ is finite over $\mathcal{G}$.*

Proof. Let $\mathcal{V}' = \mathcal{V} \uplus \{x_f \mid f \in \mathcal{G} \setminus \mathcal{F}\}$. We define a mapping I from $\mathcal{T}(\mathcal{G}, \mathcal{V})$ to $\mathcal{T}(\mathcal{F}, \mathcal{V}')$ as follows:

$$I(t) = \begin{cases} t & \text{if } t \in \mathcal{V} \\ f(I(t_1), \ldots, I(t_n)) & \text{if } t = f(t_1, \ldots, t_n) \text{ and } f \in \mathcal{F} \\ x_f \star I(t_1) \star \cdots \star I(t_n) & \text{if } t = f(t_1, \ldots, t_n) \text{ and } f \in \mathcal{G} \setminus \mathcal{F} \end{cases}$$

Note that $I(u) = u$ for $u \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. Obviously $I(C[t\sigma]) = I(C)[I(t)I(\sigma)]$ and hence we immediately obtain that $s \to_{\mathcal{P}}^{\epsilon} t$ implies $I(s) \to_{\mathcal{P}}^{\epsilon} I(t)$ and that $s \to_{\mathcal{R}} t$ implies $I(s) \to_{\mathcal{R}} I(t)$ because $\ell = I(\ell)$ and $r = I(r)$ for all $\ell \to r \in \mathcal{P} \cup \mathcal{R}$. To show that any $I(t)$ is terminating with respect to $\mathcal{R}$ whenever $t \in \mathcal{T}(\mathcal{G}, \mathcal{V})$ is terminating with respect to $\mathcal{R}$, we show that if $I(t) \to_{\mathcal{R}} u$ for some term $u \in \mathcal{T}(\mathcal{F}, \mathcal{V}')$ then there exists a term $s \in \mathcal{T}(\mathcal{G}, \mathcal{V})$ with $t \to_{\mathcal{R}} s$ and $I(s) = u$. Now let $I(t) = C[\ell\sigma] \to_{\mathcal{R}} C[r\sigma] = u$ for some $\ell \to r \in \mathcal{R}$. Clearly $t = C'[v\sigma']$ for some C', v, and σ' with $C = I(C')$, $\ell = I(v)$, and $\sigma = I(\sigma')$. By left head variable freeness of $\mathcal{R}$ we obtain $v = \ell \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. Hence by injectivity of I we conclude $t = C'[\ell\sigma'] \to_{\mathcal{R}} C'[r\sigma'] = s$. These observations guarantee that any presupposed minimal sequence using terms from $\mathcal{T}(\mathcal{G}, \mathcal{V})$ is transformed by the mapping I into a minimal sequence using terms from $\mathcal{T}(\mathcal{F}, \mathcal{V}')$. It follows that $(\mathcal{P}, \mathcal{R})$ is finite over the signature $\mathcal{G}$. □

The next result prepares for the soundness proof of $\mathcal{U}_1$.

Lemma 8.61. *Let $(\mathcal{P}, \mathcal{R})$ be an ℓ-applicative DP problem. If $\ell\sigma \to^{\epsilon} r\sigma$ with $\ell \to r \in \mathcal{P}$ then $(\ell\sigma){\downarrow_{\mathcal{U}(\mathcal{F})}} \to_{\mathcal{P}{\downarrow_{\mathcal{U}(\mathcal{F})}}}^{\epsilon} r{\downarrow_{\mathcal{U}(\mathcal{F})}}\sigma{\downarrow_{\mathcal{U}(\mathcal{F})}}$.*

Proof. By Lemma 8.22 and the assumption that ℓ is head variable free. □

Theorem 8.62. *The DP processor $\mathcal{U}_1$ is sound and complete.*

[2]An alternative proof has been independently obtained by Sternagel and Thiemann [164].

Proof. Let $(\mathcal{P},\mathcal{R})$ be an ℓ-applicative DP problem. If $\star \notin \mathcal{F}$ then $\mathcal{U}_1((\mathcal{P},\mathcal{R})) = \{(\mathcal{P},\mathcal{R})\}$ and there is nothing to show. If $\star \in \mathcal{F}$ then according to the preceding lemma we may assume that the signature of the DP problem $(\mathcal{P},\mathcal{R})$ is $\mathcal{F}$.

We first show soundness. Suppose the DP problem $(\mathcal{P}{\downarrow_{\mathcal{U}}}, \mathcal{U}^{+}_{\eta}(\mathcal{R}))$ is finite. We have to show that $(\mathcal{P},\mathcal{R})$ is finite. Suppose to the contrary that $(\mathcal{P},\mathcal{R})$ is not finite. So there exists a minimal rewrite sequence

$$s_1 \to^{\epsilon}_{\mathcal{P}} t_1 \to^{*}_{\mathcal{R}} s_2 \to^{\epsilon}_{\mathcal{P}} t_2 \to^{*}_{\mathcal{R}} \cdots \tag{8.2}$$

By Lemmata 8.61 and 8.22 together with the claim in the proof of Theorem 8.17, this sequence can be transformed into

$$s_1{\downarrow_{\mathcal{U}}} \to^{\epsilon}_{\mathcal{P}\downarrow_{\mathcal{U}}} u_1 \to^{*}_{\mathcal{U}} t_1{\downarrow_{\mathcal{U}}} \to^{*}_{\mathcal{U}^{+}_{\eta}(\mathcal{R})} s_2{\downarrow_{\mathcal{U}}} \to^{\epsilon}_{\mathcal{P}\downarrow_{\mathcal{U}}} u_2 \to^{*}_{\mathcal{U}} t_2{\downarrow_{\mathcal{U}}} \to^{*}_{\mathcal{U}^{+}_{\eta}(\mathcal{R})} \cdots$$

It remains to show that all terms u_1, u_2, ... are terminating with respect to $\mathcal{U}^{+}_{\eta}(\mathcal{R})$. Fix i. We have $u_i{\downarrow_{\mathcal{C}'}} = t_i{\downarrow_{\mathcal{U}}}{\downarrow_{\mathcal{C}'}} = t_i$. Due to the minimality of (8.2), t_i is terminating with respect to $\mathcal{R}$ and, according to Lemma 8.16, also with respect to $\mathcal{R}_\eta$. Hence, due to the proof of Theorem 8.28, u_i is terminating with respect to $\mathcal{U}^{+}_{\eta}(\mathcal{R})$.

Next we show completeness of the DP processor $\mathcal{U}_1$. Suppose that the DP problem $(\mathcal{P}{\downarrow_{\mathcal{U}}}, \mathcal{U}^{+}_{\eta}(\mathcal{R}))$ is not finite. So there exists a minimal rewrite sequence

$$s_1 \to^{\epsilon}_{\mathcal{P}\downarrow_{\mathcal{U}}} t_1 \to^{*}_{\mathcal{U}^{+}_{\eta}(\mathcal{R})} s_2 \to^{\epsilon}_{\mathcal{P}\downarrow_{\mathcal{U}}} t_2 \to^{*}_{\mathcal{U}^{+}_{\eta}(\mathcal{R})} \cdots$$

Using Lemmata 8.26 and 8.27 this sequence can be transformed into

$$s_1{\downarrow_{\mathcal{C}'}} \to^{\epsilon}_{\mathcal{P}} t_1{\downarrow_{\mathcal{C}'}} \to^{*}_{\mathcal{R}_\eta} s_2{\downarrow_{\mathcal{C}'}} \to^{\epsilon}_{\mathcal{P}} t_2{\downarrow_{\mathcal{C}'}} \to^{*}_{\mathcal{R}_\eta} \cdots$$

In order to conclude that the DP problem $(\mathcal{P},\mathcal{R})$ is not finite, it remains to show that the terms $t_1{\downarrow_{\mathcal{C}'}}$, $t_2{\downarrow_{\mathcal{C}'}}$, ... are terminating with respect to $\mathcal{R}_\eta$. This follows from the assumption that the terms t_1, t_2, ... are terminating with respect to $\mathcal{U}^{+}_{\eta}(\mathcal{R})$ in connection with the proof of Theorem 8.17. An application of Lemma 8.16 concludes the proof. □

Theorem 8.63. *The DP processor* ❄ *is sound and complete.*

Proof. Let $(\mathcal{P},\mathcal{R})$ be a ❄-stable DP problem. We show that every minimal rewrite sequence

$$s_1 \to^{\epsilon}_{\mathcal{P}} t_1 \to^{*}_{\mathcal{R}} s_2 \to^{\epsilon}_{\mathcal{P}} t_2 \to^{*}_{\mathcal{R}} \cdots$$

can be transformed into the minimal sequence

$$❄(s_1) \to^{\epsilon}_{❄(\mathcal{P})} ❄(t_1) \to^{*}_{\mathcal{R}} ❄(s_2) \to^{\epsilon}_{❄(\mathcal{P})} ❄(t_2) \to^{*}_{\mathcal{R}} \cdots$$

and vice versa. This follows from the following three observations.

$s_i \to_{\mathcal{P}}^{\epsilon} t_i$ if and only if $\circledast(s_i) \to_{\circledast(\mathcal{P})}^{\epsilon} \circledast(t_i)$

We have $s_i \to_{\mathcal{P}}^{\epsilon} t_i$ if and only if $s_i = \ell\sigma$ and $t_i = r\sigma$ with $\ell \to r \in \mathcal{P}$. Since $\circledast(\mathcal{P})$ is well-defined and $\circledast$ is injective on terms, the latter is equivalent to

$$\circledast(s_i) = \circledast(\ell\sigma) = \circledast(\ell)\sigma \to_{\circledast(\mathcal{P})}^{\epsilon} \circledast(r)\sigma = \circledast(r\sigma) = \circledast(t_i)$$

$t_i \to_{\mathcal{R}}^{*} s_{i+1}$ if and only if $\circledast(t_i) \to_{\mathcal{R}}^{*} \circledast(s_{i+1})$

Since t_i and s_{i+1} have the same root symbol we can write $t_i = f(u_1, \dots, u_n)$ and $s_{i+1} = f(u'_1, \dots, u'_n)$. If $\circledast(f)$ is undefined or $n = 0$ then $\circledast(s_i) = s_i$, $s_i \to_{\mathcal{R}}^{*} t_i$, and $t_i = \circledast(t_i)$. Suppose $\circledast(f) = k$. Since t_i is an instance of a right-hand side of a pair in $\mathcal{P}$ and $\circledast(\mathcal{P})$ is well-defined, u_k cannot be a variable. Write $u_k = g(v_1, \dots, v_m)$. According to $\circledast$-stability, u_k is root stable and thus $u'_k = g(v'_1, \dots, v'_m)$. Hence

$$\begin{aligned} t_i &= f(u_1, \dots, u_{k-1}, g(v_1, \dots, v_m), u_{k+1}, \dots, u_n) \\ s_{i+1} &= f(u'_1, \dots, u'_{k-1}, g(v'_1, \dots, v'_m), u'_{k+1}, \dots, u'_n) \end{aligned}$$

and

$$\begin{aligned} \circledast(t_i) &= \circledast_{fg}(u_1, \dots, u_{k-1}, v_1, \dots, v_m, u_{k+1}, \dots, u_n) \\ \circledast(s_{i+1}) &= \circledast_{fg}(u'_1, \dots, u'_{k-1}, v'_1, \dots, v'_m, u'_{k+1}, \dots, u'_n). \end{aligned}$$

Consequently, $t_i \to_{\mathcal{R}}^{*} s_{i+1}$ if and only if $u_j \to_{\mathcal{R}}^{*} u'_j$ for $1 \leqslant j \leqslant n$ with $j \neq k$ and $v_j \to_{\mathcal{R}}^{*} v'_j$ for $1 \leqslant j \leqslant m$ if and only if $\circledast(t_i) \to_{\mathcal{R}}^{*} \circledast(s_{i+1})$.

t_i terminates with respect to $\mathcal{R}$ if and only if $\circledast(t_i)$ terminates with respect to $\mathcal{R}$

This follows immediately from the observation above that all reductions in t_i take place in the arguments u_j or v_j. □

Corollary 8.64. *The DP processor $\mathcal{U}_2$ is sound and complete.*

Proof. Immediate from Theorems 8.62 and 8.63 together with the fact that the composition of sound and complete DP processors yields a sound and complete DP processor. □

The next example shows that $\circledast$-stability is essential for soundness.

Example 8.65. Consider the non-terminating ATRS $\mathcal{R}$ consisting of the two rules

$$\mathsf{f}\ \mathsf{a} \to \mathsf{g}\ \mathsf{a} \qquad\qquad \mathsf{g} \to \mathsf{f}$$

which induces the infinite DP problem $(\mathcal{P}, \mathcal{R})$ with $\mathcal{P}$ consisting of the rules

$$\mathsf{f} \mathbin{\sharp} \mathsf{a} \to \mathsf{g} \mathbin{\sharp} \mathsf{a} \qquad\qquad \mathsf{f} \mathbin{\sharp} \mathsf{a} \to \mathsf{g}^\sharp$$

Since $\mathcal{P}{\downarrow_\mathcal{U}} = \mathcal{P}$ and $\mathcal{U}_1$ is sound, the DP problem $(\mathcal{P}, \mathcal{U}_\eta^+(\mathcal{R}))$ is also infinite. The set $❄(\mathcal{P}{\downarrow_\mathcal{U}})$ consists of the rules

$$\mathsf{f}_1^\sharp(\mathsf{a}) \to \mathsf{g}_1^\sharp(\mathsf{a}) \qquad\qquad \mathsf{f}_1^\sharp(\mathsf{a}) \to \mathsf{g}^\sharp$$

Clearly, the DP problem $(❄(\mathcal{P}), \mathcal{U}_\eta^+(\mathcal{R}))$ is finite. Note that $(\mathcal{P}, \mathcal{U}_\eta^+(\mathcal{R}))$ is not ❄-stable as $\mathsf{g} \to^{\epsilon}_{\mathcal{U}_\eta^+(\mathcal{R})} \mathsf{f}$.

Since ❄-stability is undecidable in general, for automation we need to approximate strong root stability. We present a simple criterion which is based on the term approximation TCAP from [59], where it was used to give a better approximation of dependency graphs.

Definition 8.66 ([59])**.** Let $\mathcal{R}$ be a TRS and t a term. The term $\mathsf{TCAP}_\mathcal{R}(t)$ is inductively defined as follows. If t is a variable, $\mathsf{TCAP}_\mathcal{R}(t)$ is a fresh variable. If $t = f(t_1, \ldots, t_n)$ then we let $u = f(\mathsf{TCAP}_\mathcal{R}(t_1), \ldots, \mathsf{TCAP}_\mathcal{R}(t_n))$ and define $\mathsf{TCAP}_\mathcal{R}(t)$ to be u if u does not unify with the left-hand side of a rule in $\mathcal{R}$, and a fresh variable otherwise.

Lemma 8.67. *A term t is strongly root stable for a TRS $\mathcal{R}$ if $\mathsf{TCAP}_\mathcal{R}(t) \notin \mathcal{V}$.*

Proof. The only possibility for $\mathsf{TCAP}_\mathcal{R}(t) \notin \mathcal{V}$ is when $t = f(t_1, \ldots, t_n)$ and $u = f(\mathsf{TCAP}_\mathcal{R}(t_1), \ldots, \mathsf{TCAP}_\mathcal{R}(t_n))$ does not unify with a left-hand side of a rule in $\mathcal{R}$. Assume to the contrary that t is not strongly root stable. Then there are a substitution σ and a left-hand side ℓ of a rule in $\mathcal{R}$ such that $t\sigma \to_\mathcal{R}^* \ell\tau$. Write $\ell = f(l_1, \ldots, l_n)$. We have $t\sigma = f(t_1\sigma, \ldots, t_n\sigma)$ with $t_i\sigma \to_\mathcal{R}^* l_i\tau$ for $1 \leqslant i \leqslant n$. Hence $\mathsf{TCAP}_\mathcal{R}(t_i)\delta_i = l_i\tau$ for some substitution δ_i ([59, proof of Theorem 13]). Since the terms $\mathsf{TCAP}_\mathcal{R}(t_1), \ldots, \mathsf{TCAP}_\mathcal{R}(t_n)$ are linear and do not share variables, it follows that u unifies with ℓ, contradicting the assumption. ☐

Example 8.68. Consider the DP problem $(\mathcal{P}{\downarrow_\mathcal{U}}, \mathcal{U}_\eta^+(\mathcal{R}))$ of Example 8.56 with $\mathcal{P}{\downarrow_\mathcal{U}}$ consisting of the rule

$$\mathsf{a}_1(x) \star^\sharp \mathsf{a} \to \mathsf{a}_1(\mathsf{a}_1(\mathsf{a})) \star^\sharp x$$

and $\mathcal{U}_\eta^+(\mathcal{R})$ consisting of the rules

$$\mathsf{a}_2(x, \mathsf{a}) \to \mathsf{a}_2(\mathsf{a}_1(\mathsf{a}), x) \qquad \mathsf{a} \star x \to \mathsf{a}_1(x) \qquad \mathsf{a}_1(x) \star y \to \mathsf{a}_2(x, y)$$

Since $\mathsf{TCAP}_{\mathcal{U}_\eta^+(\mathcal{R})}(\mathsf{a}_1(\mathsf{a}_1(\mathsf{a}))) = \mathsf{a}_1(\mathsf{a}_1(\mathsf{a}))$ is not a variable, $\mathsf{a}_1(\mathsf{a}_1(\mathsf{a}))$ is strongly root stable. Hence $(\mathcal{P}{\downarrow_\mathcal{U}}, \mathcal{U}_\eta^+(\mathcal{R}))$ is ❄-stable.

