Ramsey Quantifiers in First-Order Logic: Complexity and Applications to Verification

Thesis approved by the
Department of Computer Science
University of Kaiserslautern-Landau
for the award of the Doctoral Degree

Doctor of Natural Sciences (Dr. rer. nat.)

to

Pascal Bergsträßer

Date of Defense	:	May 21, 2025
Dean	:	Prof. Dr. Christoph Garth
Reviewers	:	Prof. Dr. Anthony W. Lin Dr. Georg Zetzsche Prof. Dr. Leonid Libkin

DE-386

Abstract

The verification of software correctness is becoming increasingly important due to the widespread use of computers, particularly in safety-critical environments. Many complex systems (such as those involving numeric data types, recursion, or multithreading) naturally exhibit an unbounded number of program states (a.k.a. configurations). These are referred to as *infinite-state systems* and their verification is an active research area with numerous unresolved problems. A key focus in this field is the verification of safety and liveness properties. While safety verification can be reduced to reachability via finite paths, liveness verification involves reasoning about infinite executions, making it considerably more challenging. Most approaches for checking liveness properties are devised for specific classes of systems.

We study so-called *Ramsey quantifiers*, which state the existence of infinite cliques in the graph defined by a formula, in various first-order theories. We show that over automatic structures, which is the class of first-order structures whose domain and relations are regular, Ramsey quantifiers can be *evaluated* in logarithmic space. For tree-regular relations we give a polynomial-time evaluation algorithm if a deterministic bottom-up tree automaton is provided or if the relation is transitive. We furthermore study Ramsey quantifiers in more specific theories. Linear arithmetics over integers/reals are prominent theories featured in SMT solvers. We show that the Ramsey quantifier can be eliminated from existential formulas in linear arithmetic over the integers, reals, and their mixture in polynomial time. This directly enables the use of highly optimized SMT solvers. As an application, we provide a general framework for liveness verification across a wide range of infinite-state systems. For example, we prove precise complexity results for recurrent reachability with generalized Büchi condition over (tree-)regular relations and linear liveness for succinct one-counter automata, reversal-bounded counter machines, continuous vector addition systems with states, and Parikh automata.

As a second major application, we identify the problem of checking whether a given formula is *monadically decomposable*, i.e. equivalent to a Boolean combination of formulas, each depending only on a single free variable. Monadic decompositions can, for example, be used in constraint databases, string analysis, and quantifier elimination. We prove precise complexity results in the case of quantifier-free formulas in linear integer/real arithmetic. We additionally explore a related problem: checking whether a relation on words (resp. trees) is *(tree-)recognizable*, i.e. a finite union of Cartesian products of (tree-)regular languages.

Resulting Publications

Parts of this thesis appeared in the following joint publications:

[Ber+24] P. Bergsträßer, M. Ganardi, A. W. Lin, and G. Zetzsche. "Ramsey Quantifiers in Linear Arithmetics". In: *Proceedings of the ACM on Programming Languages* 8.POPL (2024), pp. 1–32. DOI: 10.1145/3632843.

[BG23] P. Bergsträßer and M. Ganardi. "Revisiting Membership Problems in Subclasses of Rational Relations". In: *38th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2023, Boston, MA, USA, June 26-29, 2023*. IEEE, 2023, pp. 1–14. DOI: 10.1109/LICS56636.2023.10175722.

[Ber+22] P. Bergsträßer, M. Ganardi, A. W. Lin, and G. Zetzsche. "Ramsey Quantifiers over Automatic Structures: Complexity and Applications to Verification". In: *37th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2022, Haifa, Israel, August 2-5, 2022*. Ed. by C. Baier and D. Fisman. ACM, 2022, 28:1–28:14. DOI: 10.1145/3531130.3533346.

The following publication by the author is not part of this thesis:

[BKLZ24] P. Bergsträßer, C. Köcher, A. W. Lin, and G. Zetzsche. "The Power of Hard Attention Transformers on Data Sequences: A formal language theoretic perspective". In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. Ed. by A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang. 2024. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/af58a33861ac45472ea1cc5860d2b13e-Paper-Conference.pdf.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisors, Anthony and Georg. Your exceptional guidance and unwavering support have been instrumental in shaping both my research career and personal growth throughout this journey. Your dedication and commitment to excellent research have inspired me at every stage and I feel incredibly fortunate to have worked under your supervision.

I am also immensely grateful to Moses. You have been like a third advisor to me. Your brilliant ideas, enriching discussions, and attention to detail have had a profound impact not only on this work, but, more importantly, you taught me what it means to be a good researcher. I deeply appreciate the time and energy you have devoted to our collaboration.

I thank all my (former) colleagues for the fun conversations at Mensa, in the coffee room, or during dinner when we had guests. A special thanks to Oliver for making coffee and sharing his snacks.

To my parents, Kerstin and Ralph, thank you from the bottom of my heart for always believing in me and for encouraging me to choose my own path. Your unconditional support has been a constant source of strength and motivation throughout my academic career. I could not have reached this point without you. I also thank Vivien and Jens for always cheering me on and being there for me. To my wonderful girlfriend Lena, your loving and emotional support carried me through the final stretch of this journey. Thank you for reminding me of what truly matters in life.

This thesis is a reflection of all the support, wisdom, and guidance I have received and I am forever grateful to those who have helped me along the way.

Contents

1 Introduction

As computers have become an integral part of all areas of life, the reliability and correctness of programs, especially in safety-critical systems, is becoming increasingly important. Here, instead of relying on mere testing, one strives to have mathematical guarantees that the implementation meets its specification. A particularly successful method in the area of formal verification is *model checking*. In the setting of model checking, the program is represented as a *transition system* consisting of a set of *program states* or *configurations* and a *transition relation* that defines all possible transitions between configurations in the system. The specification is formulated in some logical language, most commonly in *first-order logic* or *linear temporal logic* (LTL). Model checking is then the task of testing whether the transition system satisfies the logical formula. Usually, the property expressed by the specification falls into one of two categories: safety or liveness. Loosely speaking, *safety* can be thought of as the property that "something bad never happens". This means, if started in the initial configuration, the transition system will never reach "bad" configurations. Whereas *liveness* properties describe that "something good eventually happens", meaning that from the initial configuration every run reaches "good" configurations. The relevance of safety and liveness becomes particularly apparent when considering ω-regular specifications, i.e. given by a nondeterministic Büchi automaton, since any ω-regular property can be written as the conjunction of a safety and a liveness property [AS87]. When verifying programs, the number of possible inputs and therefore also the number of reachable configurations is a priori not bounded. In this case it is natural to assume that the number of configurations is possibly infinite. Such a transition system is said to be *infinite-state*. Even if there is an upper bound on the number of possible configurations (e.g. in hardware verification), it is often useful to consider infinite-state abstractions. The reason is that the upper bound on the number of configurations can become very large compared to the representation size of the transition system, which is known as the *state-explosion problem*. The abstraction as an infinite-state system, however, can have a more succinct finite *symbolic representation*. Model checking with respect to such symbolically representable infinite-state systems is referred to as *symbolic model checking* [Bur+90; McM93]. To verify that an infinite-state system satisfies a safety condition, it suffices to check whether there exists a path from the initial configuration to some bad configuration. Thus, it is enough to solve the *reachability problem* for the considered class of systems. Liveness verification, however, is extremely challenging. Instead of reachability via finite paths, one has to reason about *infinite paths* since a counterexample for a liveness condition is an infinite run in the transition system such that good behavior is never observed. An example of a liveness condition is termination of a loop. Here, a counterexample is an infinite execution that never fulfills the loop's exit condition. Due to its complexity, most approaches to

liveness verification are typically designed to only solve a simpler subproblem. It turns out that similar challenges arise when considering the seemingly unrelated problems of recognizability and monadic decomposability, which we will define later. In this thesis we identify a unifying framework that helps solve all the aforementioned problems as well as provides precise complexity results.