8.6.2. Innermost Termination

We start this section with a motivating example (which is related to Example 8.55).

Example 8.69. Consider the ATRS $\mathcal{R}$ consisting of the rule $x\ (\mathsf{a}\ \mathsf{a}) \to (\mathsf{a}\ \mathsf{a}\ \mathsf{a})\ x$. The only SCC in the dependency graph is $\mathcal{P} := \{x \mathbin{\sharp} (\mathsf{a}\ \mathsf{a}) \to (\mathsf{a}\ \mathsf{a}\ \mathsf{a}) \mathbin{\sharp} x\}$. Since $\mathcal{P} \cup \mathcal{R}$ is not left head variable free the processor $\mathcal{U}_1$ cannot be applied. For proving innermost termination the usable rules processor [59] transforms $(\mathcal{P}, \mathcal{R})$ into $(\mathcal{P}, \varnothing)$ since the rule in $\mathcal{R}$ is not usable. Because $\mathcal{U}_1$ is sound for innermost termination (cf. Theorem 8.71), Example 8.55 finishes the innermost termination proof of $\mathcal{R}$.

In the following we deal with applicative DP problems $(\mathcal{P}, \mathcal{R})$ for innermost termination. The following is the counterpart of Lemma 8.60 for innermost termination. Note that in contrast to Lemma 8.60, the result holds for arbitrary DP problems.

Lemma 8.70. *Let $(\mathcal{P}, \mathcal{R})$ be a DP problem over the signature $\mathcal{G}$. If $(\mathcal{P}, \mathcal{R})$ is innermost finite over $\mathcal{F}$ then $(\mathcal{P}, \mathcal{R})$ is innermost finite over $\mathcal{G}$.*

Proof. Let $\mathcal{V}' = \mathcal{V} \uplus \{x_t \mid t \in \mathcal{T}(\mathcal{G}, \mathcal{V})\}$. Similar to the proof of Lemma 8.60 we define a mapping I from $\mathcal{T}(\mathcal{G}, \mathcal{V})$ to $\mathcal{T}(\mathcal{F}, \mathcal{V}')$ as follows:

$$I(t) = \begin{cases} t & \text{if } t \in \mathcal{V} \\ f(I(t_1), \dots, I(t_n)) & \text{if } t = f(t_1, \dots, t_n) \text{ and } f \in \mathcal{F} \\ x_t & \text{if } t = f(t_1, \dots, t_n) \text{ and } f \in \mathcal{G} \setminus \mathcal{F} \end{cases}$$

Note that $I(t) = t$ for $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. First we show that $s \to_{\mathcal{P}}^{\epsilon} t$ implies $I(s) \to_{\mathcal{P}}^{\epsilon} I(t)$. From $s \to_{\mathcal{P}}^{\epsilon} t$ we get $s = \ell\sigma$ and $t = r\sigma$ for some $\ell \to r \in \mathcal{P}$. Since $\ell, r \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ we get $\ell = I(\ell)$ and $r = I(r)$ and consequently $I(s) = \ell I(\sigma) \to_{\mathcal{P}}^{\epsilon} rI(\sigma) = I(t)$. Since $t \to_{\mathcal{R}} s$ need not imply $I(t) \to_{\mathcal{R}} I(s)$ in general, we restrict ourselves to terms satisfying a special property. Let T be the set of all terms in $\mathcal{T}(\mathcal{G}, \mathcal{V})$ whose $\mathcal{G} \setminus \mathcal{F}$-rooted subterms are normal forms. Note that if $C[\ell\sigma] \in T$ for some $\ell \to r \in \mathcal{R}$ then $I(C) = C$ since C cannot contain a symbol from $\mathcal{G} \setminus \mathcal{F}$. Hence $t \to_{\mathcal{R}} s$ with $t \in T$ implies $I(t) = C[\ell I(\sigma)] \to_{\mathcal{R}} C[rI(\sigma)] = I(s)$ and $I(s) \in T$ (since $\to_{\mathcal{R}}$ cannot introduce function symbols from $\mathcal{G} \setminus \mathcal{F}$). Together with the fact that $I(s) \in NF(\mathcal{R})$ whenever $s \in NF(\mathcal{R})$ this ensures that $I(t) \xrightarrow{\mathsf{i}}{}_{\mathcal{R}}^{!} I(s)$ whenever $t \xrightarrow{\mathsf{i}}{}_{\mathcal{R}}^{!} s$ and $t \in T$. Hence any presupposed sequence

$$s_1 \to_{\mathcal{P}}^{\epsilon} t_1 \xrightarrow{\mathsf{i}}{}_{\mathcal{R}}^{!} s_2 \to_{\mathcal{P}}^{\epsilon} t_2 \xrightarrow{\mathsf{i}}{}_{\mathcal{R}}^{!} \cdots$$

with $s_1 \in \mathit{NF}(\mathcal{R})$ (and hence $s_i, t_i \in T$ for all $i \geqslant 1$) using terms from $\mathcal{T}(\mathcal{G}, \mathcal{V})$ is transformed into a sequence

$$I(s_1) \to^{\epsilon}_{\mathcal{P}} I(t_1) \xrightarrow{\mathsf{i}}{}^{!}_{\mathcal{R}} I(s_2) \to^{\epsilon}_{\mathcal{P}} I(t_2) \xrightarrow{\mathsf{i}}{}^{!}_{\mathcal{R}} \cdots$$

with $I(s_1) \in \mathit{NF}(\mathcal{R})$ using terms from $\mathcal{T}(\mathcal{F}, \mathcal{V}')$. It follows that $(\mathcal{P}, \mathcal{R})$ is innermost finite over $\mathcal{G}$. □

Theorem 8.71. *The DP processor $\mathcal{U}_1$ is sound for innermost termination.*

Proof. Let $(\mathcal{P}, \mathcal{R})$ be an ℓ-applicative DP problem. By the preceding lemma we may assume without loss of generality that the signature of $(\mathcal{P}, \mathcal{R})$ is $\mathcal{F}$. Assume $(\mathcal{P}, \mathcal{R})$ is not innermost finite. According to Krishna Rao [109] there exists an infinite sequence

$$s_1 \to^{\epsilon}_{\mathcal{P}} t_1 \xrightarrow{\mathsf{ri}}{}^{!}_{\mathcal{R}} s_2 \to^{\epsilon}_{\mathcal{P}} t_2 \xrightarrow{\mathsf{ri}}{}^{!}_{\mathcal{R}} \cdots$$

with $s_1 \in \mathit{NF}(\mathcal{R})$. We show that there is a sequence

$$s_1{\downarrow_{\mathcal{U}}} \to^{\epsilon}_{\mathcal{P}{\downarrow_{\mathcal{U}}}} t'_1 \xrightarrow{\mathsf{i}}{}^{!}_{\mathcal{U}^+_\eta(\mathcal{R})} s_2{\downarrow_{\mathcal{U}}} \to^{\epsilon}_{\mathcal{P}{\downarrow_{\mathcal{U}}}} t'_2 \xrightarrow{\mathsf{i}}{}^{!}_{\mathcal{U}^+_\eta(\mathcal{R})} \cdots$$

with terms t'_1, t'_2, … such that $t_i \to^*_{\mathcal{U}} t'_i$ for $i \geqslant 1$. Fix i and let $\ell \to r$ be the rule from $\mathcal{P}$ that is used in $s_i \to^{\epsilon}_{\mathcal{P}} t_i$. So $t_i = r\sigma$ for some substitution σ. Lemma 8.61 yields $s_i{\downarrow_{\mathcal{U}}} \to^{\epsilon}_{\mathcal{P}{\downarrow_{\mathcal{U}}}} t'_i$ for the term $t'_i = r{\downarrow_{\mathcal{U}}}\sigma{\downarrow_{\mathcal{U}}}$. Clearly $t_i \to^*_{\mathcal{U}} t'_i$. Repeated application of Lemma 8.41 yields

$$t'_i \xrightarrow{\mathsf{i}}{}^{*}_{\mathcal{U}^+_\eta(\mathcal{R})} s'_{i+1} \;{}_{\mathcal{U}}^{*}{\leftarrow}\; s_{i+1}$$

for some term s'_{i+1} that is a normal form of $\mathcal{R}_\eta{\downarrow_{\mathcal{U}}}$ due to Lemma 8.37 and the fact that s_{i+1} is a normal form of $\mathcal{R}$. It follows that $s'_{i+1} \xrightarrow{\mathsf{i}}{}^{*}_{\mathcal{U}^+_\eta(\mathcal{R})} s_{i+1}{\downarrow_{\mathcal{U}}}$ by repeated applications of Lemma 8.37 and innermost normalizing s'_{i+1} with respect to $\mathcal{U}$. (Note that $\mathcal{U}$ is terminating and confluent.) Since $s_{i+1}{\downarrow_{\mathcal{U}}}$ is a normal form of $\mathcal{U}^+_\eta(\mathcal{R})$, we obtain $t'_i \xrightarrow{\mathsf{i}}{}^{!}_{\mathcal{U}^+_\eta(\mathcal{R})} s_{i+1}{\downarrow_{\mathcal{U}}}$. Since $s_1{\downarrow_{\mathcal{U}}} \in \mathit{NF}(\mathcal{U}^+_\eta(\mathcal{R}))$ whenever $s_1 \in \mathit{NF}(\mathcal{R})$ (Lemma 8.37) we conclude that the DP problem $(\mathcal{P}{\downarrow_{\mathcal{U}}}, \mathcal{U}^+_\eta(\mathcal{R}))$ is not innermost finite, as desired. □

Theorem 8.72. *The DP processor ❄ is sound and complete for innermost termination.*

Proof. Let $(\mathcal{P}, \mathcal{R})$ be ❄-stable. Every infinite sequence

$$s_1 \to^{\epsilon}_{\mathcal{P}} t_1 \xrightarrow{\mathsf{i}}{}^{!}_{\mathcal{R}} s_2 \to^{\epsilon}_{\mathcal{P}} t_2 \xrightarrow{\mathsf{i}}{}^{!}_{\mathcal{R}} \cdots$$

can be transformed into the sequence

$$\circledast(s_1) \rightarrow^{\epsilon}_{\circledast(\mathcal{P})} \circledast(t_1) \xrightarrow{\mathsf{i}}{}^{!}_{\mathcal{R}} \circledast(s_2) \rightarrow^{\epsilon}_{\circledast(\mathcal{P})} \circledast(t_2) \xrightarrow{\mathsf{i}}{}^{!}_{\mathcal{R}} \cdots$$

and vice versa. This is obvious from the first observation in the proof of Theorem 8.63 and the following two facts: (1) $t_i \xrightarrow{\mathsf{i}}{}^{*}_{\mathcal{R}} s_{i+1}$ if and only if $\circledast(t_i) \xrightarrow{\mathsf{i}}{}^{*}_{\mathcal{R}} \circledast(s_{i+1})$ and (2) $s_i \in \mathit{NF}(\mathcal{R})$ if and only if $\circledast(s_i) \in \mathit{NF}(\mathcal{R})$ for all $i \geqslant 1$ (which follow from the proof of the second observation in the proof of Theorem 8.63). □

Corollary 8.73. *The DP processor $\mathcal{U}_2$ is sound for innermost termination.*

Proof. Immediate from Theorems 8.71 and 8.72 together with the fact that the composition of sound DP processors yields a sound processor. □

The next example shows that $\mathcal{U}_1$ is not complete for innermost DP problems. (Note that Example 8.32 on page 205 does not provide a counterexample.)

Example 8.74. Consider the ATRS $\mathcal{R}$

$$\mathsf{f}\ x \rightarrow \mathsf{g}\ \mathsf{a}\ x \qquad\qquad \mathsf{g}\ x \rightarrow \mathsf{f} \qquad\qquad \mathsf{g} \rightarrow \mathsf{h}$$

which is innermost terminating because the rule $\mathsf{g}\ x \rightarrow \mathsf{f}$ cannot be used in an innermost sequence and without this rule $\mathcal{R}$ is easily seen to be terminating. Hence also the DP problem $(\mathcal{P}, \mathcal{R})$ with $\mathcal{P}$ consisting of

$$\mathsf{f}^{\sharp}\ x \rightarrow \mathsf{g}\ \mathsf{a}^{\sharp}\ x \qquad\qquad \mathsf{f}^{\sharp}\ x \rightarrow \mathsf{g}^{\sharp}\ \mathsf{a} \qquad\qquad \mathsf{g}^{\sharp}\ x \rightarrow \mathsf{f}$$

is innermost terminating. However, after applying the DP processor $\mathcal{U}_1$ there is an infinite innermost sequence:

$$\mathsf{f}^{\sharp}\ x \rightarrow^{\epsilon}_{\mathcal{P}\downarrow_{\mathcal{U}}} \mathsf{g}_1(\mathsf{a})^{\sharp}\ x \xrightarrow{\mathsf{i}}_{\mathcal{U}^{+}_{\eta}(\mathcal{R})} \mathsf{f}^{\sharp}\ x \rightarrow^{\epsilon}_{\mathcal{P}\downarrow_{\mathcal{U}}} \cdots$$

Note that $\mathsf{f}^{\sharp}\ x \in \mathit{NF}(\mathcal{U}^{+}_{\eta}(\mathcal{R}))$.

Because of the completeness of $\circledast$, $\mathcal{U}_2$ inherits incompleteness for innermost termination from $\mathcal{U}_1$.

8.7. Experiments

The transformations presented in this paper are implemented in the termination prover $\mathsf{T_TT_2}$ [107]. For experimentation version 7.0.2 of the termination problem data base (TPDB)[3] has been considered which contains 195 ATRSs for full and

[3]http://termination-portal.org/wiki/TPDB

	direct			as processor						
	8.7	8.17	8.17 8.30	none	$\mathcal{A}$	$\mathcal{A}'$	$\mathcal{U}_1$	$\mathcal{U}_2$	8.17 8.30 $\mathcal{U}_2$	$\sum$
subterm criterion	1	47	48	41	–	–	41	58	61	61
matrix (1)	4	90	101	66	68	86	95	101	109	110
matrix (2)	7	108	131	108	111	128	133	134	138	138
matrix (3)	9	109	132	110	114	133	136	138	140	142

Table 8.1.: Full termination for 195 ATRSs.

18 for innermost rewriting. All tests have been performed on a single core of a server equipped with eight dual-core AMD Opteron® processors 885 running at a clock rate of 2.6 GHz and 64 GB of main memory. Comprehensive details of the experiments[4] give evidence that the proposed transformations ease proving termination and upper bounds on the derivational complexity.

For proving (innermost) termination we considered two popular termination methods, namely the subterm criterion [73] and matrix interpretations [44] of dimensions one to three. For a matrix of dimension d the coefficients are represented by $5 - d$ bits, one additional bit is allowed for intermediate results. Both methods are integrated within the dependency pair framework using dependency graph reasoning and usable rules as proposed in [59, 60, 68].

Table 8.1 differentiates between applying the transformations as a preprocessing step (direct) or within the dependency pair framework (as processor). For rows labeled "matrix", the numbers in parentheses refer to the dimension of the interpretations. The direct method of Corollary 8.7 (Theorem 8.17, Theorems 8.17 and 8.30) applies to 10 (141, 170) systems. If used directly, the numbers in the table refer to the systems that could be proved terminating in case of a successful transformation. Mirroring (when the original system is not left head variable free) does increase applicability of our (direct) transformation significantly.

The middle part of Table 8.1 states the number of successful termination proofs for transformation $\mathcal{A}$ ([59, 174]) and the processors $\mathcal{U}_1$ (Definition 8.54) and $\mathcal{U}_2$ (Definition 8.59). Since transformation $\mathcal{A}$ does not preserve minimality (Example 8.75 in Section 8.8) one cannot use it together with the subterm criterion. In [174] it is shown that minimality is preserved when the transformation $\mathcal{A}$ is fused with the reduction pair and usable rules processors. Our implementation

[4]`http://cl-informatik.uibk.ac.at/software/ttt2/11jar/`

	direct			as processor					
			8.42					8.42	
								8.30	
	8.7	8.42	8.30	none	$\mathcal{A}$	$\mathcal{U}_1$	$\mathcal{U}_2$	$\mathcal{U}_2$	$\sum$
subterm criterion	2	60	62	52	–	53	79	83	83
matrix (1)	4	101	112	76	97	114	120	126	126
matrix (2)	7	120	143	120	140	145	146	151	151
matrix (3)	9	121	144	122	144	149	150	153	154

Table 8.2.: Innermost termination for 213 ATRSs.

is based on the processor presented in [174, Theorem 6.17(D)]. In the column labeled $\mathcal{A}$ the transformation is fused with the reduction pair processor based on matrix interpretations while for column $\mathcal{A}'$ in addition the usable rules are computed based on TCAP. The first version is more suitable for a comparison with our processors (since $\mathcal{U}_1$ and $\mathcal{U}_2$ do also not incorporate usable rules) while the second version shows that transformation $\mathcal{A}$ can be combined with other termination criteria to obtain more advanced processors. Nevertheless the processors $\mathcal{U}_1$ and $\mathcal{U}_2$ admit more successful termination proofs. (In [174] further non-trivial extensions of the transformation $\mathcal{A}$ are considered.)

It is a trivial exercise to extend mirroring to DP problems. Our experiments revealed that (a) mirroring works better for the direct approach (hence we did not incorporate it into the middle block of the table) and (b) the uncurrying processors should be applied before other termination processors. Although Theorem 8.17 and the processor $\mathcal{U}_2$ are incomparable in power we recommend the usage of the processor. One reason is the increased strength and another one the modularity which allows to prevent pitfalls like Example 8.24. Last but not least, the processors $\mathcal{U}_1$ and $\mathcal{U}_2$ are not only sound but also complete (for full termination) which makes them suitable for non-termination analysis in principle. At least with $\mathsf{T_TT_2}$ we could not detect that this makes proving non-termination easier.

The right block of Table 8.1 gives the accumulated score for our transformations. As reference the total number of systems is given (labeled $\sum$) that any method in the corresponding row could prove terminating, showing that the cost for the auxiliary uncurrying rules is negligible compared to the gains in power. To see how the uncurrying transformation improves the power of a "full" termination prover we dropped it from the 2010 competition version of $\mathsf{T_TT_2}$. Then the number of successful termination proofs for applicative TRSs drops from 157 to 131.

	TMI (1)			TMI (2)			TMI (3)			TMI (4)		
			8.43			8.43			8.43			8.43
	–	8.7	8.47	–	8.7	8.47	–	8.7	8.47	–	8.7	8.47
dc/idc	3	3	4	10	10	14	12	14	26	12	16	28

Table 8.3.: (Innermost) derivational complexity for 195 (213) ATRSs.

Table 8.2 shows the results for innermost termination and admits similar conclusions. In contrast to the experiments reported in [202], for this table $\mathsf{T_TT_2}$ uses an approximation for the *innermost* dependency graph [59, 68] and drops non-usable rules [59]. Due to the latter, the results for columns $\mathcal{A}$ and $\mathcal{A}'$ coincide.

Table 8.3 reports the performance of $\mathsf{T_TT_2}$ for derivational complexity. Since $\mathsf{T_TT_2}$ has no special methods for proving *innermost* derivational complexity, the numbers for dc and idc coincide. In this table columns labeled "–" do not use any preprocessing transformation whereas 8.7, 8.43/8.47 indicate applications of Corollary 8.7 and Theorems 8.43/8.47, respectively. We remark that Corollary 8.7 preserves derivational complexity. This is straightforward from [95, Lemma 2.1(3)]. Here TMIs of dimension one to four as presented in Theorem 8.1 are considered. Coefficients of TMIs are represented with $\max\{2, 5 - d\}$ bits; again an additional bit is allowed for intermediate results. If Theorem 8.43 is used as preprocessing transformation, TMIs can, e.g., show 26 systems to have at most cubic derivational complexity while without uncurrying (with Theorem 8.7) the method only applies to 12 (14) systems. Especially for larger dimensions in Table 8.3 our transformation admits significant gains in power. We only tested the direct transformations, because proofs with dependency pairs give upper bounds on the derivational complexity much beyond exponential [124]. Since many of the ATRSs in this testbed contain partially applied terms or head variables in the right-hand sides this hampers applicability of Corollary 8.7.

8.8. Related Work

The transformation $\mathcal{A}$ of Giesl *et al.* [59] requires *proper* applicative DP problems, which are DP problems with the property that all occurrences of each constant have the same number of arguments. No uncurrying rules are added to the processed DP problems. This destroys minimality which means that not all DP processors (i.e., only those not relying on minimality) may be applied after transformation $\mathcal{A}$. The following example is from [174].

Example 8.75. Consider the ℓ-applicative DP problem $(\mathcal{P}, \mathcal{R})$ with $\mathcal{P}$ consisting of the rewrite rule $(\mathsf{g}\ x)\ (\mathsf{h}\ y) \mathbin{\sharp} z \to z\ z \mathbin{\sharp} z$ and $\mathcal{R}$ consisting of the rules

$$\begin{array}{ll} \mathsf{c}\ x\ y \to x & \mathsf{c}\ (\mathsf{g}\ x)\ y \to \mathsf{c}\ (\mathsf{g}\ x)\ y \\ \mathsf{c}\ x\ y \to y & \mathsf{c}\ x\ (\mathsf{g}\ y) \to \mathsf{c}\ x\ (\mathsf{g}\ y) \end{array}$$

The DP problem $(\mathcal{P}, \mathcal{R})$ is not finite because of the minimal rewrite sequence:

$$\begin{aligned} (\mathsf{g}\ x)\ (\mathsf{h}\ x) \mathbin{\sharp} (\mathsf{c}\ \mathsf{g}\ \mathsf{h}\ x) &\to^{\epsilon}_{\mathcal{P}} (\mathsf{c}\ \mathsf{g}\ \mathsf{h}\ x)\ (\mathsf{c}\ \mathsf{g}\ \mathsf{h}\ x) \mathbin{\sharp} (\mathsf{c}\ \mathsf{g}\ \mathsf{h}\ x) \\ &\to_{\mathcal{R}} (\mathsf{g}\ x)\ (\mathsf{c}\ \mathsf{g}\ \mathsf{h}\ x) \mathbin{\sharp} (\mathsf{c}\ \mathsf{g}\ \mathsf{h}\ x) \\ &\to_{\mathcal{R}} (\mathsf{g}\ x)\ (\mathsf{h}\ x) \mathbin{\sharp} (\mathsf{c}\ \mathsf{g}\ \mathsf{h}\ x) \end{aligned}$$

Applying the DP processor $\mathcal{U}_1$ produces $(\mathcal{P}{\downarrow_{\mathcal{U}}}, \mathcal{U}^{+}_{\eta}(\mathcal{R}))$ with $\mathcal{P}{\downarrow_{\mathcal{U}}}$ consisting of the rewrite rule $\mathsf{g}_1(x) \star \mathsf{h}_1(y) \star^{\sharp} z \to z \star z \star^{\sharp} z$ and $\mathcal{U}^{+}_{\eta}(\mathcal{R})$ consisting of the rules

$$\begin{array}{ccc} \mathsf{c}_2(x, y) \to x & \mathsf{c}_2(\mathsf{g}_1(x), y) \to \mathsf{c}_2(\mathsf{g}_1(x), y) & \mathsf{g} \star x \to \mathsf{g}_1(x) \\ \mathsf{c}_2(x, y) \to y & \mathsf{c}_2(x, \mathsf{g}_1(y)) \to \mathsf{c}_2(x, \mathsf{g}_1(y)) & \mathsf{h} \star x \to \mathsf{h}_1(x) \\ \mathsf{c} \star x \to \mathsf{c}_1(x) & \mathsf{c}_1(x) \star y \to \mathsf{c}_2(x, y) & \end{array}$$

This DP problem is not finite:

$$\begin{aligned} \mathsf{g}_1(x) \star \mathsf{h}_1(x) \star^{\sharp} (\mathsf{c}_2(\mathsf{g}, \mathsf{h}) \star x) &\to^{\epsilon}_{\mathcal{P}\downarrow_{\mathcal{U}}} (\mathsf{c}_2(\mathsf{g}, \mathsf{h}) \star x) \star (\mathsf{c}_2(\mathsf{g}, \mathsf{h}) \star x) \star^{\sharp} (\mathsf{c}_2(\mathsf{g}, \mathsf{h}) \star x) \\ &\to^{*}_{\mathcal{U}^{+}_{\eta}(\mathcal{R})} (\mathsf{g} \star x) \star (\mathsf{h} \star x) \star^{\sharp} (\mathsf{c}_2(\mathsf{g}, \mathsf{h}) \star x) \\ &\to^{*}_{\mathcal{U}^{+}_{\eta}(\mathcal{R})} \mathsf{g}_1(x) \star \mathsf{h}_1(x) \star^{\sharp} (\mathsf{c}_2(\mathsf{g}, \mathsf{h}) \star x) \end{aligned}$$

Note that $\mathsf{c}_2(\mathsf{g}, \mathsf{h}) \star x$ is terminating with respect to $\mathcal{U}^{+}_{\eta}(\mathcal{R})$.

The uncurrying rules are essential in this example, even though in the original DP problem all occurrences of each constant have the same number of arguments. Indeed, transformation $\mathcal{A}$ leaves out the uncurrying rules, resulting in a DP problem that admits infinite rewrite sequences but no minimal ones since one has to instantiate the variable z in $\mathsf{g}_1(x) \star \mathsf{h}_1(y) \star^{\sharp} z \to z \star z \star^{\sharp} z$ by a term that contains a subterm of the form $\mathsf{c}_2(\mathsf{g}_1(s), t)$ or $\mathsf{c}_2(s, \mathsf{g}_1(t))$ and the rules $\mathsf{c}_2(\mathsf{g}_1(x), y) \to \mathsf{c}_2(\mathsf{g}_1(x), y)$ and $\mathsf{c}_2(x, \mathsf{g}_1(y)) \to \mathsf{c}_2(x, \mathsf{g}_1(y))$ ensure that these terms are non-terminating.

Thiemann [174, Sections 6.2 and 6.3] addresses the loss of minimality by incorporating reduction pairs, usable rules, and argument filterings into the transformation $\mathcal{A}$. (The first two refinements were considered in the column labeled $\mathcal{A}'$ in Table 8.1.) In [174] it is further remarked that transformation $\mathcal{A}$ works better for innermost termination than for termination. This also holds for our processors $\mathcal{U}_1$ and $\mathcal{U}_2$ (cf. Section 8.7).

Recently Sternagel and Thiemann have generalized uncurrying to relative rewriting and non-applicative signatures and formalized it in the theorem prover Isabelle/HOL [164]. In particular their work comprises the certification of Theorem 8.17 (see [164, Corollary 11]) and the soundness direction of Theorem 8.62 (see [164, Theorem 15]) and Corollary 8.64 (see [164, Theorem 20]), respectively.

Aoto and Yamada [9, 10] present transformation techniques for proving termination of simply typed ATRSs. After performing η-saturation, head variables are eliminated by instantiating them with ‘template’ terms of the appropriate type. In a final step, the resulting ATRS is translated into functional form.

Example 8.76. Consider again the ATRS $\mathcal{R}$ of Example 8.8. Suppose we adopt the following type declarations: $\mathsf{0} : \mathsf{int}$, $\mathsf{s} : \mathsf{int} \to \mathsf{int}$, $\mathsf{nil} : \mathsf{list}$, $(:) : \mathsf{int} \to \mathsf{list} \to \mathsf{list}$, $\mathsf{id} : \mathsf{int} \to \mathsf{int}$, $\mathsf{add} : \mathsf{int} \to \mathsf{int} \to \mathsf{int}$, and $\mathsf{map} : (\mathsf{int} \to \mathsf{int}) \to \mathsf{list} \to \mathsf{list}$. The head variable f in the right-hand side $:\ (f\ x)\ (\mathsf{map}\ f\ y)$ has type $\mathsf{int} \to \mathsf{int}$. There are three template terms of this type: s, id, and $\mathsf{add}\ z$. Instantiating f by these three terms in $\mathcal{R}_\eta$ produces the ATRS $\mathcal{R}'$:

$$\begin{aligned}
\mathsf{id}\ x &\to x & \mathsf{map}\ f\ \mathsf{nil} &\to \mathsf{nil} \\
\mathsf{add}\ \mathsf{0} &\to \mathsf{id} & \mathsf{map}\ \mathsf{s}\ (:\ x\ y) &\to\ :\ (\mathsf{s}\ x)\ (\mathsf{map}\ \mathsf{s}\ y) \\
\mathsf{add}\ \mathsf{0}\ y &\to \mathsf{id}\ y & \mathsf{map}\ \mathsf{id}\ (:\ x\ y) &\to\ :\ (\mathsf{id}\ x)\ (\mathsf{map}\ \mathsf{id}\ y) \\
\mathsf{add}\ (\mathsf{s}\ x)\ y &\to \mathsf{s}\ (\mathsf{add}\ x\ y) & \mathsf{map}\ (\mathsf{add}\ z)\ (:\ x\ y) &\to\ :\ (\mathsf{add}\ z\ x)\ (\mathsf{map}\ (\mathsf{add}\ z)\ y)
\end{aligned}$$

The TRS $\mathcal{R}'{\downarrow_\mathcal{U}}$ is terminating because its rules are oriented from left to right by the lexicographic path order. According to the main result of [9], the *simply typed* ATRS $\mathcal{R}$ is terminating, too.

The advantage of the simply typed approach is that no uncurrying rules are necessary because the application symbol has been eliminated from $\mathcal{R}'{\downarrow_\mathcal{U}}$. This typically results in simpler termination proofs. It is worthwhile to investigate whether a version of head variable instantiation can be developed for the untyped case. We would like to stress that with the simply typed approach one obtains termination only for those terms which are simply typed. Our approach, when it works, provides termination for all terms, irrespective of *any* typing discipline. In [8] the dependency pair method is adapted to deal with simply typed ATRSs. Again, head variable instantiation plays a key role.

Applicative term rewriting is not the only model for capturing higher-order aspects. The S-expression rewrite systems of Toyama [182] have a richer structure than applicative systems, which makes proving termination often easier. The notion of strong computability is often employed for proving termination of typed lambda calculi and variations like typed rewriting calculi [32]. Recent methods (e.g. [23, 89]) use types to exploit strong computability, leading to powerful termination methods which are directly applicable to higher-order systems. In [111] strong computability is used to analyze the termination of simply typed ATRSs with the dependency pair method, and recently this approach was extended to higher-order rewrite systems [110]. Finally, in [101] many concepts from the dependency pair framework have been lifted to algebraic functional systems, a higher-order concept based on simple types and explicit β-reduction.

While in this article we used termination techniques for ordinary TRSs to show termination of uncurried TRSs, it is worth noting that there is a specialized technique for uncurried TRSs. Van Bakel and Fernández [20] introduced the class of *curryfied* TRSs and its type system for normalization. This class contains almost all uncurried TRSs. Notable exceptions are ℓ-ATRSs that contain a rule $\ell \to r$ with $\mathrm{aa}(\ell) > 0$. In [20] it is shown that a typable curryfied TRS is terminating if it satisfies the *general scheme* of [88].

We are not aware of other investigations dedicated to (derivational) complexity analysis of ATRSs.

Acknowledgments

We thank the reviewers and Georg Moser for their helpful comments and René Thiemann for suggesting that the proof of [74, Theorem 33] contains a gap.

Part V

Automation

9. CSI – A Confluence Tool

Publication Details

Harald Zankl, Bertram Felgenhauer, and Aart Middeldorp, CSI – A Confluence Tool, In Proceedings of the 23rd International Conference on Automated Deduction (CADE 2011), Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence) 6803, pp. 499–505, Springer, 2011.
doi: `10.1007/978-3-642-22438-6_38`

Abstract

This paper describes a new confluence tool for term rewrite systems. Due to its modular design, the few techniques implemented so far can be combined flexibly. Methods developed for termination analysis are adapted to prove and disprove confluence. Preliminary experimental results show the potential of our tool.

9.1. Introduction

We describe a new automatic tool for (dis)proving confluence of first-order rewrite systems (TRSs for short). Our tool is developed in Innsbruck, the city at the *c*onfluence of the two rivers Sill and Inn, and abbreviated CSI. It is available from

`http://cl-informatik.uibk.ac.at/software/csi`

and supports two new techniques for disproving confluence and very few but recent techniques for establishing confluence. CSI is open-source, equipped with a strategy language, and accessible via a simple web interface.