1.1 Regular Model Checking

Frequently used symbolic representations of infinite-state systems are in terms of automata, where the binary (one-step) transition relation is *(tree-)regular* and given as an automaton reading two words/trees synchronously (a.k.a. *synchronous transducer*). Model checking transition systems that possess (tree-)regular transition relations is referred to as *regular model checking* (see e.g. the survey [Abd+04]). Regular model checking is undecidable in general since even the transition relation of a Turing machine is regular, which implies that already reachability is undecidable [AK86]. Thus, general algorithms, including so-called *acceleration techniques* [JN00; Bou+00; Abd+02; Bar+05], that compute the effect of arbitrarily long (but finite) sequences of transitions, can only provide semi-decision procedures. However, many useful subclasses of systems where acceleration is guaranteed to terminate have been identified. Examples are pushdown systems [Bur+90; Cau92; FWW97; ES01], which can be used to model recursive programs, and ground tree rewriting systems [BT12; Lin12]. Once (a finite representation of) the set of reachable configurations is computed, safety can immediately be checked. Liveness, however, requires additional techniques. Most approaches for liveness analysis are devised for special classes of systems. For instance, it was shown by Bouajjani, Esparza, and Maler [BEM97] that LTL model checking, which allows the verification of a wide range of liveness properties, for pushdown systems is EXP-complete and in P if the LTL formula for the specification is fixed. For this, the automata-theoretic approach by Vardi and Wolper [VW86] was used, where instead of checking whether every run satisfies the LTL formula, the complement is checked, i.e. whether there exists a run that satisfies the negation of the formula. For the complement one can construct a (exponentially sized) Büchi automaton accepting the runs that fulfill the negated LTL formula and take the product with the system under consideration. Then it remains to check a Büchi condition for the resulting system, i.e. whether there exists a run that visits some set of configurations infinitely often. This problem is called *recurrent reachability*. Thus, LTL model checking is reduced to recurrent reachability over some system whose size is exponential in the formula size but polynomial in the size of the given system. For example, recurrent reachability was also shown to be decidable in polynomial time for ground tree rewriting systems [Löd06]. A more general approach for recurrent reachability was developed by To and Libkin [TL08; TL10]. They prove that under the assumption that the *reachability relation*, i.e. the transitive closure of the transition relation, is (tree-)regular and supplied as input in form of an automaton over words/trees, recurrent reachability is decidable in polynomial time. Examples of transition systems that have (tree-)regular reachability relations include pushdown sys-

tems [Esp+00], ground tree rewriting systems [Löd06], and PA-processes [LS05]. In all of these examples, an automaton for the reachability relation can even be constructed in polynomial time. Thus, recurrent reachability can be decided fully automatically in polynomial time for these systems.

Linear arithmetic systems Instead of describing infinite-state systems via automata, in some cases it can be more suitable to use logical formulas. This is for example the case for various systems with counters and/or clocks. Here, one most commonly uses first-order logic (or fragments thereof), where formulas may use existential and universal quantification and variables range over some domain. The most important first-order theories in the context of verifying infinite-state systems are *linear integer arithmetic* (LIA), *linear real arithmetic* (LRA), and their mixture *linear integer real arithmetic* (LIRA). Their domains are the integers/reals and they allow addition and the usual linear order on numbers. These theories are also the most prominent theories in the area of *satisfiability modulo theories* (SMT), where satisfiability of a quantifier-free or existential formula in some fixed theory is to be checked. Even though satisfiability is NP-complete in the existential fragments of all of the three theories, SMT solvers such as Z3 [MB08] or cvc5 [Bar+22] have made an enormous stride forward in the last decades to the extent that they are now capable of solving practical industrial instances. Examples of systems that can be modeled using linear arithmetic over integers/reals include recursive programs with numeric data types [HL11], programs with dense/discrete clocks [CJ99; Dan+00; DSK01; Dan01], linear hybrid systems [BH06], and numeric abstractions of programs that manipulate lists and arrays [Bou+11; HL11]. Again, safety properties can be immediately verified if the reachability relation is given by a formula in linear arithmetic by checking satisfiability using highly optimized SMT solvers. For liveness, however, a general framework that directly enables the use of SMT solvers is missing so far. Dang and Ibarra [DI02] provide a general framework for deciding *linear liveness*, a form of recurrent reachability where the set of target configurations is defined via linear arithmetic formulas, but the resulting formulas, whose satisfiability has to be checked, are large and contain multiple quantifier alternations. SMT solvers, however, are, as remarked above, specialized for formulas in the existential fragment, which only allows a single block of existential quantifiers.

1.2 Ramsey Quantifiers Come to Rescue

We address the lack of an efficient general framework for proving liveness properties by using so-called *Ramsey quantifiers*, that were first introduced by Magidor and Malitz [MM77] and are in the literature therefore also known as *Magidor-Malitz quantifiers*. Intuitively, a Ramsey quantifier, denoted by $\exists^{\text{ram}}$, states the existence of an infinite clique in the graph defined by a formula. More precisely, for a formula $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ with vectors of variables $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$, where $\boldsymbol{x}$ and $\boldsymbol{y}$ have the same dimension, the Ramsey quantified formula

$$\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$$

Algorithm 1 Example of a non-terminating program

```
1: real x_1 ← input-real()
2: int x_2 ← input-int()
3: assert x_1 > 0
4: while x_2 > 0 do
5:     real t_1 ← input-real()
6:     assert t_1 ≥ 0.5x_1 + 0.5
7:     x_1 ← t_1
8:     int t_2 ← input-int()
9:     assert t_2 ≥ 0
10:    x_2 ← x_2 − ⌊x_1⌋ − t_2
11: end while
```

is satisfied for some valuation $\boldsymbol{c}$ of $\boldsymbol{z}$ if and only if there exists an *infinite (directed) clique* with respect to $\boldsymbol{c}$, i.e. an infinite sequence $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots$ of pairwise distinct vectors such that

$$\varphi(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{c}) \text{ for all } 1 \leq i < j.$$

For example, let $\varphi := x < y \wedge x \leq z$ be a formula in linear real arithmetic. Then $\exists^{\text{ram}} x, y \colon \varphi(x, y, z)$ expresses the existence of a strictly increasing infinite sequence that is bounded by z. Over the reals, every valuation c of z admits such a sequence (e.g. take any strictly increasing sequence that converges to c). However, if φ is interpreted as a formula in linear integer arithmetic, then clearly such a sequence does not exist for any c. In the above definition we used the notion of *directed* cliques. In the literature (see e.g. [MM77]), Ramsey quantifiers are usually defined in the *undirected* semantics, i.e. cliques force edges in both directions. Clearly, the undirected version is expressible in the directed version. We deviate from the standard undirected definition since the directed notion allows for simpler proofs and, more importantly, can immediately be applied to liveness verification.

Using Ramsey quantifiers to verify liveness To demonstrate how Ramsey quantifiers can be used for expressing liveness properties, let us consider Algorithm 1. The program contains two variables x_1, x_2 and two temporary variables t_1, t_2, that are only used in the loop body. The variables x_1, t_1 can take real values and the variables x_2, t_2 only integer values. It is not hard to see that the reachability relation $(x_1, x_2) \to^+ (y_1, y_2)$ after at least one iteration, where y_1, y_2 denote the new values for x_1, x_2, can be formulated as LIRA formula as follows:

$$x_1 > 0 \wedge x_2 > 0 \wedge y_1 \geq 0.5x_1 + 0.5 \wedge y_2 \leq x_2 - \lfloor 0.5x_1 + 0.5 \rfloor$$

Here, $\lfloor x \rfloor$ returns the greatest integer smaller than or equal to x. Non-termination of the loop can now be expressed using the Ramsey quantified formula

$$\psi := \exists^{\text{ram}} (x_1, x_2), (y_1, y_2) \colon (x_1, x_2) \to^+ (y_1, y_2).$$

Since we require the elements of an infinite clique to be pairwise distinct, we additionally have to check whether there are reachable values for x_1, x_2 from which the same values can be reached again, thereby forming a loop. This additional condition is directly expressible in first-order logic. Now, the while loop in the program does not terminate if and only if ψ is true or some values can be reached multiple times. In this example, ψ is indeed satisfied by setting x_1 to a value strictly between 0 and 1, say 0.5, and x_2 to 1 at the beginning and then always choosing $t_1 = 0.5x_1 + 0.5$ and $t_2 = 0$. This has the effect that t_1 is always in the middle of x_1 and 1, which means that the values for x_1 converge to 1 but never reach 1. Therefore, $\lfloor x_1 \rfloor$ is always 0 and x_2 is constantly 1. Thus, we find the infinite clique $(a_i, b_i)_{i \geq 1}$ with $a_1 := 0.5$, $a_{i+1} := 0.5a_i + 0.5$, and $b_i := 1$ for all $i \geq 1$ in the reachability relation, witnessing satisfiability of ψ. To be able to automatically check the truth value of ψ, we will consider the problem of *eliminating* the Ramsey quantifier by only introducing new existentially quantified variables. Thus, after elimination, we can then use SMT solvers to check satisfiability of a formula in the existential fragment.

Termination proof rules For Algorithm 1 it was easy to manually find a formula for the reachability relation. In practice, instead of expressing the precise reachability relation, one usually tries to find an approximation. For proving termination, an overapproximation of the reachability relation might suffice. Podelski and Rybalchenko [PR04] give a proof rule for termination/liveness, which concerns covering the reachability relation by a finite union of well-founded relations. For simplicity, let us only consider the case of a single well-founded relation T that should cover the reachability relation $\to^+$. Then the verification condition can be stated as follows:

$$\to \subseteq T \quad \wedge \quad T \circ \to \subseteq T \quad \wedge \quad T \text{ is well-founded} \tag{1.1}$$

In this context, well-foundedness of T is defined as the absence of an infinite T-chain, i.e. an infinite sequence $s_1, s_2, \ldots$ such that $(s_i, s_{i+1}) \in T$ for all $i \geq 1$. Clearly, if T is well-founded, then it neither contains a loop, i.e. $(s, s) \in T$ for some s, nor an infinite clique. Here, recall that in our definition of an infinite clique, we demand the elements to be pairwise distinct. Thus, if T satisfies Equation (1.1), then it also satisfies

$$\to \subseteq T \quad \wedge \quad T \circ \to \subseteq T \quad \wedge \quad \text{no } T\text{-loop} \quad \wedge \quad \text{no infinite } T\text{-clique}. \tag{1.2}$$

Since T covers $\to^+$ and $\to^+$ is transitive, Equation (1.2) also implies termination, i.e. that there does not exist an infinite $\to$-chain. However, Equation (1.2) imposes a weaker condition on T than Equation (1.1). To see this, consider for example the transition relation $\to := \{(i+1, i) \mid i \in \mathbb{N}\}$ and the covering $T := \{(i, j) \mid i, j \in \mathbb{N}, i > j\} \cup \{(i, i-1) \mid i \leq 0\}$. Then T satisfies Equation (1.2), which means that $\to$ terminates. However, T is not well-founded, meaning that Equation (1.1) cannot prove termination. More importantly, the advantage of Equation (1.2) is that the conditions can be easily expressed in first-order logic augmented with the Ramsey quantifier, which is a more general solution to the problem that well-foundedness is not a first-order property [CK90] than just adding ad-hoc well-foundedness conditions [Gre+12; BPR13]. The absence of T-loops is

a first-order property definable by $\neg\exists x\colon T(x,x)$ and the absence of an infinite T-clique can be defined with the help of a Ramsey quantifier by $\neg\exists^{\text{ram}} x,y\colon T(x,y)$. Hence, after eliminating the Ramsey quantifier, SMT solvers can be used to automatically check whether a synthesized or provided overapproximation of the reachability relation suffices to prove termination of the program. We remark that some techniques (e.g. [CPR11]) realizing the proof rules of [PR04] synthesize overapproximations that are guaranteed to be well-founded by construction, but these heavily limit the shapes of the constructed relations.

Reachability relations definable in linear arithmetics In general, the reachability relation might not be definable in linear arithmetic, even if the program only uses integer/real variables and linear arithmetic. Minsky machines, for example, are a well-known Turing-complete model that only has access to two integer counters that can be incremented, decremented, and tested for zero [Min67]. Due to their computational power, the reachability problem for Minsky machines is undecidable and the reachability relation is not definable in linear integer arithmetic. However, there is a plethora of natural more restrictive systems where the reachability relation can even be defined *effectively* and *efficiently* in linear arithmetic. Examples are succinct one-counter automata [Li+20], reversal-bounded counter machines with one unrestricted counter [HL11], continuous vector addition systems with states [BH17], and Parikh automata [Sei+04]. For all of these systems, a formula in the existential fragment of linear integer/real arithmetic for the reachability relation can be constructed in polynomial time. Thus, eliminating Ramsey quantifiers from existential formulas efficiently can help solving liveness for all of these systems with precise complexity, even enabling the use of optimized SMT solvers.

Regular reachability relations We have seen above how Ramsey quantifiers can be used to express liveness properties if the reachability relation is definable in some logic. In the context of regular model checking, instead of logical formulas, the reachability relation of many interesting classes of systems can be captured by (tree-)automata. Then liveness problems can be reduced to the evaluation of the Ramsey quantifier over *transitive* (tree-)regular relations. As already mentioned, To and Libkin [TL08] showed that recurrent reachability for transitive (tree-)regular relations is decidable in polynomial time. As an application they obtain that for every system, whose reachability relation is (tree-)regular and computable in polynomial time, recurrent reachability is solvable in polynomial time. As mentioned above, systems that fulfill this property are for example pushdown systems, ground tree rewriting systems, and PA-processes. However, some classes of systems that are considered in regular model checking do not have effectively (tree-)regular reachability relations (e.g. since they are Turing-complete models). In this case, the best one can hope for is to be able to compute (tree-)regular over- or underapproximations that are close enough to the actual reachability relation such that recurrent (non-)reachability can be proved. There exist several semi-decision procedures that use techniques like abstraction and widening to compute such over- or underapproximations (see e.g. [DLS02; BLW03; Abd+04; BHV04; Bou+12; BT12; Leg12]).

Approximations of reachability relations are no longer guaranteed to be transitive. In [To10] an exponential-time upper bound for recurrent reachability over, not necessarily transitive, (tree-)regular relations is given. Understanding the precise complexity of evaluating Ramsey quantifiers over general (tree-)regular relations helps to solve liveness problems if only a (tree-)regular approximation of the reachability relation is provided.

1.3 More Applications of Ramsey Quantifiers

We now turn to further applications of Ramsey quantifiers or, more specifically, the decision version, called *infinite clique problem*, that checks whether $\exists^{\text{ram}} x, y \colon R(x, y)$ holds for a given relation R (e.g. by a formula or an automaton).

Recognizability A k-ary relation R is called *recognizable* if it can be written as $R = \bigcup_{i=1}^{n} L_{i,1} \times \cdots \times L_{i,k}$ for regular languages $L_{i,j}$. Intuitively, the k tapes are independent, in the sense that every tape has its own automaton and the automata are only allowed to communicate at the end of the computation via final states. Whereas in case of regular relations a single automaton controls the heads of all the tapes, but all heads move synchronously. Using a product construction, one can show that the recognizable relations form a subclass of the regular relations. The problem of *recognizability* (of regular relations) asks whether a given regular relation is recognizable. It was shown by Barceló et al. [Bar+19] that this problem is NL-complete or PSPACE-complete depending on whether the regular relation is given as deterministic or nondeterministic automaton, respectively. Their proof uses the well-known fact that a regular relation is recognizable if and only if certain regular equivalence relations have finite index [CCG06]. Checking whether an equivalence relation has infinite index can in turn be reduced to the infinite clique problem over the complement of the equivalence relation. In particular, from their proof one can extract that the infinite clique problem for regular relations is NL-complete (even if the relation is given as nondeterministic automaton). The precise complexity of the analogous problem for tree-regular relations, called *tree-recognizability*, remained open. Only a (double) exponential-time upper bound follows from [To10] if the relation is given by a (non)deterministic tree automaton since the complements of the equivalence relations are not necessarily transitive. Hence, understanding the complexity of the infinite clique problem for tree-regular relations may help to also settle the complexity of tree-recognizability. The *ω-recognizability* problem for ω-regular relations was shown to be decidable in double (resp. triple) exponential time if the relation is given by a deterministic parity automaton (resp. nondeterministic Büchi automaton) [LS19b]. Here, the proof uses the fact that one only has to decide the infinite clique problem for complements of ω-regular equivalence relations. Although it is not known whether the infinite clique problem for arbitrary ω-regular relations is decidable, it is promising to determine the precise complexity in case of these special ω-regular relations in order to make progress on the ω-recognizability problem.

Monadic decomposability In terms of logic, recognizability is related to the problem of *monadic decomposability*. Here, we say that a formula is monadically decomposable if it can be written as a Boolean combination of unary formulas, i.e. formulas with only a single free variable. For example, the formula $\varphi(x, y) := x + y \geq 2 \wedge x \geq 0 \wedge y \geq 0$ in linear integer arithmetic is monadically decomposable since it can be written as

$$(x \geq 2 \wedge y \geq 0) \vee (x \geq 1 \wedge y \geq 1) \vee (x \geq 0 \wedge y \geq 2).$$

Intuitively, as in the recognizability notion, the variables in a monadically decomposable formula are independent from each other. The tight connection of the two problems becomes apparent when considering the (ω-)automatic structures LIA, LRA, and LIRA. Using the characterization via finite-index equivalence relations, one can show that over these structures a formula is monadically decomposable if and only if the corresponding (ω-)regular relation is (ω-)recognizable. Monadic decomposability has a wide range of applications. Essentially, it can help whenever simplification by removing dependencies between variables leads to more efficient algorithms. This is for example the case in the context of constraint databases [GRS99; Cho+03; Lib03], string analysis [Vea+17; Hag+20], theorem proving [KV14], compiler optimization [ASU86], and quantifier elimination [Lib03; Hag+20].