We assume familiarity with term rewriting and confluence [17, 172]. The remainder of this paper is organized as follows. In Section 9.2 the main techniques supported by CSI are summarized. Implementation issues are addressed in Section 9.3 and Section 9.4 concludes with preliminary experimental results.

9.2. Techniques

Besides Knuth and Bendix' criterion [100] (joinability of critical pairs for terminating systems), CSI supports the techniques described below.

Non-Confluence

To disprove confluence of a TRS $\mathcal{R}$ we consider peaks

$$t \ {}^{\leqslant m}\!\!\leftarrow t_1 \leftarrow s \rightarrow u_1 \rightarrow^{\leqslant n} u \tag{9.1}$$

such that $t_1 = s[r_1\sigma]_p \leftarrow s[\ell_1\sigma]_p = s = s[\ell_2\sigma]_q \rightarrow s[r_2\sigma]_q = u_1$ with $\ell_1 \rightarrow r_1$, $\ell_2 \rightarrow r_2 \in \mathcal{R}$, $q \leqslant p$, and $p \in \mathcal{P}\mathsf{os}(s[\ell_2]_q)$. This includes critical overlaps and some variable overlaps. In order to test non-joinability of t and u we consider ground instances of t and u. Let c_x be a fresh constant for every variable x and let $\hat{t}$ denote the result of replacing every variable in a term t by the corresponding constant. Since for terms s and w we have $s \rightarrow_{\mathcal{R}} w$ if and only if $\hat{s} \rightarrow_{\mathcal{R}} \hat{w}$, it follows that terms t and u are joinable if and only if $\hat{t}$ and $\hat{u}$ are joinable. In order to test non-joinability of $\hat{t}$ and $\hat{u}$ we overapproximate the sets of reducts for $\hat{t}$ and $\hat{u}$ and check if the intersection is empty.

The first approach is based on TCAP, which was introduced to obtain a better approximation of dependency graphs [59]. Let t be a term. The term $\mathrm{TCAP}(t)$ is inductively defined as follows. If t is a variable, $\mathrm{TCAP}(t)$ is a fresh variable. If $t = f(t_1, \ldots, t_n)$ then we let $u = f(\mathrm{TCAP}(t_1), \ldots, \mathrm{TCAP}(t_n))$ and define $\mathrm{TCAP}(t)$ to be u if u does not unify with the left-hand side of a rule in $\mathcal{R}$, and a fresh variable otherwise.

Lemma 9.1. *If $\hat{t}$ and $\hat{u}$ are joinable then* $\mathrm{TCAP}(\hat{t}\,)$ *and* $\mathrm{TCAP}(\hat{u})$ *unify.* □

In the sequel we use the result in its contrapositive form, i.e., whenever $\mathrm{TCAP}(\hat{t}\,)$ and $\mathrm{TCAP}(\hat{u})$ are not unifiable then $\hat{t}$ and $\hat{u}$ are not joinable.

The following example motivates why replacing variables by constants is beneficial.

Example 9.2. Consider the TRS $\mathcal{R}$ consisting of the rules $\mathsf{f}(x, y) \rightarrow \mathsf{g}(x)$ and $\mathsf{f}(x, y) \rightarrow \mathsf{g}(y)$. Note that $\mathrm{TCAP}(\mathsf{g}(x)) = \mathsf{g}(x')$ and $\mathrm{TCAP}(\mathsf{g}(y)) = \mathsf{g}(y')$ are unifiable but since x and y are different normal forms it is beneficial to replace them by fresh constants such that unification fails. We have $\mathrm{TCAP}(\mathsf{g}(\mathsf{c}_x)) = \mathsf{g}(\mathsf{c}_x)$ is not unifiable with $\mathsf{g}(\mathsf{c}_y) = \mathrm{TCAP}(\mathsf{g}(\mathsf{c}_y))$.

The next example illustrates Lemma 9.1.

Example 9.3. Consider the TRS $\mathcal{R} = \{\mathsf{a} \to \mathsf{f}(\mathsf{a},\mathsf{b}), \mathsf{f}(\mathsf{a},\mathsf{b}) \to \mathsf{f}(\mathsf{b},\mathsf{a})\}$ from [177] and the peak $\hat{t} = \mathsf{f}(\mathsf{f}(\mathsf{b},\mathsf{a}),\mathsf{b}) \ {}^{2}\!\!\leftarrow \mathsf{f}(\mathsf{a},\mathsf{b}) \to \mathsf{f}(\mathsf{b},\mathsf{a}) = \hat{u}$. Since $\mathrm{TCAP}(\hat{t}\,) = \mathsf{f}(\mathsf{f}(\mathsf{b},x),\mathsf{b})$ and $\mathrm{TCAP}(\hat{u}) = \mathsf{f}(\mathsf{b},y)$ are not unifiable $\mathcal{R}$ is not confluent.

We remark that Lemma 9.1 subsumes the case that t and u are different normal forms or that t and u have different root symbols which do not occur at the root of any left-hand side in $\mathcal{R}$. The latter amounts to $t(\epsilon) \neq u(\epsilon)$ and $t(\epsilon) \neq \ell(\epsilon) \neq u(\epsilon)$ for all $\ell \to r \in \mathcal{R}$, which is the test performed in [11].

Our second approach is based on tree automata. Let $\mathcal{R}$ be a left-linear TRS and L a set of ground terms. A tree automaton $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ is *compatible* [53] with $\mathcal{R}$ and L if $L \subseteq \mathcal{L}(\mathcal{A})$ and for each $\ell \to r \in \mathcal{R}$ and state substitution $\sigma\colon \mathcal{V}\mathsf{ar}(\ell) \to Q$, $r\sigma \to_{\Delta}^{*} q$ whenever $\ell\sigma \to_{\Delta}^{*} q$. The extension to arbitrary TRSs that we use in our implementation is described in [105]. Here $\mathcal{L}(\mathcal{A})$ is the language accepted by a tree automaton $\mathcal{A}$. In the following $\to_{\mathcal{R}}^{*}(L)$ denotes the set $\{t \mid s \to_{\mathcal{R}}^{*} t \text{ for some } s \in L\}$.

Theorem 9.4. *Let $\mathcal{R}$ be a TRS, $\mathcal{A}$ a tree automaton, and L a set of ground terms. If $\mathcal{A}$ is compatible with $\mathcal{R}$ and L then $\to_{\mathcal{R}}^{*}(L) \subseteq \mathcal{L}(\mathcal{A})$.* $\square$

We overapproximate the sets of terms reachable from $\hat{t}$ and $\hat{u}$ using tree automata, i.e., we construct tree automata $\mathcal{A}_1$ and $\mathcal{A}_2$ (by *tree automata completion* [105]) such that $\to_{\mathcal{R}}^{*}(\{\hat{t}\,\}) \subseteq \mathcal{L}(\mathcal{A}_1)$ and $\to_{\mathcal{R}}^{*}(\{\hat{u}\}) \subseteq \mathcal{L}(\mathcal{A}_2)$ and conclude non-joinability of $\hat{t}$ and $\hat{u}$ if $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2) = \varnothing$, which is decidable.

Example 9.5. Consider Lévy's TRS $\mathcal{R}$ from [84]

$$\begin{array}{llll}
\mathsf{f}(\mathsf{a},\mathsf{a}) \to \mathsf{g}(\mathsf{b},\mathsf{b}) & \mathsf{a} \to \mathsf{a}' & \mathsf{f}(\mathsf{a}',x) \to \mathsf{f}(x,x) & \mathsf{f}(x,\mathsf{a}') \to \mathsf{f}(x,x) \\
\mathsf{g}(\mathsf{b},\mathsf{b}) \to \mathsf{f}(\mathsf{a},\mathsf{a}) & \mathsf{b} \to \mathsf{b}' & \mathsf{g}(\mathsf{b}',x) \to \mathsf{g}(x,x) & \mathsf{g}(x,\mathsf{b}') \to \mathsf{g}(x,x)
\end{array}$$

and $\hat{t} = \mathsf{f}(\mathsf{a}',\mathsf{a}') \ {}^{*}\!\!\leftarrow \mathsf{f}(\mathsf{a},\mathsf{a}) \to^{*} \mathsf{g}(\mathsf{b}',\mathsf{b}') = \hat{u}$. We have $\to_{\mathcal{R}}^{*}(\{\hat{t}\,\}) = \{\hat{t}\,\}$ and $\to_{\mathcal{R}}^{*}(\{\hat{u}\}) = \{\hat{u}\}$. Consequently $\to_{\mathcal{R}}^{*}(\{\hat{t}\,\}) \cap \to_{\mathcal{R}}^{*}(\{\hat{u}\}) = \varnothing$ and hence we conclude non-joinability of $\hat{t}$ and $\hat{u}$ which yields the non-confluence of $\mathcal{R}$. Note that $\mathrm{TCAP}(\hat{t}\,) = x$ and $\mathrm{TCAP}(\hat{u}) = y$ unify.

Order-Sorted Decomposition

Next we focus on a criterion that allows to decompose a TRS $\mathcal{R}$ into TRSs $\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n$ where $\mathcal{R}$ is confluent whenever all $\mathcal{R}_i$ are confluent. Order-sorted decomposition is a generalization of persistent decomposition [11, Definition 2] to ordered sorts. It is based on a result in [47].

Theorem 9.6. *Let $\mathcal{R}$ be a TRS and $\langle \mathcal{F}, \mathcal{V} \rangle$ an order-sorted signature with sorts $\mathcal{S}$ equipped with a strict order $\succ$. Assume that the following conditions hold:*

1. *$\mathcal{R}$ is compatible with $\mathcal{S}$, i.e., rules $\ell \to r \in \mathcal{R}$ are well-sorted, with variables bound strictly in ℓ and the sort of ℓ is $\succeq$ that of r.*

2. *If $\mathcal{R}$ is non-left-linear and duplicating then for $\ell \to r \in \mathcal{R}$, variables in r are bound strictly as well. Furthermore, if $r \in \mathcal{V}$ the sort of r must be maximal.*

If $\mathcal{R}$ is confluent on well-sorted terms then $\mathcal{R}$ is confluent on all terms. □

Each sort attachment satisfying the conditions of Theorem 9.6 gives rise to a decomposition of $\mathcal{R}$ into $\max\{\mathcal{R} \cap \mathcal{T}_{\trianglelefteq\alpha}(\mathcal{F},\mathcal{V}) \times \mathcal{T}_{\trianglelefteq\alpha}(\mathcal{F},\mathcal{V}) \mid \alpha \in \mathcal{S}\}$, where $\mathcal{T}_{\trianglelefteq\alpha}(\mathcal{F},\mathcal{V})$ denotes the subterms of terms of sort $\preceq \alpha$. Note that we can replace proper subterms $t|_p : \beta$ by any other terms with sort $\preceq \beta$. Hence $\mathcal{T}_{\trianglelefteq\alpha}(\mathcal{F},\mathcal{V})$ is closed under adding terms of sort $\preceq$ that of any terms in $\mathcal{T}_{\trianglelefteq\alpha}(\mathcal{F},\mathcal{V})$. As in the many-sorted persistence case, we can find a most general ordered sort attachment consistent with any given TRS efficiently. Start by assigning sort variables to the argument and result types of all function symbols and to the variables occurring in the rules, after renaming them to ensure that no two rules share any variables. The consistency conditions, except for the maximality condition for collapsing rules, translate to inequalities $\alpha \succeq \beta$ between these type variables. To solve a system of such constraints, consider the graph with sort variables as nodes and edges from α to β whenever there is a constraint $\alpha \succeq \beta$. Then assign a distinct sort to the variables of each strongly connected component of the graph, ordered strictly by the edges between the components. A maximality constraint on β can be enforced in a second pass that equates α and β whenever $\alpha \succ \beta$. This process is demonstrated in the example below.

Example 9.7. Consider the TRS

$$1\colon \mathsf{f}(x,\mathsf{A}) \to \mathsf{G}(x) \qquad 2\colon \mathsf{f}(x,\mathsf{G}(x)) \to \mathsf{B} \qquad 3\colon \mathsf{G}(\mathsf{C}) \to \mathsf{C} \qquad 4\colon \mathsf{F}(x) \to \mathsf{F}(\mathsf{G}(x))$$

We start by assigning variables to the various sorts. Let x_i be the sort of x in rule i. Furthermore let $\mathsf{A} : A$, $\mathsf{B} : B$, $\mathsf{C} : C$, $\mathsf{f} : f_1 \times f_2 \to f$, $\mathsf{F} : F_1 \to F$ and $\mathsf{G} : G_1 \to G$. By well-sortedness we get constraints $f_1 \succeq x_1$, $f_2 \succeq A$ from the left-hand side of the first rule. By strictness of left-hand sides, we require that $x_1 \succeq f_1$. We get similar constraints from the other rules, noting that since the TRS is non-duplicating, we do not have strictness constraints on the right-hand sides. By relating the sorts of left-hand sides and right-hand sides, we obtain further constraints, namely $f \succeq G$, $f \succeq B$, $G \succeq C$ and $F \succeq F$. Denoting $\alpha \succeq \beta$ by an edge $\alpha \to \beta$, we obtain the following graph:

$$\begin{tikzcd}
 & x_2 \arrow[dl, bend left] \arrow[dr, bend left] & & & f_2 \arrow[r] \arrow[dr] & A & & \\
f_1 \arrow[ur, bend left] \arrow[dr, bend left] & & G_1 \arrow[ul, bend left] \arrow[dl] \arrow[r] \arrow[rrrr, bend right] & x_4 \arrow[r, bend left] & F_1 \arrow[l, bend left] \arrow[r] & G \arrow[r] & C & F \arrow[loop above] \\
 & x_1 \arrow[ul, bend left] & & & f \arrow[ur] \arrow[r] & B & &
\end{tikzcd}$$

The strongly connected components are $8 = \{f_2\}$, $7 = \{A\}$, $6 = \{f\}$, $5 = \{B\}$, $4 = \{G_1, x_1, f_1, x_2\}$, $3 = \{F_1, x_4\}$, $2 = \{G\}$, $1 = \{C\}$, and $0 = \{F\}$, ordered by $8 \succ 7, 2$, $6 \succ 5, 2$, and $4 \succ 3 \succ 2 \succ 1$. The resulting signature is $\mathsf{A} : 7$, $\mathsf{B} : 5$, $\mathsf{C} : 1$, $\mathsf{f} : 4 \times 8 \to 6$, $\mathsf{F} : 3 \to 0$, and $\mathsf{G} : 4 \to 2$ giving rise to the decomposition into the TRSs $\{(1), (2), (3)\}$ and $\{(3), (4)\}$.

If we required maximality of the sort $2 = \{G\}$, then we would equate 2 and 3 (since $3 \succ 2$), and further with 4 (as $4 \succ 3$), 6 (as $6 \succ 2$) and 8 (as $8 \succ 2$), obtaining $8' = \{G, F_1, x_4, G_1, x_1, f_1, x_2, f, f_2\}$, ordered by $8' \succ 7, 5, 1$. The resulting signature is $\mathsf{A} : 7$, $\mathsf{B} : 5$, $\mathsf{C} : 1$, $\mathsf{f} : 8' \times 8' \to 8'$, $\mathsf{F} : 8' \to 0$, and $\mathsf{G} : 8' \to 8'$. Note that here no (non-trivial) decomposition is possible.

Decreasing Diagrams

The decreasing diagrams technique [138, 139] is a complete method for confluence on countable abstract rewrite systems. The next result employs decreasing diagrams for TRSs and follows immediately from [201, Corollary 3.16]. It also serves to demonstrate the design of our tool which typically implements one criterion by combining smaller pieces via a strategy language (cf. Section 9.3). Here $\mathcal{R}_{\mathsf{d}}$ ($\mathcal{R}_{\mathsf{nd}}$) denotes the (non)duplicating rules in a TRS $\mathcal{R}$.

Theorem 9.8. *A left-linear TRS $\mathcal{R}$ is confluent if $\mathcal{R}_{\mathsf{d}}$ is terminating relative to $\mathcal{R}_{\mathsf{nd}}$ and all critical peaks of $\mathcal{R}$ are decreasing with respect to the rule labeling.* □

To exploit this theorem we need to solve relative termination problems. In [201] we show that relative termination techniques can additionally be used for labeling diagrams (also in combination with the rule labeling).

9.3. Implementation

CSI is implemented based on the open source termination tool $\mathsf{T_TT_2}$ [107] and written in OCaml. As explained in the preceding section, several criteria from termination analysis are useful for confluence. Our tool is based on few techniques, but a *strategy language* (akin to the one to control $\mathsf{T_TT_2}$) allows one to combine different criteria flexibly and to obtain a powerful tool. For a grammar of this strategy language, consult [107] or pass the option `-h` to the tool.