To see how monadic decomposability can help with quantifier elimination (as demonstrated in [Lib03; Hag+20]), let us consider the quantified formula $\varphi := \exists \boldsymbol{x} \colon \psi(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}, \boldsymbol{y}$ are vectors of variables and ψ is quantifier-free. If ψ is monadically decomposable, then, in particular, it can be written in disjunctive form

$$\psi(\boldsymbol{x}, \boldsymbol{y}) \equiv \bigvee_{i=1}^{n} \alpha_i(\boldsymbol{x}) \wedge \beta_i(\boldsymbol{y}),$$

where $\boldsymbol{x}$ is separated from $\boldsymbol{y}$. Now, by moving the existential quantifiers into the disjunction, the elimination of the variables $\boldsymbol{x}$ can be reduced to satisfiability checks:

$$\varphi \equiv \bigvee_{i=1}^{n} (\exists \boldsymbol{x} \colon \alpha_i(\boldsymbol{x})) \wedge \beta_i(\boldsymbol{y}) \equiv \bigvee_{i \colon \alpha_i(\boldsymbol{x}) \text{ is sat}} \beta_i(\boldsymbol{y})$$

Note that to apply the above, it actually suffices that $\boldsymbol{x}$ and $\boldsymbol{y}$ are independent rather than that ψ satisfies the stronger property of being monadically decomposable. This more general notion of independence, where the decomposition conforms to a partition of its variables, is called *variadic decomposability*. A simplification as above can also lead to significant improvement for the projection operation in constraint databases [Lib03].

To enable the use of SMT solvers, Veanes et al. [Vea+17] considered the problem of monadic decomposability for quantifier-free fragments of theories. They developed a generic semi-decision procedure that under some assumptions on the theory, which are for example satisfied by linear integer arithmetic, outputs a monadic decomposition if it exists. However, the generic algorithm does not terminate if the input formula is not monadically decomposable. For linear integer arithmetic and the theory of equality and uninterpreted functions (EUF) it is shown that monadic decomposability in the

quantifier-free fragment is decidable. Thus, for these specific theories the generic algorithm can be made complete by extending it with a termination check. Decidability in case of linear integer arithmetic was already shown by Ginsburg and Spanier [GS66b] in slightly different terms. This also follows from a result by Libkin [Lib03], who provided sufficient conditions on the theory under which the more general problem of variadic decomposability is decidable. These conditions are for example satisfied by LIA, LRA, and non-linear real arithmetic. Precise complexity results, however, were not known until Hague et al. [Hag+20] showed that monadic decomposability is coNP-complete in the quantifier-free fragment of linear integer arithmetic. As mentioned above, monadic decomposability can be characterized via finite-index equivalence relations. Thus, as for recognizability, it might be useful to understand the complexity of the infinite clique problem for formulas in LRA and LIRA in order to determine the precise complexity of monadic decomposability in the two theories.

1.4 Difficulties with Ramsey Quantifiers

In the 70s and 80s, Ramsey quantifiers were mainly studied from a model-theoretic perspective by identifying structures that admit elimination of Ramsey quantifiers. For example, Ramsey quantifiers were proven to be eliminable in theories that do not have the finite cover property [BK80], in complete theories of modules [Bau84], and abelian groups [Bau77]. Some results, however, do not give an effective elimination procedure. In fact, Kierstead and Remmel [KR83] constructed examples of decidable theories that admit elimination of Ramsey quantifiers but where the elimination cannot be made effective. Schmerl and Simpson [SS82] provided an effective elimination procedure of Ramsey quantifiers in linear integer arithmetic. It can be obtained from [DI02] that also linear arithmetic over the reals and LIRA allow effective elimination of Ramsey quantifiers. The same was shown for Archimedian real closed fields and therefore also for non-linear real arithmetic (allowing multiplication) by Cowles [Cow79]. The provided elimination procedures, however, have high complexity, partly since they assume already quantifier-free formulas, and the resulting formulas have multiple quantifier alternations. Thus, in particular, the precise complexity of checking satisfiability of existential formulas with a single leading Ramsey quantifier, as in the setting of liveness discussed above, remained an open problem in all of the mentioned theories.

Ramsey quantifiers in automatic structures The effective elimination procedures for the mentioned decidable theories imply that also the extension of first-order logic with Ramsey quantifiers results in decidable theories. A large class of structures that also enjoy this property is the class of so-called *automatic structures* (see e.g. the survey by Grädel [Grä20]). Loosely speaking, a structure is (word-)automatic if its domain and all of its relations are definable by automata on words. For example, linear integer arithmetic is an automatic structure since the integers can be encoded in binary as words over the alphabet $\{0, 1\}$ and an automaton can check the order relation and perform the standard algorithm for addition. Due to the effective closure properties of regular

languages/relations, it can be observed that the first-order theory of every automatic structure is decidable. Furthermore, it was shown by Rubin [Rub08] that decidability is retained when the logic is extended with Ramsey quantifiers. More precisely, the Ramsey quantifier can be effectively *evaluated* over a regular relation R given as an automaton, i.e. one can construct an automaton accepting all (encodings of) valuations of z such that $\exists^{\mathrm{ram}} x, y \colon R(x, y, z)$ holds. Automatic structures can also be defined such that the automaton model works on trees or infinite words, leading to the notions of tree- and ω-automatic structures, respectively. For example, *Skolem arithmetic*, the structure over the integers with multiplication, is tree-automatic and LRA and LIRA are ω-automatic. It was already observed by Kartzow [Kar11] that the first-order theory of every tree-automatic structure is still decidable when extended with Ramsey quantifiers. In particular, this implies that the infinite clique problem for tree-regular relations is decidable. An exponential-time upper bound was stated by To [To10] in the context of recurrent reachability, which is logspace equivalent to the infinite clique problem. However, it was not clear whether this upper bound is tight and what the complexity of the evaluation of the Ramsey quantifier over tree-regular relations is. In the case of infinite words not much is understood. Decidability of the infinite clique problem for ω-regular relations is a longstanding open problem [Kus10].

1.5 Contributions

We now provide a brief summary of the main results that are developed in this thesis.

Ramsey quantifiers in automatic structures Let us start with Ramsey quantifiers in automatic structures, which was the setting with applications in regular model checking in the motivation above. It already follows from [Bar+19] that the infinite clique problem over regular relations is NL-complete. We will consider the more general setting of evaluating the Ramsey quantifier on a $(2d + k)$-ary regular relation R, i.e. computing an automaton for all valuations of $\boldsymbol{z}$ such that $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ holds, where $\boldsymbol{x}$ and $\boldsymbol{y}$ have dimension d and $\boldsymbol{z}$ has dimension k. We show that given R, one can construct such an automaton in logspace. In particular, this implies that the infinite clique problem (and therefore also recurrent reachability) is in NL, providing a simpler proof than the one in [Bar+19]. For tree-regular relations we prove that if R is given by a deterministic bottom-up tree automaton, then a nondeterministic tree automaton for the evaluation of the Ramsey quantifier can be constructed in polynomial time. It follows that the infinite clique problem for tree-regular relations given as nondeterministic tree automata is solvable in exponential time, which was already stated by To [To10] in the context of recurrent reachability. Moreover, we provide a highly nontrivial matching lower bound using a reduction from intersection non-emptiness of tree automata. This is surprising since an analogous reduction in the word case would yield PSPACE-hardness, but as mentioned above this problem is in NL.

The complexity bounds on the infinite clique problem directly transfer to recurrent reachability. We show that recurrent reachability *with generalized Büchi condition*, i.e.

with multiple target sets that are part of the input, is PSPACE-complete for regular relations and EXP-complete for tree-regular relations. For transitive tree-regular relations (e.g. reachability relations) we show that the Ramsey quantifier can be evaluated in polynomial time instead of exponential time as for arbitrary tree-regular relations. This implies that also recurrent reachability is decidable in polynomial time for transitive tree-regular relations, which was already shown by To and Libkin [TL08]. An exponential-time upper bound for recurrent reachability with generalized Büchi condition for (tree-)regular relations was already shown in [To10] in the transitive case. We show that the PSPACE (resp. EXP) lower bound above already holds for transitive relations.

We further show that over unranked tree-regular relations, the Ramsey quantifier can be evaluated with the same complexity as in the ranked case. As a concrete application, this shows that recurrent reachability (with generalized Büchi condition) is P-complete (resp. EXP-complete) for regular subtree and flat prefix rewriting systems, which were introduced by Löding and Spinrath [LS19b].

As a second major application, besides recurrent reachability, we already identified the problem of tree-recognizability, i.e. the problem of deciding whether a given tree-regular relation can be written as a finite union of Cartesian products of tree-regular languages. We use the observation that there we only need to apply the Ramsey quantifier to *co-transitive* relations, i.e. complements of transitive relations. We show that the infinite clique problem for co-transitive tree-regular relations is P-complete. This allows us to show that tree-recognizability for tree-regular relations R is P-complete if R is given by a deterministic (top-down or bottom-up) tree automaton and EXP-complete if R is given by a nondeterministic tree automaton. Interestingly, applying the generic infinite clique decision procedure for arbitrary tree-regular relations would only give an EXP (resp. 2-EXP) upper bound. This shows the importance of the co-transitivity notion. In the word case, however, the generic decision procedure for the infinite clique problem suffices to show that recognizability is NL-complete (resp. PSPACE-complete) for regular relations given by a (non)deterministic automaton, which was already proven by Barceló et al. [Bar+19].

Ramsey quantifiers in linear arithmetics For relations that are given as existential formulas in linear integer/real arithmetic we consider the problem of eliminating the Ramsey quantifier, i.e. our goal is to compute an existential formula equivalent to

$$\psi := \exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \exists \boldsymbol{w} \colon \gamma(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{z}),$$

where γ is quantifier-free. Recall that, for example, the reachability relation of many classes of systems is expressible in existential linear integer/real arithmetic. As our first step, we deal with the variables $\boldsymbol{w}$ that are existentially quantified within the scope of the Ramsey quantifier. We show that $\boldsymbol{w}$ can be exchanged for linearly many new Ramsey quantified variables while also only linearly increasing the formula size. This is much more efficient than applying quantifier elimination, for which the best known algorithms [Haa+24; CMS24] incur an exponential blow-up if restricted to a single block

of existential quantifiers in linear integer arithmetic. Thus, we can assume that ψ is of the form $\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, where φ is quantifier-free. We then show that if φ is a formula in LIA, LRA, or LIRA, one can construct in polynomial time an existential formula of linear size that is equivalent to ψ, i.e. where the Ramsey quantifier is eliminated.

As already motivated, this has several applications. We consider the *linear liveness problem* that asks for the existence of an infinite run such that between every pair of infinitely many configurations on the run a condition specified in existential linear integer/real arithmetic is satisfied. This is a more general problem than the one considered in [DI02], that only checks for an infinite run such that infinitely many configurations separately satisfy some formula (as in recurrent reachability). With linear liveness one can for example express that some real clock value grows unboundedly on a run rather than being convergent. As already mentioned above, several classes of systems involving counters/clocks possess reachability relations that are definable in linear integer/real arithmetic. For example, using the results that the reachability relation of succinct one-counter automata [Li+20], reversal-bounded counter machines with one unrestricted counter [HL11], continuous vector addition systems with states [BH17], and Parikh automata [Sei+04] can be computed in polynomial time, we show that linear liveness is NP-complete for all of these classes of systems.

A further application is a recently raised question [FG19a] in the context of well-structured transition systems of how to check whether a relation given as a formula in linear integer arithmetic is a well-quasi-ordering (WQO). Our elimination procedure of Ramsey quantifiers allows us to infer that this problem is coNP-complete for quantifier-free LIA formulas.

Recall that the problem of monadic decomposability of formulas in LIA, LRA, and LIRA can be reduced to the infinite clique problem via checking finite index of certain equivalence relations. We will make use of this connection to show that monadic decomposability is coNP-complete for quantifier-free formulas in LIA, LRA, and LIRA. In case of LIA this result was already known [Hag+20], but our approach is asymptotically significantly more efficient.

Recognizability Finally, let us come back to the notion of recognizability that is related to monadic decomposability. We show that ω-recognizability of ω-regular relations is NL-complete or PSPACE-complete depending on whether a deterministic parity or nondeterministic Büchi automaton is given, respectively. This means that the problem has the same complexity as in the finite-word case. Here, we improve on the double (resp. triple) exponential-time upper bound shown in [LS19b]. For finite words we consider the problem for a more general class of relations than regular relations. *Deterministic rational relations* are accepted by deterministic multitape automata, whose heads can move asynchronously. We show that the problem of recognizability in this more general class of relations is in P for binary relations and in coREXP for relations of higher arity. Decidability was shown by Carton, Choffrut, and Grigorieff [CCG06], whose algorithm runs in elementary time for fixed arity, and a double exponential-time upper bound for binary relations follows from [Val75]. However, decidability of the problem of checking

whether a deterministic rational relation is regular remained open in [CCG06] and was also stated as an open problem by Ibarra and Trân [IT12]. We observe that this problem can be reduced to recognizability and show that it is in P for binary relations and in $(2k-4)$-EXP for relations of arity $k > 2$.

1.6 Organization

We start in Chapter 2 by introducing basic concepts from automata theory and logic.

In Chapter 3 we formally define the central notion of this thesis, the Ramsey quantifier, and discuss its connections to liveness verification and recognizability/monadic decomposability.

In Chapter 4 we consider Ramsey quantifiers in (tree-)automatic structures and prove the main results on their evaluation. The word-automatic case is covered in Section 4.2 and the tree-automatic case in Section 4.3. Applications are discussed in Section 4.4. The chapter is concluded with Ramsey quantifiers in unranked tree-automatic structures and a look beyond 2-Ramsey quantifiers (where, loosely speaking, cliques in hypergraphs are considered) in Sections 4.5 and 4.6, respectively.

We then continue with Ramsey quantifiers in linear integer/real arithmetic in Chapter 5, where we start by proving in Section 5.2 how existential quantifiers in the scope of a Ramsey quantifier can be eliminated. This will be used in the elimination procedures of the Ramsey quantifier in LIA, LRA, and LIRA in Sections 5.3, 5.4, and 5.5, respectively. We conclude the chapter with applications in Section 5.6 and experimental results in Section 5.7.

In Chapter 6 we first prove in Section 6.2 the precise complexity result on ω-recognizability of ω-regular relations. In Section 6.3 we then face finite words and consider recognizability in the class of deterministic rational relations. The observation that the problem of regularity of deterministic rational relations can be reduced to recognizability is presented in Section 6.4.

We conclude with a summary and future work in Chapter 7.

2 Preliminaries

Before we start with the contributions, we introduce notations, definitions, and statements that we will encounter at several places in this thesis. After fixing general notations and standard notions from computability and complexity theory in Sections 2.1 and 2.2, respectively, we consider various automata models over finite/infinite words/trees defining languages/relations in Section 2.3. At the end of this chapter, we introduce in Section 2.4 the required notions from logic, specifically first-order logic.

2.1 General Notations

Let us fix some standard notation that will be used throughout this thesis.

Sets For sets S_1 and S_2 we write $S_1 \subseteq S_2$ if S_1 is a (non-strict) subset of S_2. We denote the strict containment by $S_1 \subsetneq S_2$. We use the standard binary set operators union $\cup$, intersection $\cap$, minus $\setminus$, and Cartesian product $\times$. If $S_1 \subseteq S_2$ is clear from the context, we write $\overline{S_1} := S_2 \setminus S_1$ for the complement of S_1. For a set S and $n \geq 0$ we also write S^n for the n-fold product $S \times \cdots \times S$ where S^0 is defined as the set that only contains the empty tuple. Let $2^S := \{S' \mid S' \subseteq S\}$ denote the power set of S. A partition of S is a subset $P = \{B_1, \ldots, B_n\} \subseteq 2^S$ such that $B_1 \cup \cdots \cup B_n = S$ and $B_i \cap B_j = \emptyset$ for all $1 \leq i < j \leq n$. The sets $B_1, \ldots, B_n$ are called blocks. We denote the set of natural numbers by $\mathbb{N} := \{1, 2, \ldots\}$ and $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$. For $i, j \in \mathbb{N}_0$ we often write $[i, j] := \{k \in \mathbb{N}_0 \mid i \leq k \leq j\}$. Furthermore, we use $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$ to denote the set of integers, rationals, and reals, respectively.

Relations A relation is a subset $R \subseteq S_1 \times \cdots \times S_k$ for sets $S_1, \ldots, S_k$. We call R a k-ary relation. For a k-ary relation R and a set $I \subseteq [1, k]$ we let $\pi_I(R)$ denote the projection of R to the components in I. If $I = \{i\}$ is a singleton set, we also write $\pi_i(R)$ for $\pi_I(R)$.

Orders Let $\leq \subseteq S \times S$. The pair $(S, \leq)$ is a *well-quasi-ordering* (WQO) if $\leq$ is reflexive and transitive and additionally every infinite sequence $a_1, a_2, \ldots \in S$ contains an increasing pair $a_i \leq a_j$ with $1 \leq i < j$. A well-quasi-ordering $(S, \leq)$ where $\leq$ is in addition antisymmetic is called a *well-partial-ordering* (WPO). The relation $\leq$ is said to be *well-founded* if there does not exist an infinite strictly decreasing sequence, i.e. an infinite sequence $a_1, a_2, \ldots \in S$ such that $a_{i+1} \leq a_i$ and $a_i \not\leq a_{i+1}$ for all $i \geq 1$. It is well-known that $(S, \leq)$ is a WQO if and only if it is a quasi-ordering, $\leq$ is well-founded, and there does not exist an infinite antichain, i.e. an infinite sequence $a_1, a_2, \ldots \in S$ such that $a_i \not\leq a_j$ for all $i, j \geq 1$ with $i \neq j$. Moreover, if $(S, \leq)$ is a WQO, then every

infinite sequence $a_1, a_2, \ldots \in S$ contains an infinite increasing subsequence, i.e. there are $1 \leq i_1 < i_2 < \ldots$ such that $a_{i_j} \leq a_{i_{j+1}}$ for all $j \geq 1$. This implies that if $(S, \leq)$ is a WPO, then every infinite sequence $a_1, a_2, \ldots \in S$ of pairwise distinct elements contains an infinite strictly increasing subsequence, i.e. there are $1 \leq i_1 < i_2 < \ldots$ such that $a_{i_j} \leq a_{i_{j+1}}$ and $a_{i_{j+1}} \not\leq a_{i_j}$ for all $j \geq 1$.