Automatic Mode

In its automatic mode CSI executes the strategy

```
(KB || NOTCR || (((CLOSED || DD) | add)2*)!  || sorted -order)*
```

Here identifiers in capital letters abbreviate combinations of techniques. We skip details for brevity. The command `KB` refers to Knuth and Bendix' criterion [100], `NOTCR` is a test for non-confluence as described in Section 9.2, and `sorted -order` aims for an order-sorted decomposition (cf. Section 9.2 and [47]). The operator `||` executes all those criteria in parallel—to make use of modern multi-core architectures—and the first substrategy that succeeds is used to make progress on the given problem. Since a successful call to `sorted -order` returns a list of problems, the trailing `*` ensures that the above strategy is iterated on all subproblems until no further progress can be achieved. Finally we describe the part that is still missing. `CLOSED` tests whether the critical pairs of a left-linear system are development closed [143] and `DD` implements decreasing diagrams (Section 9.2 and [201]). If these methods do not succeed the alternative `|` executes `add`, which adds new rules that might enable the other criteria to succeed ([201, Lemma 4.3 and Example 4.4]) while the postfix `2*` executes the strategy inside parentheses at most two times, i.e., `CLOSED || DD` is run again, if some rules have been added. The outermost `!` ensures that the strategy inside only succeeds if confluence could be (dis)proved.

Strategy Language

We elaborate on the strategy language to show the flexibility and modularity of our tool. In the strategy `nonconfluence -steps 2 -tcap` the flag `-tcap` tests non-joinability of terms with TCAP, as outlined in Section 9.2. With `-steps` values for m and n in the peak (9.1) on page 234 are set.

The criterion from Theorem 9.8 allows one to use the decreasing diagrams technique, provided some precondition is satisfied. To this end the composition operator `;` is employed, where `A;B` executes `B` only if `A` succeeds. Given an input TRS $\mathcal{R}$, in the strategy

```
cr -dup;matrix -dim 2*;rule_labeling;decreasing
```

the expression `cr -dup` generates the relative TRS $\mathcal{R}_{\mathsf{d}}/\mathcal{R}_{\mathsf{nd}}$, termination of which is attempted with matrix interpretations of dimension 2. If this succeeds the critical

	CSI	ACP	$\sum$
CR	61	64	67
not CR	20	18	21

(a) 106 TRSs.

	CSI	ACP	$\sum$
CR	43	42	43
not CR	47	47	47

(b) 99 TRSs.

	CSI	ACP	$\sum$
CR	6	2	6
not CR	2	2	2

(c) 9 TRSs.

Table 9.1.: Experiments.

diagrams are labeled with the rule labeling [138], before a test for decreasingness is performed. We note that the strategy language allows to label incrementally combining different (relative termination) criteria [201]. Here a critical diagram is a critical peak $t \leftarrow s \rightarrow u$ together with joining sequences $t \rightarrow^* v \; {}^*\!\!\leftarrow u$. In the implementation for every critical peak we consider all joining sequences $t \rightarrow^{\leqslant n} \cdot {}^{\leqslant n}\!\!\leftarrow u$ for which there is no smaller n that admits a common reduct.

9.4. Evaluation

For experiments[1] we used the collection from [4] which consists of 106 TRSs from the rewriting literature dealing with confluence (Table 9.1(a)), the 99 TRSs from the 2010 edition of the termination competition which are non-terminating or not known to be terminating (Table 9.1(b)), and the TRSs from [47, 201] (Table 9.1(c)). The time limit of 60 seconds was hardly ever reached.

In Table 9.1 we compare the automatic mode of our tool with ACP [11], a confluence prover that implements various techniques from the literature. On the testbench in Table 9.1(a) ACP can show more systems confluent than CSI but our tool is superior for non-confluence. The last column shows that on this testbench no tool subsumes the other one which is not the case for Tables 9.1(b)(c).

Table 9.2 elaborates on the differences of the tools' performance. Here a $\times$ indicates that the corresponding tool failed to analyze the status of the given TRS while a ✓ means that confluence (or non-confluence) could be determined. The numbers in parentheses refer to the time spent on this problem in seconds. The different blocks in Table 9.2 correspond to the different testbeds employed.

The rewriting toolkit C*i*ME3 [37] also supports confluence analysis as one of its many features. This tool exploits Newman's Lemma, i.e., for a terminating TRS confluence coincides with local confluence (the latter can then effectively be checked [100]). While this test is also contained in ACP and CSI, the novel feature of C*i*ME3 is that it can (automatically) certify such confluence proofs in the proof assistant Coq.

[1]Details are available from the CSI website.

system	CSI	ACP	status
BN98/ex6.5f	$\times(\infty)$	✓(0.4)	$\neg$CR
Der97/p204	$\times(6.6)$	✓(0.1)	CR
GL06/ex3	✓(0.6)	$\times(0.2)$	CR
GOO96/R2p	$\times(6.3)$	✓(0.1)	CR
Gra96caap/ex2	$\times(3.0)$	✓(0.1)	CR
OO03/ex1	✓(4.0)	$\times(7.0)$	CR
OO03/ex2	$\times(6.3)$	✓(0.1)	CR
Ohl94caap/ex5.12	✓(0.4)	$\times(0.1)$	$\neg$CR
TO01/ex6	$\times(6.3)$	✓(0.1)	CR

system	CSI	ACP	status
Tiw02/ex1	✓(0.3)	$\times(0.1)$	$\neg$CR
Toy98/ex1	$\times(6.1)$	✓(0.1)	CR
standards/AC	✓(2.7)	$\times(0.1)$	CR
standards/add_C	✓(4.0)	$\times(0.1)$	CR
Transformed_CS[a]	✓(4.6)	$\times(\infty)$	CR
ZFM11/ex1.1	✓(4.9)	$\times(7.0)$	CR
ZFM11/ex3.18	✓(1.0)	$\times(0.1)$	CR
ZFM11/ex3.20	✓(2.1)	$\times(0.5)$	CR
ZFM11/ex4.1	✓(0.9)	$\times(0.1)$	CR

[a]Transformed_CSR_04_PALINDROME_nokinds-noand_L

Table 9.2.: Performance difference on the three testbenches.

To conclude we stress the main attractions of CSI: To the best of our knowledge it is the only tool that implements order-sorted decomposition of rewrite systems, it employs powerful criteria for disproving confluence, and due to the modular design it allows to combine different labeling functions for the decreasing diagrams technique.

10. KBCV – Knuth-Bendix Completion Visualizer

Publication Details

Thomas Sternagel and Harald Zankl, KBCV – Knuth-Bendix Completion Visualizer, In Proceedings of the 6th International Joint Conference on Automated Reasoning (IJCAR 2012), Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence) 7364, pp. 530–536, Springer, 2012.
doi: `10.1007/978-3-642-31365-3_41`

Abstract

This paper describes a tool for Knuth-Bendix completion. In its interactive mode the user only has to select the orientation of equations into rewrite rules; all other computations (including necessary termination checks) are performed internally. Apart from the interactive mode, the tool also provides a fully automatic mode. Moreover, the generation of (dis)proofs in equational logic is supported. Finally, the tool outputs proofs in a certifiable format.

10.1. Introduction

The *Knuth-Bendix Completion Visualizer* (KBCV) is an interactive/automatic tool for Knuth-Bendix completion and equational logic proofs. This paper describes KBCV version 1.7, which features a command-line and a graphical user interface as well as a Java-applet version. The tool is available under the *GNU Lesser General Public License 3* at

`http://cl-informatik.uibk.ac.at/software/kbcv`

Completion is a procedure which takes as input a finite set of equations E (and nowadays optionally a reduction order $>$) and attempts to construct a terminating and confluent term rewrite system (TRS) R which is equivalent to E, i.e., their equational theories coincide. In case the completion procedure succeeds, R

$$\begin{array}{llll}
\text{DEDUCE} \; \dfrac{(E,R)}{(E \cup \{s \approx t\}, R)} \text{ if } s \;{}_R{\leftarrow}\; u \rightarrow_R t & & \text{ORIENT} \; \dfrac{(E \cup \{s \mathrel{\dot{\approx}} t\}, R)}{(E, R \cup \{s \rightarrow t\})} \text{ if } s > t \\[2ex]
\text{COMPOSE} \; \dfrac{(E, R \cup \{s \rightarrow t\})}{(E, R \cup \{s \rightarrow u\})} \text{ if } t \rightarrow_R u & & \text{DELETE} \; \dfrac{(E \cup \{s \approx s\}, R)}{(E,R)} \\[2ex]
\text{COLLAPSE} \; \dfrac{(E, R \cup \{s \rightarrow t\})}{(E \cup \{u = t\}, R)} \text{ if } s \overset{\sqsupset}{\rightarrow}_R u & & \text{SIMPLIFY} \; \dfrac{(E \cup \{s \mathrel{\dot{\approx}} t\}, R)}{(E \cup \{u \mathrel{\dot{\approx}} t\}, R)} \text{ if } s \rightarrow_R u
\end{array}$$

Figure 10.1.: Inference rules for completion with a fixed reduction order ($\mathcal{C}$).

represents a decision procedure for the word problem of E. Now two terms are equivalent with respect to E if and only if they reduce to the same normal form with respect to R.

The computation is done by generating a finite sequence of intermediate TRSs which constitute approximations of the equational theory of E. Following Bachmair and Dershowitz [18] the completion procedure can be modeled as an inference system like system $\mathcal{C}$ in Figure 10.1. The inference rules work on pairs (E, R) where E is a finite set of equations and R is a finite set of rewrite rules. The goal is to transform an initial pair $(E, \varnothing)$ into a pair $(\varnothing, R)$ such that R is terminating, confluent and equivalent to E. In our setting a completion procedure based on these rules may succeed (find R after finitely many steps), loop, or fail. In Figure 10.1 a reduction order $>$ is provided as part of the input. We use $s \overset{\sqsupset}{\rightarrow}_R u$ to express that s is reduced by a rule $\ell \rightarrow r \in R$ such that ℓ cannot be reduced by another rule $s \rightarrow t \in R$. The notation $s \mathrel{\dot{\approx}} t$ denotes either of $s \approx t$ and $t \approx s$.

Writing $(E, R) \vdash_{\mathcal{C}} (E', R')$ to indicate that (E', R') is obtained from (E, R) by one of the inference rules of system $\mathcal{C}$ we define a *completion procedure*:

Definition 10.1. A *completion procedure* is a program that accepts as input a finite set of equations E_0 (together with a reduction order $>$) and uses the inference rules of Figure 10.1 to construct a sequence

$$(E_0, \varnothing) \vdash_{\mathcal{C}} (E_1, R_1) \vdash_{\mathcal{C}} (E_2, R_2) \vdash_{\mathcal{C}} (E_3, R_3) \vdash_{\mathcal{C}} \cdots$$

Such a sequence is called a *run* of the completion procedure on input E_0 and $>$. A finite run $(E_0, \varnothing) \vdash_{\mathcal{C}}^n (\varnothing, R_n)$ is *successful* if R_n is locally confluent.

The following result follows from [17, Theorem 7.2.8] specialized to finite runs.

Lemma 10.2. *Let $(E_0, \varnothing) \vdash_{\mathcal{C}}^n (\varnothing, R_n)$ be a successful run of completion. Then R_n is terminating, confluent, and equivalent to E_0.* $\square$

In the sequel we assume familiarity with term rewriting, equational logic, and completion [17]. The remainder of this paper is organized as follows. In the next section the main features of KBCV are presented before Section 10.3 addresses implementation issues and experimental results. Section 10.4 concludes.

10.2. Features

KBCV offers two modes for completion, namely the Normal Mode (Section 10.2.1) and the Expert Mode (Section 10.2.2). In the GUI the user can change the mode via the menu entry *View* at any time. Irregardless of the chosen mode, termination checks are performed automatically, following the recent approach from [189]. By default, an incremental LPO is constructed and maintained by the tool but also external termination tools are supported (this option is not available in the applet version). For convenience KBCV stores a history that allows to step backwards (and forwards again) in interactive completion proofs. Apart from completion proofs, the tool can generate proofs in equational logic (Section 10.2.3) and produces output in a certifiable format.

10.2.1. Normal Mode

In *normal mode* the user can switch between *efficient* and *simple* completion. The efficient procedure executes all inference rules from Figure 10.1 in a fixed order, while the simple procedure considers a subset only.

Efficient Completion

The *efficient completion procedure* (following Huet [87], see Figure 10.2) takes a set of equations E as input and has three possible outcomes: It may terminate successfully, it may loop indefinitely, or it may fail because an equation could not be oriented into a rewrite rule.

While $E \neq \varnothing$ the user chooses an equation $s \approx t$ from E. The terms in this equation are simplified to normal form by using SIMPLIFY exhaustively. In the next step the equation is deleted if it was trivial and if so the next iteration of the loop starts. Otherwise (following the transition labeled NO) the user suggests the orientation of the equation into a rule and ORIENT performs the necessary termination check. Here the procedure might fail if the equation cannot be oriented (in either direction) with the used termination technique. But if the orientation succeeds the inferred rule is used to reduce the right-hand sides of (other) rules to normal form (COMPOSE) while COLLAPSE rewrites the left-hand sides of rules, which transforms rules into equations that go back to E. In this way the set of

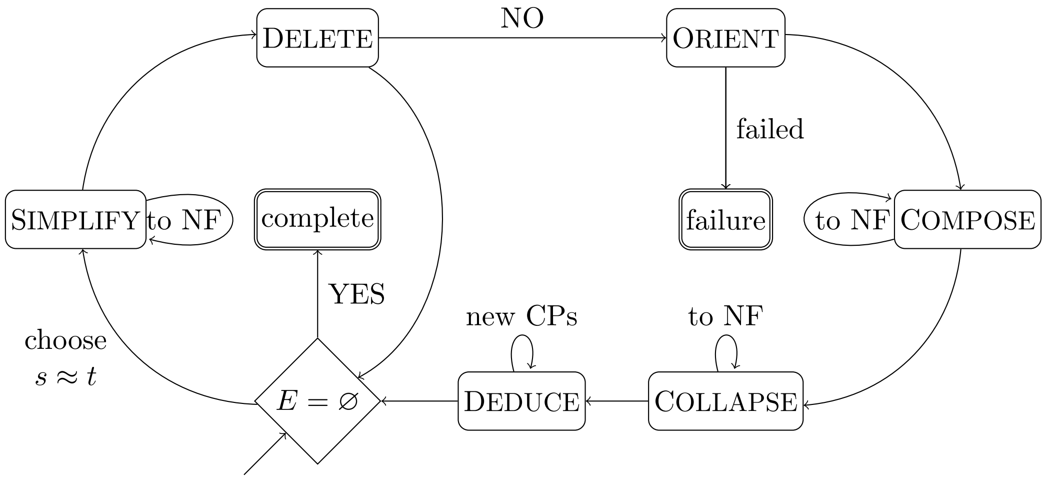


Figure 10.2.: Flow chart for the efficient completion procedure.

rules in R is kept as small as possible at all times. Afterwards DEDUCE is used to compute (all) critical pairs (between the new rule and the old rules and between the new rule and itself). If still $E \neq \varnothing$ the next iteration of the loop begins and otherwise the procedure terminates successfully yielding the terminating and confluent (complete) TRS R equivalent to the input system E.

Simple Completion

The simple procedure (following the *basic* completion procedure [17, Figure 7.1]) makes no use of COMPOSE and COLLAPSE, which means that the inference rule DEDUCE immediately follows ORIENT. Hence although correct, this procedure is not particularly efficient.

10.2.2. Expert Mode

Inference System

In the *expert mode* the user can select the equations and rewrite rules on which the desired inference rules from Figure 10.1 should be applied on. If no equations/rules are selected explicitly then all equations/rules are considered. For efficiency reasons DEDUCE does only add critical pairs emerging from overlaps that have not yet been considered. KBCV notifies the user if a complete R equivalent to the input E is obtained.