Tuples and vectors We denote a tuple $(x_1, \ldots, x_k)$ with a boldface letter $\boldsymbol{x}$, where $|\boldsymbol{x}| := k$ is the length of $\boldsymbol{x}$, and for a number n we write $\boldsymbol{n}$ for a tuple $(n, \ldots, n)$ of appropriate length. We also denote an infinite sequence $x_1, x_2, \ldots$ with a boldface letter $\boldsymbol{x}$. Most of the time, we do not distinguish between tuples and vectors and use the terms interchangeably, i.e. we neither distinguish between row and column vectors. Furthermore, for vectors of numbers $\boldsymbol{x}$ and $\boldsymbol{y}$ of dimension (length) k we write $\boldsymbol{x} + \boldsymbol{y} := (x_1 + y_1, \ldots, x_k + y_k)$ for their componentwise sum and $\boldsymbol{x} \cdot \boldsymbol{y} := \sum_{i=1}^{k} x_i \cdot y_i$ for their scalar product. We define the usual componentwise partial order $\boldsymbol{x} \leq \boldsymbol{y}$ such that $x_i \leq y_i$ for all $i \in [1, k]$. Moreover, we define $\boldsymbol{x} \ll \boldsymbol{y}$ if $x_i < y_i$ for every $i \in [1, k]$.

Big $\mathcal{O}$ notation For functions $f, g \colon \mathbb{N}_0 \to \mathbb{R}_{\geq 0}$, where $\mathbb{R}_{\geq 0}$ denotes the set of non-negative real numbers, we write $f \in \mathcal{O}(g)$ if there exists a real number $c > 0$ and a natural number n_0 such that $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$.

2.2 Computability and Complexity

In this section we briefly introduce the notions of computability and complexity theory that are needed in this thesis. For a more detailed introduction to the topic, we refer to the textbooks [Koz97; Sip97; AB09]. In particular, we assume familiarity with the definition of (nondeterministic) Turing machines.

Computation As computation model we assume Turing machines with at least two tapes. The first and the last tape are the designated *input* and *output tape*, respectively. Here, we assume that the input tape is "read only", i.e. the head can only read its content but not modify it, and the output tape is "write once", i.e. starting on the left, the head can only write to it and move to the right without reading its content. The other tapes are called *working tapes*. To measure the space used by a Turing machine, we only count the number of working tape cells visited during the computation. We define the class of functions $\exp_k$ inductively such that $\exp_0$ contains all polynomials in one variable and for $k \geq 1$ we let $\exp_k := \{2^f \mid f \in \exp_{k-1}\}$. We call a finite sequence of symbols over a finite alphabet a *word* and a set of words is called a *language* (see Section 2.3.1 for a formal definition). A deterministic Turing machine $\mathcal{M}$ computes a function on words where $u \mapsto v$ if $\mathcal{M}$ halts on input u having v written on the output tape. We say that the function is computed in

- *logspace* if there is $f \in \mathcal{O}(\log)$ such that M uses at most $f(n)$ space,

- *polynomial time* if there is a polynomial p such that $\mathcal{M}$ halts within at most $p(n)$ steps,
- *polynomial space* if there is a polynomial p such that $\mathcal{M}$ uses at most $p(n)$ space,
- *k-exponential time* if there is $f \in \exp_k$ such that $\mathcal{M}$ halts within at most $f(n)$ steps

on any input of length n. The output size of a logspace computation is polynomially bounded and that of a polynomial-space computation is exponentially bounded. For time-restricted computations the size of the output is bounded by the available time (to write down the output).

Complexity classes A *(decision) problem* is formally a language. We say that a problem (language) is *decidable* if there exists a Turing machine that given an instance (word) as an input, accepts if and only if the input represents a positive instance of the problem (the word is contained in the language). We will use the following standard classes of decision problems, called *complexity classes*:

- NL: Decidable in logspace by a nondeterministic Turing machine
- P: Decidable in polynomial time by a deterministic Turing machine
- NP: Decidable in polynomial time by a nondeterministic Turing machine
- PSPACE: Decidable in polynomial space by a deterministic Turing machine
- k-EXP: Decidable in k-exponential time by a deterministic Turing machine
- k-NEXP: Decidable in k-exponential time by a nondeterministic Turing machine

Here, we simply write EXP and NEXP for 1-EXP and 1-NEXP, respectively. Note that by [Sav70], NPSPACE, i.e. the nondeterministic analogue of PSPACE, is equal to PSPACE. The following inclusions are well-known:

$$\mathsf{NL} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE} \subseteq \mathsf{EXP} \subseteq \mathsf{NEXP} \subseteq \text{2-EXP} \subseteq \dots$$

The inclusions $\mathsf{P} \subsetneq \mathsf{EXP} \subsetneq \text{2-EXP} \subsetneq \dots$ and $\mathsf{NP} \subsetneq \mathsf{NEXP} \subsetneq \text{2-NEXP} \subsetneq \dots$ are known to be strict by the (non)deterministic time hierarchy theorem [HS65; Coo72]. Moreover, by the nondeterministic space hierarchy theorem [Sze94] and since $\mathsf{PSPACE} = \mathsf{NPSPACE}$, we have the strict inclusion $\mathsf{NL} \subsetneq \mathsf{PSPACE}$. Most famously, it is not known whether the inclusion $\mathsf{P} \subseteq \mathsf{NP}$ is strict.

A decidable problem is said to be *elementary* if there exists $k \in \mathbb{N}$ such that it is contained in k-EXP. Otherwise, it is said to be *nonelementary*. For a complexity class C we define the class coC of all problems whose complements are contained in C. Clearly, if C is defined over deterministic Turing machines, then $\mathsf{coC} = \mathsf{C}$ since the answer of a deterministic Turing machine can simply be flipped. It was shown by Immerman [Imm88] and Szelepcsényi [Sze88] that also $\mathsf{NL} = \mathsf{coNL}$.

A *(many-one) reduction* from a problem L to a problem L' is a function mapping instances w to instances w' such that $w \in L$ if and only if $w' \in L'$. A problem L is said to be C-*hard* for a complexity class C if there is a computable (within some time/space bound) reduction from any problem in C to L. Furthermore, if L is C-hard and contained in C, then L is said to be C-complete. We define NL-hardness and P-hardness using logspace reductions. For all other complexity classes mentioned above, we define hardness using polynomial-time reductions.

2.3 Formal Languages, Relations, and Automata

In this thesis we will encounter a variety of different notions of languages and relations that can be defined via automata. We introduce basic definitions and properties of the models in this section and refer to [HU79; Koz97; Sip97; GTW02; Com+08] for more details.

2.3.1 Languages over Finite Words

An *alphabet* is a finite non-empty set Σ of symbols. A *word* over an alphabet Σ is a finite sequence $a_1 \dots a_n$ of symbols $a_i \in \Sigma$. We say that the word $u = a_1 \dots a_n$ has *length* $|u| := n$ and we denote the *empty word*, i.e. the word of length 0, by ε. We write $\Sigma^n := \{a_1 \dots a_n \mid \forall 1 \leq i \leq n \colon a_i \in \Sigma\}$ for the words over Σ of length n where $\Sigma^0 := \{\varepsilon\}$. For two words $u = a_1 \dots a_n$ and $v = b_1 \dots b_m$ over Σ we write $uv := a_1 \dots a_n b_1 \dots b_m$ for their *concatenation*. The *free monoid* over an alphabet Σ is the monoid defined by the set of all words $\Sigma^* := \bigcup_{n \geq 0} \Sigma^n$ over Σ together with the concatenation operation with neutral element ε. A *language* is a subset $L \subseteq \Sigma^*$. We extend the concatenation to languages $L, S \subseteq \Sigma^*$ as $LS := \{uv \mid u \in L, v \in S\}$ and write L^n for the n-fold concatenation of L, where $L^0 := \{\varepsilon\}$, and $L^* := \bigcup_{n \geq 0} L^n$. Furthermore, we define $L^+ := L^* \setminus \{\varepsilon\}$.

For a word $w = a_1 \dots a_n \in \Sigma^*$ with $n \geq 0$ we call any word $a_1 \dots a_j$ with $j \in [0, n]$ a *prefix* of w, any word $a_i \dots a_j$ with $i, j \in [1, n]$ an *infix* of w, any word $a_i \dots a_n$ with $i \in [1, n+1]$ a *suffix* of w, and any word $a_{i_1} \dots a_{i_k}$ with $1 \leq i_1 < \dots < i_k \leq n$ and $k \in [0, n]$ a *subword* of w.

Regular languages A *nondeterministic finite automaton* (NFA) over the alphabet Σ is a tuple $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ where Q is a finite set of states, $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ is a transition relation, $q_0 \in Q$ is an initial state, and $F \subseteq Q$ is a set of final states.

A *run* of $\mathcal{A}$ on a word $w \in \Sigma^*$ is a non-empty sequence of transitions

$$(p_0, a_1, p_1)(p_1, a_2, p_2) \dots (p_{n-1}, a_n, p_n) \in \Delta^*,$$

written as $p_0 \xrightarrow{w} p_n$, such that $w = a_1 \dots a_n$. If $p_0 = p_n$, then $p_0 \xrightarrow{\varepsilon} p_n$ always denotes a run on the empty word. A run is *accepting* if $p_0 = q_0$ and $p_n \in F$. A word $w \in \Sigma^*$ is accepted by $\mathcal{A}$ if there exists an accepting run of $\mathcal{A}$ on w. We denote by $L(\mathcal{A}) \subseteq \Sigma^*$ the language of words that are accepted by $\mathcal{A}$. A language $L \subseteq \Sigma^*$ is said to be *regular* if there exists an NFA $\mathcal{A}$ such that $L = L(\mathcal{A})$.

Remark 2.3.1. Note that for NFAs and all following models of automata, we assume that the alphabet Σ is *not* part of the representation. Instead, in our algorithms we will always work with the subalphabet of Σ that is implicitly given by the transitions. This will be important when the implicitly given alphabet is much smaller than Σ. However, for simpler notation, we will mostly identify the implicitly given alphabet by the full alphabet Σ.

We will also write the transition relation Δ as a (sometimes partial) function $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$. Note that in the above definition we allow $\mathcal{A}$ to have ε-transitions, i.e. $\mathcal{A}$ can change its state along these transitions without consuming any input symbols. However, allowing ε-transitions does not significantly strengthen the model since they can be eliminated as shown in the following proposition (see e.g. [Sip97]).

Proposition 2.3.2. *Given an NFA $\mathcal{A}$, one can compute in polynomial time an NFA $\mathcal{A}'$ without ε-transitions such that $L(\mathcal{A}) = L(\mathcal{A}')$.*

Proof. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$. For a subset $S \subseteq Q$ let $E(S) := \{q' \mid \exists q \in S\colon q \to_\varepsilon^* q'\}$ be the ε-closure of S, where $\to_\varepsilon^*$ denotes the reachability relation in $\mathcal{A}$ that only uses ε-transitions. Now let $E(\mathcal{A}) := (Q, \Sigma, E(\delta), E(\{q_0\}), F)$ where $E(\delta)(q, a) := E(\delta(q, a))$ for all $q \in Q$ and $a \in \Sigma$. Here, the multiple initial states can be replaced with a new unique initial state $q_0' \notin Q$ with $E(\delta)(q_0', a) := \bigcup_{q \in E(\{q_0\})} \delta(q, a)$ for all $a \in \Sigma$. □

A *deterministic finite automaton* (DFA) over the alphabet Σ is an NFA $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ such that the transition relation Δ defines a function $Q \times \Sigma \to Q$.

Note that due to determinism, for every word $w \in \Sigma^*$ there exists a unique run of the DFA $\mathcal{A}$ on w. As discussed in Remark 2.3.1, the transition function of a DFA implicitly defines a subalphabet of Σ, i.e. technically it is a function $Q \times \Sigma' \to Q$ for some alphabet $\Sigma' \subseteq \Sigma$.

It was already observed by Rabin and Scott [RS59] that deterministic finite automata are as powerful as their nondeterministic counterparts since every NFA can be converted to a DFA that accepts the same language. This means that DFAs accept exactly the set of regular languages. The conversion is even effective:

Proposition 2.3.3. *Given an NFA $\mathcal{A}$, one can compute in polynomial space a DFA $\mathcal{A}'$ such that $L(\mathcal{A}) = L(\mathcal{A}')$.*

Proof. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$. By Proposition 2.3.2, we can assume that $\mathcal{A}$ does not contain any ε-transitions. The proof uses the so-called *subset construction*. We define the DFA $\mathcal{A}' := (2^Q, \Sigma, \delta', \{q_0\}, F')$ where $\delta'(S, a) := \bigcup_{q \in S} \delta(q, a)$ for all $S \in 2^Q$ and $a \in \Sigma$ and $F' := \{S \in 2^Q \mid S \cap F \neq \emptyset\}$. □

The class of regular languages enjoys many nice properties. One of them is closure under Boolean operations (union, intersection, complement).

Proposition 2.3.4. *Given NFAs $\mathcal{A}_1$ and $\mathcal{A}_2$, one can compute*

1) *in logspace an NFA $\mathcal{A}$ such that $L(\mathcal{A}) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$. Moreover, the construction preserves determinism.*

2) *in logspace an NFA $\mathcal{A}$ such that $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$. Moreover, the construction preserves determinism.*

3) *in polynomial space a DFA $\mathcal{A}$ such that $L(\mathcal{A}) = \overline{L(\mathcal{A}_1)}$. Moreover, if $\mathcal{A}_1$ is a DFA, then the construction requires only logspace.*

Proof. Let $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$ and $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$ be NFAs with the same implicitly given alphabets. For 1) and 2) we use a *product construction* where we assume that $\mathcal{A}_1$ and $\mathcal{A}_2$ are *complete*, i.e. $\delta_i(q, a) \neq \emptyset$ for all $i \in \{1, 2\}$, $q \in Q_i$, and $a \in \Sigma$ (restricted to the implicitly given subalphabet). In the presence of ε-transitions, we also add ε-self-loops to all states of $\mathcal{A}_1$ and $\mathcal{A}_2$. Note that every NFA can be made complete in logspace by adding a designated error state from which every word is rejected. We define the NFA $\mathcal{A} := (Q_1 \times Q_2, \Sigma, \delta, (q_0^1, q_0^2), F)$ where $\delta((p_1, p_2), a) := \delta_1(p_1, a) \times \delta_2(p_2, a)$ for all $q_1 \in Q_1$, $q_2 \in Q_2$, and $a \in \Sigma$ and $F := F_1 \times Q_2 \cup Q_1 \times F_2$ in case of 1) and $F := F_1 \times F_2$ in case of 2).

To show 3), we first assume that $\mathcal{A}_1$ is a DFA. Then the DFA $\mathcal{A} := (Q_1, \Sigma, \delta_1, q_0^1, Q \setminus F_1)$ accepts the complement of $L(\mathcal{A}_1)$. If $\mathcal{A}_1$ is an NFA, we use Proposition 2.3.3 to determinize $\mathcal{A}_1$ first. □

Note that $\overline{L(\mathcal{A}_1)}$ in Proposition 2.3.4 only refers to the complement with respect to the implicitly given alphabet by $\mathcal{A}_1$.

The *non-emptiness problem* over regular languages asks whether the language accepted by a given finite automaton is non-empty.

Proposition 2.3.5. *Given a regular language L by an NFA or DFA, it is* NL*-complete to decide whether $L \neq \emptyset$.*

Proof. We show that the problem is logspace equivalent to the reachability problem in a directed graph which is well-known to be NL-complete (e.g. see [AB09]). The language accepted by an NFA is non-empty if and only if there exists a path from the initial to the final state (we can assume that NFAs only have one final state). Thus, given an NFA $\mathcal{A}$ for L, we have that $L \neq \emptyset$ if and only if in the directed graph defined by $\mathcal{A}$ (by ignoring the edge labels) the final state is reachable from the initial state. Conversely, given a directed graph G with starting vertex s and target vertex t, we construct an DFA $\mathcal{A}$ with transitions $(p, a_{p,q}, q)$ for every edge (p, q) of G and unique symbol $a_{p,q}$, where the initial state is s and the single final state is t. Then t is reachable from s in G if and only if $L(\mathcal{A}) \neq \emptyset$. □

The *intersection non-emptiness problem* over regular languages asks whether the accepted languages of a given sequence of automata have a non-empty intersection. Again, this problem has the same complexity for NFAs and DFAs, but it is much higher than the complexity of the non-emptiness problem since the number of given automata is part of the input.

Proposition 2.3.6. *Given a sequence $\mathcal{A}_1, \ldots, \mathcal{A}_n$ of NFAs (resp. DFAs), it is* PSPACE*-complete to decide whether $L(\mathcal{A}_1) \cap \cdots \cap L(\mathcal{A}_n) \neq \emptyset$.*

Proof. For the upper bound we first compute an NFA $\mathcal{A}$ for the intersection and then check non-emptiness in nondeterministic logspace given $\mathcal{A}$ using Proposition 2.3.5. Note that this results in a polynomial-space algorithm since the number n of given automata is part of the input and the product construction from Proposition 2.3.4 for the intersection of n automata incurs an exponential blow-up in n but can still be computed in polynomial space.