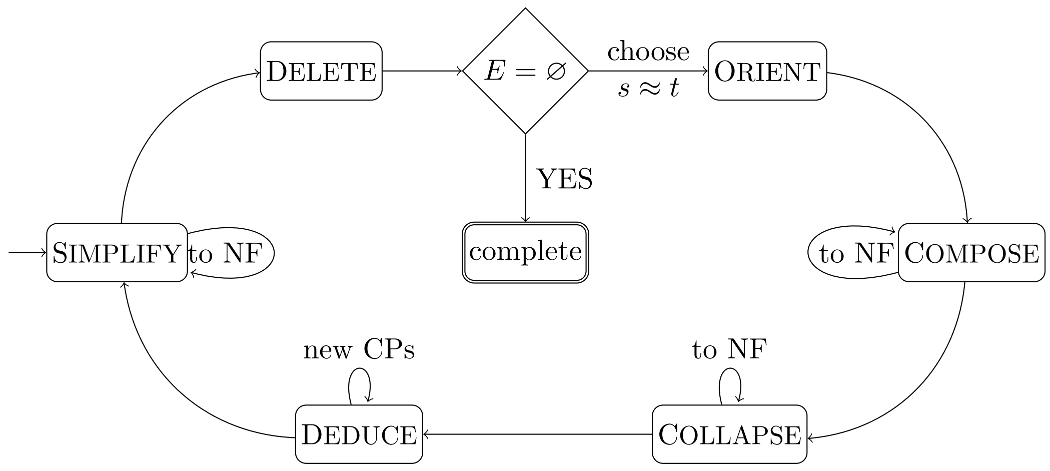


Figure 10.3.: Flow chart for the automatic mode.

Automatic Mode

At any stage of the process the user can press the button $\boxed{\textit{Completion}}$ which triggers the automatic mode of KBCV where it applies the inference rules according to the loop in Figure 10.3. Pressing the button again (during the completion attempt) stops the automatic mode and shows the current state (of the selected thread, see below). It is also possible to specify an upper limit on the loops performed in Figure 10.3 (*Settings* → *Automatic Completion*). This is especially useful to step through a completion proof with limit 1.

In Figure 10.3 the rules SIMPLIFY and DELETE operate on all equations and are applied exhaustively. If $E = \varnothing$ then R is locally confluent (since the previous DEDUCE considered all remaining critical pairs) and the procedure successfully terminates. Note that in contrast to the completion procedure from Figure 10.2 the automatic mode postpones the choice of the equation $s \approx t$. Hence KBCV can choose an equation of minimal length *after* simplification (which is typically beneficial for the course of completion) for the rule ORIENT. To maximize power, KBCV executes two threads in parallel which have different behavior for ORIENT. The first thread prefers to orient equations from left-to-right and if this is not possible it tries a right-to-left orientation (the second thread behaves dually). If this also fails another equation is selected in the next turn. (Note that it is possible that some later equation can be oriented which then simplifies the problematic equation such that it can be oriented or deleted.) A thread fails if no equation in E can be oriented in the ORIENT step.

10.2.3. Equational Logic and Certification

Since KBCV stores how rules have been deduced from equations [167], in command-line mode the command `showh` lists how rules/equations have been derived and allows to trace back the completion steps that gave rise to a rule/equation. The same mechanism facilitates KBCV to automatically transform a join $s \to_R^* \cdot {}_R^*\!\leftarrow t$ with respect to the current system R (which need not be complete yet) into a conversion with respect to the input system E, i.e., $s \leftrightarrow_E^* t$, and further into equational proofs with respect to E (*File* → *Equational Proof*).

If E could be completed into a TRS R, the recent work in [167] allows KBCV to export proof certificates (*File* → *Export Equational Proof* and *File* → *Export Completion Proof*) in CPF, a certification proof format for rewriting.[1] These proof certificates can be certified by CeTA [176], i.e., checked by a trustable program generated from the theorem prover Isabelle. Apart from the input system E and the completed TRS R such a certificate must also contain a proof that E and R are equivalent, e.g., by giving an explicit conversion $\ell \leftrightarrow_E^* r$ for each $\ell \to r \in R$.

10.3. Implementation and Experiments

KBCV is implemented in Scala,[2] an object-functional programming language which compiles to Java Byte Code. For this reason KBCV is portable and runs on Windows and Linux machines. We developed a term library in Scala (`scala-termlib`, available from KBCV's homepage) of approximately 1700 lines of code. KBCV builds upon this library and has an additional 4500 lines of code.

Besides the stand-alone version of KBCV there also is a Java-Applet version available online. The stand-alone version has three different modes: The text mode where one can interact with KBCV via the console, the graphic mode using a graphical user interface implemented in `java.swing`, and the hybrid mode where the text mode and the graphic mode are combined. In text mode typing `help` yields a list of all available commands, whereas in graphic (hybrid) mode or the Java-Applet you can select *Help* → *User Manual* to get a description of the user interface.

The stand-alone version of KBCV is able to call third party termination checkers whereas the Java-Applet version is limited to the internal LPO for termination proofs.

As input KBCV supports the XML-format for TRSs[3] and also a subset of the

[1] `http://cl-informatik.uibk.ac.at/software/cpf`
[2] `http://www.scala-lang.org/`
[3] `http://www.termination-portal.org/wiki/XTC_Format_Specification`

older TRS-format.[4] (Only one `VAR` and one `RULES` section are allowed in this order. No theory or strategy annotations are supported.) In both cases rules are interpreted as equations.

In addition KBCV supports another file format for the export and import of command logs to save and load user specific settings of KBCV. This format lists all executed commands within KBCV in a human readable form, like:

```
load ../examples/gene.trs
orient > 1
simplify
...
```

Saving the current command log is done via (*File* $\to$ *Export Command Log*) and loading works alike (*File* $\to$ *Load Command Log*). Command logs saved in the file `.kbcvinit` are loaded automatically on program startup.

Although the major attraction of KBCV clearly is its interactive mode, in the sequel experimental results demonstrate that its automatic mode can compete with state-of-the-art completion tools. To this end we extend [98, Table 1] with data for KBCV (considering 115 problems from the distribution of MKBTT).[5] Hence Table 10.1[6] compares KBCV with MKBTT [154], Maxcomp [98], and Slothrop [189]. Within a time limit of 300 seconds, KBCV completes 85 systems using its internal LPO and succeeds on an additional system when calling the external termination tool $\mathsf{T_TT_2}$ [107]. Slothrop [189] was the first tool to construct reduction orders on the fly using external termination tools and obtains 71 completed systems. MKBTT [154] adopts this approach, but additionally features *multi-completion*, i.e., considering multiple reduction orderings at the same time. Finally, the strategy of Maxcomp [98] is to handle all suitable candidate TRSs (terminating and maximal) at once. Maxcomp can complete 86 systems with LPO but since the search for maximal TRSs is coupled with the search for the reduction order this approach does not support external termination tools. All tools together can complete 95 systems. The lower part of Table 10.1 shows those systems which only one tool could complete within the given time limit. Here KBCV completed two systems where all other tools failed.

All 86 completion proofs found by KBCV (Table 10.1) could be certified by CeTA [176] (see Section 10.2.3). Since recently, MKBTT can also provide proof certificates but currently neither Maxcomp nor Slothrop support them.

[4]`http://www.lri.fr/~marche/tpdb/format.html`
[5]`http://cl-informatik.uibk.ac.at/software/mkbtt`
[6]KBCV data available at `http://cl-informatik.uibk.ac.at/software/kbcv/experiments/12ijcar`.

	LPO			termination tool		
	KBCV	MKBTT	Maxcomp	KBCV	MKBTT	Slothrop
completed	85	70	86	86	81	71
`LS94_P1`	✓			✓		
`SK90_3.26`	✓			✓		
`Slothrop_cge`					✓	
`Slothrop_equiv_proof_or`					✓	
`WS06_proofreduction`					✓	

Table 10.1.: Experimental results on 115 systems.

10.4. Conclusion

In this paper we have presented KBCV, a tool that supports interactive completion proofs. Hence it is of particular interest for students and users that are exposed to the area of completion for the first time or want to follow a completion proof step by step. Its automatic mode can compete with modern completion tools (Slothrop, MKBTT, Maxcomp) that use more advanced techniques for completion (completion with external termination tools, multi-completion, maximal-completion) but lack an interactive mode. Since KBCV records how rules have been derived, it can produce certifiable output of completion proofs and can construct (dis)proofs in equational logic.

Unfailing completion [19] is a variant of Knuth-Bendix completion, which sacrifices confluence for ground confluence. One possible direction for future work would be to integrate unfailing completion into KBCV. Another issue is to gain further efficiency by a smart design of the employed data structure [116].

Acknowledgments

We thank Christian Sternagel and the reviewers for helpful comments and suggestions concerning efficiency.

Bibliography

[1] Adian, S.: Upper bound on the derivational complexity in some word rewriting system. Doklady Mathemathics 80(2), 679–683 (2009)

[2] Alarcón, B., Lucas, S., Navarro-Marset, R.: Proving termination with matrix interpretations over the reals. In: Proc. 10th International Workshop on Termination (WST 2009). pp. 12–15 (2009)

[3] Albert, E., Arenas, P., Genaim, S., Puebla, G., Zanardini, D.: Cost analysis of object-oriented bytecode programs. Theoretical Computer Science 413(1), 142–159 (2012)

[4] Aoto, T.: Automated confluence proof by decreasing diagrams based on rule-labelling. In: Proc. 21st International Conference on Rewriting Techniques and Applications (RTA 2010). Leibniz International Proceedings in Informatics, vol. 6, pp. 7–16 (2010)

[5] Aoto, T., Toyama, Y.: Persistency of confluence. Journal of Universal Computer Science 3(11), 1134–1147 (1997)

[6] Aoto, T., Toyama, Y.: A reduction-preserving completion for proving confluence of non-terminating term rewriting systems. Logical Methods in Computer Science 8(1:31), 1–29 (2012)

[7] Aoto, T., Toyama, Y., Uchida, K.: Proving confluence of term rewriting systems via persistency and decreasing diagrams. In: Proc. Joint 25th International Conference on Rewriting Techniques and Applications and 12th International Conference on Typed Lambda Calculi and Applications (RTATLCA 2014). Lecture Notes in Computer Science (Advanced Research in Computing and Software Science), vol. 8560, pp. 46–60 (2014)

[8] Aoto, T., Yamada, T.: Dependency pairs for simply typed term rewriting. In: Proc. 16th International Conference on Rewriting Techniques and Applications (RTA 2005). Lecture Notes in Computer Science, vol. 3467, pp. 120–134 (2005)

[9] Aoto, T., Yamada, T.: Termination of simply-typed applicative term rewriting systems. In: Proc. 2nd International Workshop on Higher-Order Rewriting (HOR 2004). Technical Report AIB-2004-03, RWTH Aachen. pp. 61–65 (2004)

[10] Aoto, T., Yamada, T.: Termination of simply typed term rewriting by translation and labelling. In: Proc. 14th International Conference on Rewriting Techniques and Applications (RTA 2003). Lecture Notes in Computer Science, vol. 2706, pp. 380–394 (2003)

[11] Aoto, T., Yoshida, J., Toyama, Y.: Proving confluence of term rewriting systems automatically. In: Proc. 20th International Conference on Rewriting Techniques and Applications (RTA 2009). Lecture Notes in Computer Science, vol. 5595, pp. 93–102 (2009)

[12] Aoto, T., Hirokawa, N., Zankl, H.: Confluence Competition (CoCo) (2014). `http://coco.nue.riec.tohoku.ac.jp`

[13] Arts, T., Giesl, J.: Termination of term rewriting using dependency pairs. Theoretical Computer Science 236(1-2), 133–178 (2000)

[14] Avanzini, M., Moser, G.: A combination framework for complexity. In: Proc. 24th International Conference on Rewriting Techniques and Applications (RTA 2013). Leibniz International Proceedings in Informatics, vol. 21, pp. 55–70 (2013)

[15] Avanzini, M., Moser, G.: Polynomial path orders. Logical Methods in Computer Science 9(4:9), 1–42 (2013)

[16] Avanzini, M., Moser, G., Schnabl, A.: Automated implicit computational complexity analysis (system description). In: Proc. 4th International Joint Conference on Automated Reasoning (IJCAR 2008). Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 5195, pp. 132–138 (2008)

[17] Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press, Cambridge (1998)

[18] Bachmair, L., Dershowitz, N.: Equational inference, canonical proofs, and proof orderings. Journal of the ACM 41(2), 236–276 (1994)

[19] Bachmair, L., Dershowitz, N., Plaisted, D.: Completion without failure. In: Resolution of Equations in Algebraic Structures, Vol. 2: Rewriting Techniques. 1–30 (1989)

[20] van Bakel, S., Fernández, M.: Normalization results for typeable rewrite systems. Information and Computation 133(2), 73–116 (1997)

[21] Beklemishev, L.: Representing worms as term rewriting systems. In: Proc. Mini-Workshop: Logic, Combinatorics and Independence Results. Oberwolfach Reports, vol. 3(4), pp. 3093–3095. European Mathematical Society (2006)

[22] Bezem, M., Klop, J., van Oostrom, V.: Diagram techniques for confluence. Information and Computation 141(2), 172–204 (1998)

[23] Blanqui, F., Jouannaud, J.P., Rubio, A.: HORPO with computability closure: A reconstruction. In: Proc. 14th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2007). Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 4790, pp. 138–150 (2007)

[24] Blanqui, F., Koprowski, A.: CoLoR: a Coq library on well-founded rewrite relations and its application to the automated verification of termination certificates. Mathematical Structures in Computer Science 21(4), 827–859 (2011)

[25] Bofill, M., Borralleras, C., Rodríguez-Carbonell, E., Rubio, A.: The recursive path and polynomial ordering for first-order and higher-order terms. Journal of Logic and Computation 23(1), 263–305 (2013)

[26] Bognar, M.: A point version of decreasing diagrams. In: Proc. Accolade 1996. Dutch Graduate School in Logic. pp. 1–14 (1997). The formalization is available from `http://web.archive.org/web/20051226052550/http://www.cs.vu.nl/~mirna/`

[27] Buchholz, W.: Another rewrite system for the standard Hydra battle. In: Proc. Mini-Workshop: Logic, Combinatorics and Independence Results. Oberwolfach Reports, vol. 3(4), pp. 3099–3102. European Mathematical Society (2006)

[28] Buchholz, W.: An independence result for $(\pi^1_1 - \mathrm{CA}) + \mathrm{BI}$. Annals of Pure and Applied Logic 33, 131–155 (1987)

[29] Charalambides, C.: Enumerative Combinatorics. Chapman & Hall/CRC (2002)

[30] Chuan-Chong, C., Khee-Meng, K.: Principles and Techniques in Combinatorics. World Scientific Publishing Company (1992)

[31] Cichon, E.: A short proof of two recently discovered independence results using recursion theoretic methods. Proc. American Mathematical Society 87(4), 704–706 (1983)

[32] Cirstea, H., Kirchner, C.: The simply typed rewriting calculus. In: Proc. 3rd International Workshop on Rewriting Logic and its Applications (WRLA 2000). Electronic Notes in Theoretical Computer Science, vol. 36, pp. 24–42 (2000)

[33] Codish, M., Lagoon, V., Stuckey, P.: Solving partial order constraints for LPO termination. Journal on Satisfiability, Boolean Modeling and Computation 5(1-4), 193–215 (2008)

[34] Cok, D., Deharbe, D., Weber, T.: Satisfiability modulo theories competition (SMT-COMP) (2014). `http://sourceforge.smtcomp.org`

[35] Contejean, E., Courtieu, P., Forest, J., Pons, O., Urbain, X.: Certification of automated termination proofs. In: Proc. 6th International Symposium on Frontiers of Combining Systems (FroCoS 2007). Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 4720, pp. 148–162 (2007)

[36] Contejean, E., Marché, C., Tomás, A.P., Urbain, X.: Mechanically proving termination using polynomial interpretations. Journal of Automated Reasoning 34(4), 325–363 (2005)

[37] Contejean, E., Courtieu, P., Forest, J., Pons, O., Urbain, X.: Automated certified proofs with C*i*ME3. In: Proc. 22nd International Conference on Rewriting Techniques and Applications (RTA 2011). Leibniz International Proceedings in Informatics, vol. 10, pp. 21–30 (2011)

[38] Dershowitz, N.: 33 Examples of termination. In: French Spring School of Theoretical Computer Science. Lecture Notes in Computer Science, vol. 909, pp. 16–26 (1995)

[39] Dershowitz, N.: Orderings for term-rewriting systems. Theoretical Computer Science 17, 279–301 (1982)

[40] Dershowitz, N.: Trees, ordinals, and termination. In: Proc. 5th International Joint Conference on Theory and Practice of Software Development (TAPSOFT 1993). Lecture Notes in Computer Science, vol. 668, pp. 243–250 (1993)

[41] Dershowitz, N., Jouannaud, J.P.: Rewrite systems. In: Handbook of Theoretical Computer Science. vol. B: Formal Models and Semantics. pp. 243–320. Elsevier (1990)

[42] Dershowitz, N., Manna, Z.: Proving termination with multiset orderings. Communications of the ACM 22(8), 465–476 (1979)

[43] Dershowitz, N., Moser, G.: The Hydra battle revisited. In: Rewriting, Computation and Proof, Essays Dedicated to Jean-Pierre Jouannaud on the Occasion of his 60th Birthday. Lecture Notes in Computer Science, vol. 4600, pp. 1–27 (2007)

[44] Endrullis, J., Waldmann, J., Zantema, H.: Matrix interpretations for proving termination of term rewriting. Journal of Automated Reasoning 40(2-3), 195–220 (2008)

[45] Felgenhauer, B.: A proof order for decreasing diagrams. In: Proc. 1st International Workshop on Confluence (IWC 2012). pp. 7–14 (2012)

[46] Felgenhauer, B.: Rule labeling for confluence of left-linear term rewrite systems. In: Proc. 2nd International Workshop on Confluence (IWC 2013). pp. 23–27 (2013)

[47] Felgenhauer, B., Zankl, H., Middeldorp, A.: Proving confluence with layer systems. In: Proc. 31st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2011). Leibniz International Proceedings in Informatics, vol. 13, pp. 288–299 (2011)

[48] Fleischer, R.: Die another day. Theory of Computing Systems 44(2), 205–214 (2009)

[49] Fuhs, C., Giesl, J., Middeldorp, A., Schneider-Kamp, P., Thiemann, R., Zankl, H.: SAT solving for termination analysis with polynomial interpretations. In: Proc. 10th International Conference on Theory and Applications of Satisfiability Testing (SAT 2007). Lecture Notes in Computer Science, vol. 4501, pp. 340–354 (2007)

[50] Galdino, A., Ayala-Rincón, M.: A formalization of Newman's and Yokouchi's lemmas in a higher-order language. Journal of Formal Reasoning 1(1), 39–50 (2008)

[51] Galdino, A., Ayala-Rincón, M.: A formalization of the Knuth-Bendix(-Huet) critical pair theorem. Journal of Automated Reasoning 45(3), 301–325 (2010)

[52] Gebhardt, A., Hofbauer, D., Waldmann, J.: Matrix evolutions. In: Proc. 9th International Workshop on Termination (WST 2007). pp. 4–8 (2007)

[53] Genet, T.: Decidable approximations of sets of descendants and sets of normal forms. In: Proc. 9th International Conference on Rewriting Techniques and Applications (RTA 1998). Lecture Notes in Computer Science, vol. 1379, pp. 151–165 (1998)

[54] Gentzen, G.: Die Widerspruchsfreiheit der reinen Zahlentheorie. Mathematische Annalen 122, 493–565 (1936)

[55] Geser, A., Hofbauer, D., Waldmann, J., Zantema, H.: On tree automata that certify termination of left-linear term rewriting systems. Information and Computation 205(4), 512–534 (2007)

[56] Geser, A.: Relative termination. PhD thesis, Universität Passau, Germany (1990). Available as: Report 91-03, Ulmer Informatik-Berichte, Universität Ulm (1991)

[57] Giesl, J., Raffelsieper, M., Schneider-Kamp, P., Swiderski, S., Thiemann, R.: Automated termination proofs for Haskell by term rewriting. ACM Transactions on Programming Languages and Systems 33(2), 1–39 (2011)

[58] Giesl, J., Thiemann, R., Schneider-Kamp, P.: The dependency pair framework: Combining techniques for automated termination proofs. In: Proc. 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2004). Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 3717, pp. 301–331 (2004)

[59] Giesl, J., Thiemann, R., Schneider-Kamp, P.: Proving and disproving termination of higher-order functions. In: Proc. 5th International Symposium on Frontiers of Combining Systems (FroCoS 2005). Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 3717, pp. 216–231 (2005)

[60] Giesl, J., Thiemann, R., Schneider-Kamp, P., Falke, S.: Mechanizing and improving dependency pairs. Journal of Automated Reasoning 37(3), 155–203 (2006)

[61] Giesl, J., Mesnard, F., Rubio, A., Thiemann, R., Waldmann, J.: Termination competition (TermComp) (2014). `http://termination-portal.org/wiki/Termination_Competition`

[62] Godoy, G., Tiwari, A.: Confluence of shallow right-linear rewrite systems. In: Proc. 14th Annual Conference of the European Association for Computer Science Logic (CSL 2005). Lecture Notes in Computer Science, vol. 3634, pp. 541–556 (2005)

[63] Goodstein, R.L.: On the restricted ordinal theorem. Journal of Symbolic Logic 9(2), 33–41 (1944)

[64] Gramlich, B.: Confluence without termination via parallel critical pairs. In: Proc. 21st International Colloquium on Trees in Algebra and Programming (CAAP 1996). Lecture Notes in Computer Science, vol. 1059, pp. 211–225 (1996)

[65] Gramlich, B.: Relating innermost, weak, uniform and modular termination of term rewriting systems. In: Proc. 3rd International Conference on Logic Programming and Automated Reasoning (LPAR 1992). Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 624, pp. 285–296 (1992)

[66] Gulwani, S., Mehra, K., Chilimbi, T.: SPEED: precise and efficient static estimation of program computational complexity. In: Proc. 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2009). pp. 127–139 (2009)

[67] Hamano, M., Okada, M.: A relationship among Gentzen's proof-reduction, Kirby-Paris' Hydra game and Buchholz's Hydra game. Mathematical Logic Quarterly 43, 103–120 (1997)

[68] Hirokawa, N., Middeldorp, A.: Automating the dependency pair method. Information and Computation 199(1-2), 172–199 (2005)

[69] Hirokawa, N., Middeldorp, A.: Commutation via relative termination. In: Proc. 2nd International Workshop on Confluence (IWC 2013). pp. 29–33 (2013)

[70] Hirokawa, N., Middeldorp, A.: Decreasing diagrams and relative termination. In: Proc. 5th International Joint Conference on Automated Reasoning (IJCAR 2010). Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 6173, pp. 487–501 (2010)

[71] Hirokawa, N., Middeldorp, A.: Decreasing diagrams and relative termination. Journal of Automated Reasoning 47(4), 481–501 (2011)

[72] Hirokawa, N., Middeldorp, A.: Polynomial interpretations with negative coefficients. In: Proc. 7th International Conference on Artificial Intelligence and Symbolic Computation (AISC 2004). Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 3249, pp. 185–198 (2004)

[73] Hirokawa, N., Middeldorp, A.: Tyrolean Termination Tool: Techniques and features. Information and Computation 205(4), 474–511 (2007)

[74] Hirokawa, N., Middeldorp, A., Zankl, H.: Uncurrying for termination. In: Proc. 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2008). Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 5330, pp. 667–681 (2008)

[75] Hirokawa, N., Moser, G.: Automated complexity analysis based on the dependency pair method. In: Proc. 4th International Joint Conference on Automated Reasoning (IJCAR 2008). Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 5195, pp. 364–379 (2008)

[76] Hirokawa, N., Moser, G.: Complexity, graphs, and the dependency pair method. In: Proc. 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2008). Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 5330, pp. 652–666 (2008)

[77] Hofbauer, D., Lautemann, C.: Termination proofs and the length of derivations (preliminary version). In: Proc. 3rd International Conference on Rewriting Techniques and Applications (RTA 1989). Lecture Notes in Computer Science, vol. 355, pp. 167–177 (1989)

[78] Hofbauer, D., Waldmann, J.: Complexity bounds from relative termination proofs. Presentation at the Workshop on Proof Theory and Rewriting, Obergurgl (2006). Available from `http://www.imn.htwk-leipzig.de/~waldmann/talk/06/rpt/rel/main.pdf`

[79] Hofbauer, D., Waldmann, J.: Termination of string rewriting with matrix interpretations. In: Proc. 17th International Conference on Rewriting Techniques and Applications (RTA 2006). Lecture Notes in Computer Science, vol. 4098, pp. 328–342 (2006)

[80] Hofbauer, D.: Termination proofs by context-dependent interpretations. In: Proc. 12th International Conference on Rewriting Techniques and Applications (RTA 2001). Lecture Notes in Computer Science, vol. 2051, pp. 108–121 (2001)

[81] Hofbauer, D.: Termination proofs by multiset path orderings imply primitive recursive derivation lengths. Theoretical Computer Science 105(1), 129–140 (1992)

[82] Hoffmann, J., Aehlig, K., Hofmann, M.: Multivariate amortized resource analysis. ACM Transactions on Programming Languages and Systems 34(3), 1–14 (2012)

[83] Hong, H., Jakuš, D.: Testing positiveness of polynomials. Journal of Automated Reasoning 21(1), 23–38 (1998)

[84] Huet, G.: Confluent reductions: Abstract properties and applications to term rewriting systems. Journal of the ACM 27(4), 797–821 (1980)

[85] Huet, G.: Residual theory in lambda-calculus: A formal development. Journal of Functional Programming 4(3), 371–394 (1994)

[86] Huet, G., Lankford, D.: On the uniform halting problem for term rewriting systems. Technical Report 282, INRIA, Le Chesnay, France (1978)

[87] Huet, G.: A complete proof of correctness of the Knuth-Bendix completion algorithm. Journal of Computer and System Sciences 23(1), 11–21 (1981)

[88] Jouannaud, J.P., Okada, M.: A computation model for executable higher-order algebraic specification languages. In: Proc. 6th International Annual IEEE Symposium on Logic in Computer Science (LICS 1991). pp. 350–361 (1991)

[89] Jouannaud, J.P., Rubio, A.: Polymorphic higher-order recursive path orderings. Journal of the ACM 54(1) (2007)

[90] Jouannaud, J.P., van Oostrom, V.: Diagrammatic confluence and completion. In: Proc. 36th International Colloquium on Automata, Languages and Programming (ICALP 2009). Lecture Notes in Computer Science, vol. 5556, pp. 212–222 (2009)

[91] Jungers, R., Protasov, V., Blondel, V.: Efficient algorithms for deciding the type of growth of products of integer matrices. Linear Algebra and its Applications 428(10), 2296–2311 (2008)

[92] Just, W., Weese, M.: Discovering Modern Set Theory. I: The Basics. American Mathematical Society (1996)

[93] Kaliszyk, C., Sternagel, T.: Initial experiments on deriving a complete HOL simplification set. In: Proc. 3rd International Workshop on Proof Exchange for Theorem Proving (PxTP 2013). Easychair Proceedings in Computing, vol. 14, pp. 77–86 (2013)

[94] Kamin, S., Lévy, J.: Two generalizations of the recursive path ordering. Unpublished manuscript, University of Illinois (1980)

[95] Kennaway, R., Klop, J., Sleep, M.R., de Vries, F.J.: Comparing curried and uncurried rewriting. Journal of Symbolic Computation 21(1), 15–39 (1996)

[96] Kirby, L., Paris, J.: Accessible independence results for Peano arithmetic. Bulletin of the London Mathematical Society 14, 285–325 (1982)

[97] Klein, D., Hirokawa, N.: Confluence of non-left-linear TRSs via relative termination. In: Proc. 18th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-18). Lecture Notes in Computer Science (Advanced Research in Computing and Software Science), vol. 7180, pp. 258–273 (2012)

[98] Klein, D., Hirokawa, N.: Maximal completion. In: Proc. 22nd International Conference on Rewriting Techniques and Applications (RTA 2011). Leibniz International Proceedings in Informatics, vol. 10, pp. 71–80 (2011)

[99] Klop, J., van Oostrom, V., de Vrijer, R.: A geometric proof of confluence by decreasing diagrams. Journal of Logic and Computation 10(3), 437–460 (2000)

[100] Knuth, D., Bendix, P.: Simple word problems in universal algebras. In: Computational Problems in Abstract Algebra. Pergamon Press, New York 263–297 (1970)

[101] Kop, C., van Raamsdonk, R.: Higher order dependency pairs for algebraic functional systems. In: Proc. 22nd International Conference on Rewriting Techniques and Applications (RTA 2011). Leibniz International Proceedings in Informatics, vol. 10, pp. 203–218 (2011)

[102] Kop, C., van Raamsdonk, F.: Dynamic dependency pairs for algebraic functional systems. Logical Methods in Computer Science 8(2:10), 1–51 (2012)

[103] Koprowski, A., Waldmann, J.: Arctic termination ... below zero. In: Proc. 19th International Conference on Rewriting Techniques and Applications (RTA 2008). Lecture Notes in Computer Science, vol. 5117, pp. 202–216 (2008)

[104] Koprowski, A., Waldmann, J.: Max/plus tree automata for termination of term rewriting. Acta Cybernetica 19(2), 357–392 (2009)

[105] Korp, M., Middeldorp, A.: Match-bounds revisited. Information and Computation 207(11), 1259–1283 (2009)

[106] Korp, M., Middeldorp, A.: Proving termination of rewrite systems using bounds. In: Proc. 18th International Conference on Rewriting Techniques and Applications (RTA 2007). Lecture Notes in Computer Science, vol. 4533, pp. 273–287 (2007)

[107] Korp, M., Sternagel, C., Zankl, H., Middeldorp, A.: Tyrolean Termination Tool 2. In: Proc. 20th International Conference on Rewriting Techniques and Applications (RTA 2009). Lecture Notes in Computer Science, vol. 5595, pp. 295–304 (2009)

[108] Kovács, L., Moser, G., Voronkov, A.: On transfinite Knuth-Bendix orders. In: Proc. 23rd International Conference on Automated Deduction (CADE 2011). Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 6803, pp. 384–399 (2011)

[109] Krishna Rao, M.R.K.: Some characteristics of strong innermost normalization. Theoretical Computer Science 239, 141–164 (2000)

[110] Kusakari, K., Isogai, Y., Sakai, M., Blanqui, F.: Static dependency pair method based on strong computability for higher-order rewrite systems. IEICE Transactions 92-D(10), 2007–2015 (2009)

[111] Kusakari, K., Sakai, M.: Enhancing dependency pair method using strong computability in simply-typed term rewriting. Applicable Algebra in Engineering, Communication and Computing 18(5), 407–431 (2007)

[112] Lankford, D.: On proving term rewrite systems are noetherian. Technical Report MTP-3, Louisiana Technical University, Ruston, LA, USA (1979)

[113] Lepper, I.: Simply terminating rewrite systems with long derivations. Archive for Mathematical Logic 43(1), 1–18 (2004)

[114] Lepper, I.: Derivation lengths and order types of Knuth-Bendix orders. Theoretical Computer Science 269(1-2), 433–450 (2001)

[115] Lescanne, P.: Termination of rewrite systems by elementary interpretations. Formal Aspects of Computing 7(1), 77–90 (1995)

[116] Lescanne, P.: Completion procedures as transition rules + control. In: Proc. 3rd International Joint Conference on Theory and Practice of Software Development (TAPSOFT 1989). Lecture Notes in Computer Science, vol. 351, pp. 28–41 (1989)

[117] Lucas, S.: Automatic proofs of termination with elementary interpretations. In: Proc. 9th Spanish Conference on Programming and Languages (PROLE 2009). Electronic Notes in Theoretical Computer Science, vol. 258, pp. 41–61 (2009)

[118] Ludwig, M., Waldmann, U.: An extension of the Knuth-Bendix ordering with LPO-like properties. In: Proc. 14th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2007). Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 4790, pp. 348–362 (2007)

[119] Manolios, P., Vroon, D.: Ordinal arithmetic: Algorithms and mechanization. Journal of Automated Reasoning 34(4), 387–423 (2005)

[120] Middeldorp, A.: Modular properties of term rewriting systems. PhD thesis, Vrije Universiteit, Amsterdam (1990)

[121] Middeldorp, A., Ohsaki, H., Zantema, H.: Transforming termination by self-labelling. In: Proc. 13th International Conference on Automated Deduction (CADE 1996). Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 1104, pp. 373–387 (1996)

[122] Middeldorp, A., Moser, G., Neurauter, F., Waldmann, J., Zankl, H.: Spectral radius theory for automated complexity analysis of rewrite systems. In: Proc. 4th International Conference on Algebraic Informatics (CAI 2011). Lecture Notes in Computer Science, vol. 6742, pp. 1–20 (2011)

[123] Moser, G.: The Hydra battle and Cichon's principle. Applicable Algebra in Engineering, Communication and Computing 20(2), 133–158 (2009)

[124] Moser, G., Schnabl, A.: The derivational complexity induced by the dependency pair method. In: Proc. 20th International Conference on Rewriting Techniques and Applications (RTA 2009). Lecture Notes in Computer Science, vol. 5595, pp. 255–267 (2009)

[125] Moser, G., Schnabl, A.: The derivational complexity induced by the dependency pair method. Logical Methods in Computer Science 7(3:1), 1–38 (2011)

[126] Moser, G., Schnabl, A., Waldmann, J.: Complexity analysis of term rewriting based on matrix and context dependent interpretations. In: Proc. 28th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2008). Leibniz International Proceedings in Informatics, vol. 2, pp. 304–315 (2008)

[127] Moser, G., Schnabl, A.: Proving quadratic derivational complexities using context dependent interpretations. In: Proc. 19th International Conference on Rewriting Techniques and Applications (RTA 2008). Lecture Notes in Computer Science, vol. 5117, pp. 276–290 (2008)

[128] Nagele, J., Thiemann, R.: Certification of confluence proofs using CeTA. In: Proc. 3rd International Workshop on Confluence (IWC 2014). pp. 19–23 (2014)

[129] Nagele, J., Zankl, H.: Certification of confluence proofs via decreasing diagrams. (2014). In progress.

[130] Neurauter, F., Middeldorp, A.: Polynomial interpretations over the reals do not subsume polynomial interpretations over the integers. In: Proc. 21st International Conference on Rewriting Techniques and Applications (RTA 2010). Leibniz International Proceedings in Informatics, vol. 6, pp. 243–258 (2010)

[131] Neurauter, F., Zankl, H., Middeldorp, A.: Revisiting matrix interpretations for polynomial derivational complexity of term rewriting. In: Proc. 17th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-17). Lecture Notes in Computer Science (Advanced Research in Computing and Software Science), vol. 6397, pp. 550–564 (2010)

[132] Nipkow, T.: More Church-Rosser proofs. Journal of Automated Reasoning 26(1), 51–66 (2001)

[133] Nipkow, T., Paulson, L., Wenzel, M.: Isabelle/HOL – A Proof Assistant for Higher-Order Logic. vol. 2283 of Lecture Notes in Computer Science. (2002)

[134] Noschinski, L., Emmes, F., Giesl, J.: Analyzing innermost runtime complexity of term rewriting by dependency pairs. Journal of Automated Reasoning 51(1), 27–56 (2013)

[135] Ohlebusch, E.: On the modularity of confluence of constructor-sharing term rewriting systems. In: Proc. 19th International Colloquium on Trees in Algebra and Programming (CAAP 1994). Lecture Notes in Computer Science, vol. 787, pp. 261–275 (1994)

[136] Ohlebusch, E.: A simple proof of sufficient conditions for the termination of the disjoint union of term rewriting systems. Bulletin of the EATCS 50, 223–228 (1993)

[137] Okui, S.: Simultaneous critical pairs and Church-Rosser property. In: Proc. 9th International Conference on Rewriting Techniques and Applications (RTA 1998). Lecture Notes in Computer Science, vol. 1379, pp. 2–16 (1998)

[138] van Oostrom, V.: Confluence by decreasing diagrams – converted. In: Proc. 19th International Conference on Rewriting Techniques and Applications (RTA 2008). Lecture Notes in Computer Science, vol. 5117, pp. 306–320 (2008)

[139] van Oostrom, V.: Confluence by decreasing diagrams. Theoretical Computer Science 126(2), 259–280 (1994)

[140] van Oostrom, V.: Confluence for abstract and higher-order rewriting. PhD thesis, Vrije Universiteit, Amsterdam (1994)

[141] van Oostrom, V.: Confluence via critical valleys. In: Proc. 6th International Workshop on Higher-Order Rewriting (HOR 2012). pp. 9–11 (2012)

[142] van Oostrom, V.: Decreasing proof orders – interpreting conversions in involutive monoids. In: Proc. 1st International Workshop on Confluence (IWC 2012). pp. 1–4 (2012)

[143] van Oostrom, V.: Developing developments. Theoretical Computer Science 175(1), 159–181 (1997)

[144] van Oostrom, V.: Modularity of confluence constructed. In: Proc. 4th International Joint Conference on Automated Reasoning (IJCAR 2008). Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 5195, pp. 348–363 (2008)

[145] van Oostrom, V.: Random descent. In: Proc. 18th International Conference on Rewriting Techniques and Applications (RTA 2007). Lecture Notes in Computer Science, vol. 4533, pp. 314–328 (2007)

[146] Otto, C., Brockschmidt, M., von Essen, C., Giesl, J.: Automated termination analysis of Java bytecode by term rewriting. In: Proc. 21st International Conference on Rewriting Techniques and Applications (RTA 2010). Leibniz International Proceedings in Informatics, vol. 6, pp. 259–276 (2010)

[147] Oyamaguchi, M.: The Church-Rosser property for ground term-rewriting systems is decidable. Theoretical Computer Science 49, 43–79 (1987)

[148] Oyamaguchi, M., Ohta, Y.: A new parallel closed condition for Church-Rosser of left-linear term rewriting systems. In: Proc. 8th International Conference on Rewriting Techniques and Applications (RTA 1997). Lecture Notes in Computer Science, vol. 1232, pp. 187–201 (1997)

[149] Pfenning, F.: A proof of the Church-Rosser theorem and its representation in a logical framework. Technical Report CMU-CS-92-186, School of Computer Science, Carnegie Mellon University (1992)

[150] Podelski, A., Rybalchenko, A.: A complete method for the synthesis of linear ranking functions. In: Proc. 5th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI 2004). Lecture Notes in Computer Science, vol. 2937, pp. 239–251 (2004)

[151] Rose, H.: Linear Algebra: A Pure Mathematical Approach. Birkhäuser (2002)

[152] Rosen, B.: Tree-manipulating systems and Church-Rosser theorems. Journal of the ACM 20(1), 160–187 (1973)

[153] Ruiz-Reina, J., Alonso, J., Hidalgo, M., Martín-Mateos, F.: Formal proofs about rewriting using ACL2. Annals of Mathematics and Artificial Intelligence 36(3), 239–262 (2002)

[154] Sato, H., Winkler, S., Kurihara, M., Middeldorp, A.: Multi-completion with termination tools (system description). In: Proc. 4th International Joint Conference on Automated Reasoning (IJCAR 2008). Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 5195, pp. 306–312 (2008)

[155] Schnabl, A.: Derivational complexity analysis revisited. PhD thesis, University of Innsbruck (2012)

[156] Schnabl, A.: Cdiprover3: A tool for proving derivational complexities of term rewriting systems. In: Proc. 20th European Summer School on Logic, Language, and Information–Student Sessions (ESSLLI 2008). Lecture Notes in Computer Science, vol. 6211, pp. 142–154 (2009)

[157] Schnabl, A.: Context dependent interpretations. Master's thesis, University of Innsbruck (2007)

[158] Schneider-Kamp, P., Giesl, J., Serebrenik, A., Thiemann, R.: Automated termination proofs for logic programs by term rewriting. ACM Transactions on Computational Logic 11(1), 49 (2010)

[159] Serre, D.: Matrices: Theory and Applications. Springer (2002)

[160] Shankar, N.: A mechanical proof of the Church-Rosser theorem. Journal of the ACM 35(3), 475–522 (1988)

[161] Spoto, F., Mesnard, F., Payet, É.: A termination analyzer for Java bytecode based on path-length. ACM Transactions on Programming Languages and Systems 32(3), 1–70 (2010)

[162] Sternagel, C., Middeldorp, A.: Root-labeling. In: Proc. 19th International Conference on Rewriting Techniques and Applications (RTA 2008). Lecture Notes in Computer Science, vol. 5117, pp. 336–350 (2008)

[163] Sternagel, C., Thiemann, R.: Abstract rewriting. Archive of Formal Proofs (2010)

[164] Sternagel, C., Thiemann, R.: Generalized and formalized uncurrying. In: Proc. 8th International Workshop on Frontiers of Combining Systems. Lecture Notes in Computer Science, vol. 6989, pp. 243–258 (2011)

[165] Sternagel, C., Thiemann, R.: Signature extensions preserve termination – an alternative proof via dependency pairs. In: Proc. 19th Annual Conference of the European Association for Computer Science Logic (CSL 2010). Lecture Notes in Computer Science, vol. 6247, pp. 514–528 (2010)

[166] Sternagel, C., Thiemann, R.: Formalizing Knuth-Bendix orders and Knuth-Bendix completion. In: Proc. 24th International Conference on Rewriting Techniques and Applications (RTA 2013). Leibniz International Proceedings in Informatics, vol. 21, pp. 287–302 (2013)

[167] Sternagel, T., Thiemann, R., Zankl, H., Sternagel, C.: Recording completion for finding and certifying proofs in equational logic. In: Proc. 1st International Workshop on Confluence (IWC 2012). pp. 31–36 (2012)

[168] Støvring, K.: Extending the extensional lambda calculus with surjective pairing is conservative. Logical Methods in Computer Science 2(2:1), 1–14 (2006)

[169] Ströder, T., Giesl, J., Brockschmidt, M., Frohn, F., Fuhs, C., Hensel, J., Schneider-Kamp, P.: Proving termination and memory safety for programs

with pointer arithmetic. In: Proc. 7th International Joint Conference on Automated Reasoning (IJCAR 2014). Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 8562, pp. 208–223 (2014)

[170] Stump, A., Zantema, H., Kimmell, G., Omar, R.: A rewriting view of simple typing. Logical Methods in Computer Science 9(1:4), 1–29 (2012)

[171] Takahashi, M.: Parallel reductions in λ-calculus. Information and Computation 118(1), 120–127 (1995)

[172] TeReSe: Term Rewriting Systems. vol. 55 of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press (2003)

[173] Thiemann, R.: Certification of confluence proofs using CeTA. In: Proc. 1st International Workshop on Confluence (IWC 2012). p. 45 (2012)

[174] Thiemann, R.: The DP framework for proving termination of term rewriting. PhD thesis, RWTH Aachen (2007). Available as: Report AIB-2007-17, Aachener Informatik-Berichte, RWTH Aachen

[175] Thiemann, R.: A formalization of termination techniques in Isabelle/HOL. Habilitation thesis, University of Innsbruck (2013)

[176] Thiemann, R., Sternagel, C.: Certification of termination proofs using CeTA. In: Proc. 22nd International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2009). Lecture Notes in Computer Science, vol. 5674, pp. 452–468 (2009)

[177] Tiwari, A.: Deciding confluence of certain term rewriting systems in polynomial time. In: Proc. 17th International Annual IEEE Symposium on Logic in Computer Science (LICS 2002). pp. 447–457 (2002)

[178] Touzet, H.: Encoding the Hydra battle as a rewrite system. In: Proc. 23rd International Symposium on Mathematical Foundations of Computer Science (MFCS 1998). Lecture Notes in Computer Science, vol. 1450, pp. 267–276 (1998)

[179] Toyama, Y.: Commutativity of term rewriting systems. In: Programming of Future Generation Computers II. North-Holland 393–407 (1988)

[180] Toyama, Y.: On the Church-Rosser property for the direct sum of term rewriting systems. Journal of the ACM 34(1), 128–143 (1987)

[181] Toyama, Y.: On the Church-Rosser property of term rewriting systems. Technical Report 17672, NTT ECL (1981)

[182] Toyama, Y.: Termination of S-expression rewriting systems: Lexicographic path ordering for higher-order terms. In: Proc. 15th International Conference on Rewriting Techniques and Applications (RTA 2004). Lecture Notes in Computer Science, vol. 3091, pp. 40–54 (2004)

[183] TPDB: Termination problem data base (2010). version 7.0.2. Available from `http://termcomp.uibk.ac.at/status/downloads/tpdb-7.0.2.tar.gz`

[184] Turing, A.: Checking a large routine. In: Report of a Conference on High Speed Automatic Calculating Machines. pp. 67–68 (1949)

[185] Urban, C., Miné, A.: An abstract domain to infer ordinal-valued ranking functions. In: Proc. 23rd European Symposium on Programming (ESOP 2014). Lecture Notes in Computer Science, vol. 8410, pp. 412–431 (2014)

[186] Waldmann, J.: Weighted automata for proving termination of string rewriting. Journal of Automata, Languages and Combinatorics 12(4), 545–570 (2007)

[187] Waldmann, J.: Polynomially bounded matrix interpretations. In: Proc. 21st International Conference on Rewriting Techniques and Applications (RTA 2010). Leibniz International Proceedings in Informatics, vol. 6, pp. 357–372 (2010)

[188] Waldmann, U.: Semantics of order-sorted specifications. Theoretical Computer Science 94(1), 1–35 (1992)

[189] Wehrman, I., Stump, A., Westbrook, E.: Slothrop: Knuth-Bendix completion with a modern termination checker. In: Proc. 17th International Conference on Rewriting Techniques and Applications (RTA 2006). Lecture Notes in Computer Science, vol. 4098, pp. 287–296 (2006)

[190] Weiermann, A.: Termination proofs for term rewriting systems by lexicographic path orderings imply multiply recursive derivation lengths. Theoretical Computer Science 139(1-2), 355–362 (1995)

[191] Winkler, S., Middeldorp, A.: Termination tools in ordered completion. In: Proc. 5th International Joint Conference on Automated Reasoning (IJCAR 2010). Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 6173, pp. 518–532 (2010)

[192] Winkler, S., Sato, H., Middeldorp, A., Kurihara, M.: Multi-completion with termination tools. Journal of Automated Reasoning 50(3), 317–354 (2013)

[193] Winkler, S., Zankl, H., Middeldorp, A.: Beyond Peano arithmetic – Automatically proving termination of the Goodstein sequence. In: Proc. 24th International Conference on Rewriting Techniques and Applications (RTA 2013). Leibniz International Proceedings in Informatics, vol. 21, pp. 335–351 (2013)

[194] Winkler, S., Zankl, H., Middeldorp, A.: Ordinals and Knuth-Bendix orders. In: Proc. 18th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-18). Lecture Notes in Computer Science (Advanced Research in Computing and Software Science), vol. 7180, pp. 420–434 (2012)

[195] Wolfram Research, Inc.: Mathematica. Version 10.0, Wolfram Research, Inc. Champaign, Illinois (2014)

[196] Xi, H.: Towards automated termination proofs through "freezing". In: Proc. 9th International Conference on Rewriting Techniques and Applications (RTA 1998). Lecture Notes in Computer Science, vol. 1379, pp. 271–285 (1998)

[197] Zankl, H.: Confluence by decreasing diagrams – formalized. In: Proc. 24th International Conference on Rewriting Techniques and Applications (RTA 2013). Leibniz International Proceedings in Informatics, vol. 21, pp. 352–367 (2013)

[198] Zankl, H.: Confluence by decreasing diagrams – formalized. Computing Research Repository abs/1210.1100v2, 17 pages (2013)

[199] Zankl, H.: Decreasing diagrams. Archive of Formal Proofs (2013). Formal proof development, `http://afp.sf.net/entries/Decreasing-Diagrams.shtml`

[200] Zankl, H., Felgenhauer, B., Middeldorp, A.: CSI – A confluence tool. In: Proc. 23rd International Conference on Automated Deduction (CADE 2011). Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 6803, pp. 499–505 (2011)

[201] Zankl, H., Felgenhauer, B., Middeldorp, A.: Labelings for decreasing diagrams. In: Proc. 22nd International Conference on Rewriting Techniques and Applications (RTA 2011). Leibniz International Proceedings in Informatics, vol. 10, pp. 377–392 (2011)

[202] Zankl, H., Hirokawa, N., Middeldorp, A.: Uncurrying for innermost termination and derivational complexity. In: Proc. 5th International Workshop on Higher-Order Rewriting (HOR 2010). Electronic Proceedings in Theoretical Computer Science, vol. 49, pp. 46–57 (2011)

[203] Zankl, H., Winkler, S., Middeldorp, A.: Automating ordinal interpretations. In: Proc. 12th International Workshop on Termination (WST 2012). pp. 94–98 (2012)

[204] Zankl, H., Korp, M.: The derivational complexity of the bits function and the derivation gap principle. In: Proc. 11th International Workshop on Termination (WST 2010). (2010). 5 pages

[205] Zankl, H., Korp, M.: Modular complexity analysis for term rewriting. Logical Methods in Computer Science 10(1:19), 1–33 (2014)

[206] Zankl, H., Korp, M.: Modular complexity analysis via relative complexity. In: Proc. 21st International Conference on Rewriting Techniques and Applications (RTA 2010). Leibniz International Proceedings in Informatics, vol. 6, pp. 385–400 (2010)

[207] Zankl, H., Korp, M.: On implementing modular complexity analysis. In: Proc. 8th International Workshop on the Implementation of Logics (IWIL 2010). Easychair Proceedings in Computing, vol. 2, pp. 42–47 (2010)

[208] Zankl, H., Middeldorp, A.: Satisfiability of non-linear (ir)rational arithmetic. In: Proc. 16th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-16). Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 6355, pp. 481–500 (2010)

[209] Zantema, H.: The termination hierarchy for term rewriting. Applicable Algebra in Engineering, Communication and Computing 12(1-2), 3–19 (2001)

[210] Zantema, H.: Termination. In: Term Rewriting Systems. Cambridge University Press, Cambridge 181–259 (2003)