The lower bound was shown by Kozen [Koz77] and already holds for DFAs. The idea is to compute from a given polynomial-space bounded deterministic Turing machine a sequence of DFAs such that the intersection of their accepted languages is precisely the set of encodings of accepting computations of the given Turing machine. □

The *universality problem* asks whether the complement of a given regular language is empty. The following proposition shows that if the regular language is given by an NFA, then the exponential blow-up for the complementation of NFAs is in some sense unavoidable.

Proposition 2.3.7. *Given a regular language L, deciding whether $\overline{L} = \emptyset$ is* PSPACE*-complete if L is given by an NFA and* NL*-complete if L is given by a DFA.*

Proof. The upper bounds follow from Proposition 2.3.4 by computing an automaton for the complement and checking with Proposition 2.3.5 whether the complement is empty. P-hardness for DFAs follows from a reduction from the emptiness problem, which by Proposition 2.3.5 is NL-complete, since a DFA for the complement can be computed in logspace using Proposition 2.3.4. The PSPACE lower bound for NFAs can be shown by a reduction from non-acceptance of polynomial-space bounded deterministic Turing machines and was first shown by Meyer and Stockmeyer [MS72]. □

2.3.2 Languages over Infinite Words

In the following we introduce the notions of automata over infinite words that are needed in this thesis. For a more detailed overview we refer to [GTW02].

An *infinite word* over an alphabet Σ is an infinite sequence $a_1 a_2 \dots$ of symbols $a_i \in \Sigma$. We denote by Σ^ω the set of all infinite words over Σ. An *ω-language* is a subset $L \subseteq \Sigma^\omega$.

ω-regular languages A *nondeterministic Büchi automaton* (NBA) over the alphabet Σ has the same format as an NFA $\mathcal{B} = (Q, \Sigma, \Delta, q_0, F)$. A *run* of $\mathcal{B}$ on an infinite word $w \in \Sigma^\omega$ is an infinite sequence of transitions $(p_0, a_1, p_1)(p_1, a_2, p_2) \dots \in \Delta^\omega$, written as $p_0 \xrightarrow{w} p_n$, such that $w = a_1 a_2 \dots$. A run is *accepting* if $p_0 = q_0$ and $p_i \in F$ for infinitely many $i \geq 0$. We denote by $L(\mathcal{B}) \subseteq \Sigma^\omega$ the language of infinite words that are accepted by $\mathcal{B}$, i.e. on which there exists an accepting run of $\mathcal{B}$. An ω-language $L \subseteq \Sigma^\omega$ is said to be *ω-regular* if there exists an NBA $\mathcal{B}$ such that $L = L(\mathcal{B})$.

Unlike in the finite-word case, *deterministic Büchi automata* (DBA), where Δ defines a function from $Q \times \Sigma$ to Q, are less expressive than nondeterministic ones. However, there are deterministic models that capture the whole class of ω-regular languages. One of them are deterministic parity automata. A *deterministic parity automaton* (DPA)

over the alphabet Σ is a tuple $\mathcal{B} = (Q, \Sigma, \Delta, q_0, c)$ where Q, Δ, and q_0 are as in a DBA and $c\colon Q \to \{1, \dots, k\}$ for $k \in \mathbb{N}$ is a coloring function of states. For a run ρ of $\mathcal{B}$ on an infinite word let $\mathrm{Inf}(\rho) \subseteq Q$ denote the set of states that occur infinitely often in ρ. The run ρ is accepting if $\min\{c(q) \mid q \in \mathrm{Inf}(\rho)\}$ is even.

As already mentioned, for every NBA there exists a DPA that accepts the same ω-language. Construction is even effective and an exponential-time upper bound was first given by Safra [Saf88], known as Safra's construction.

Proposition 2.3.8. *Given an NBA $\mathcal{B}$, one can compute in exponential time a DPA $\mathcal{B}'$ such that $L(\mathcal{B}) = L(\mathcal{B}')$.*

Note that Safra's construction originally produces a deterministic Rabin automaton, for which the construction is optimal, but the Rabin acceptance condition can easily be converted to a parity acceptance condition. Construction in the reverse direction is much simpler [KKV01].

Proposition 2.3.9. *Given a DPA $\mathcal{B}$, one can compute in logspace an NBA $\mathcal{B}'$ such that $L(\mathcal{B}) = L(\mathcal{B}')$.*

The class of ω-regular languages is closed under Boolean operations. For union and intersection constructions are similar as in the finite-word case. Complementation of an ω-regular language given as a DPA is also easy by assigning every state p the color $c(p) + 1$. If we are given an NBA instead, an NBA for the complement can be computed in polynomial space [SVW87].

Proposition 2.3.10. *Given NBAs (resp. DPAs) $\mathcal{B}_1$ and $\mathcal{B}_2$, one can compute*

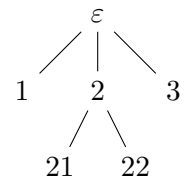
1) *in logspace an NBA (resp. DPA) $\mathcal{B}$ such that $L(\mathcal{B}) = L(\mathcal{B}_1) \cup L(\mathcal{B}_2)$.*
2) *in logspace an NBA (resp. DPA) $\mathcal{B}$ such that $L(\mathcal{B}) = L(\mathcal{B}_1) \cap L(\mathcal{B}_2)$.*
3) *in polynomial space (resp. logspace) an NBA (resp. DPA) $\mathcal{B}$ such that $L(\mathcal{B}) = \overline{L(\mathcal{B}_1)}$.*

We call an infinite word $w \in \Sigma^\omega$ *ultimately periodic* if it can be written as $w = uv^\omega$ for finite words $u, v \in \Sigma^*$ with $|v| > 0$. Here, v^ω denotes the infinite concatenation of v with itself. The following lemma is well-known.

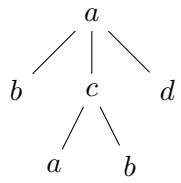
Lemma 2.3.11. *Every non-empty ω-regular language contains an ultimately periodic word.*

Proof. Let $\mathcal{B} = (Q, \Sigma, \Delta, q_0, F)$ be an NBA that accepts at least one infinite word. Let $w = a_1 a_2 \ldots \in L(\mathcal{B})$ and $\rho = (p_0, a_1, p_1)(p_1, a_2, p_2)\ldots$ be an accepting run of $\mathcal{B}$ on w. Since ρ is accepting, some final state $q \in F$ must appear infinitely often in ρ, i.e. there are $0 \leq i_1 < i_2 < \ldots$ such that $p_{i_j} = q$ for all $j \geq 1$. Now, the ultimately periodic word $a_1 \ldots a_{i_1}(a_{i_1+1} \ldots a_{i_2})^\omega$ is accepted by $\mathcal{B}$. □

We can use Lemma 2.3.11 to decide the *non-emptiness problem* for ω-regular languages.



(a) An arranged tree domain.



(b) A tree with tree domain as in (a).

Figure 2.1: Illustration of a tree domain and a corresponding tree. We also write a tree as in (b) as a string $a(b, c(a, b), d)$.

Proposition 2.3.12. *Given an NBA $\mathcal{B}$, it is* NL*-complete to decide whether $L(\mathcal{B}) \neq \emptyset$.*

Proof. Let $\mathcal{B} = (Q, \Sigma, \Delta, q_0, F)$ be an NBA. By Lemma 2.3.11, it suffices to check whether $\mathcal{B}$ accepts an ultimately periodic word. To this end, we nondeterministically guess a final state $q \in F$ and check whether q is reachable from q_0 and q itself. This can be done in nondeterministic logspace since reachability in a directed graph is in NL (see e.g. [AB09]).

Note that NL-hardness follows from Proposition 2.3.5 by adding an infinite padding to finite words. □

2.3.3 Languages over Finite Trees

Recall that we formally only defined words over a finite alphabet, but we can generalize the definition to infinite alphabets in the natural way, which allows us to write $\mathbb{N}^*$ for the set of all words over the natural numbers. A *tree domain* is a non-empty set $D \subseteq \mathbb{N}^*$ such that

(i) D is prefix closed, i.e. $uv \in D$ implies $u \in D$,

(ii) for all $v \in \mathbb{N}^*$ and $1 \leq j \leq i$ if $vi \in D$, then $vj \in D$, and

(iii) each node $v \in D$ has only finitely many *children* $vi \in D$ where $i \in \mathbb{N}$.

An *unranked tree* over an alphabet Σ is a function $t \colon \mathrm{dom}(t) \to \Sigma$ where $\mathrm{dom}(t)$ is a finite tree domain. A *ranked alphabet* is a finite alphabet Σ where every symbol $a \in \Sigma$ has a rank $\mathrm{rk}(a) \geq 0$. A *ranked tree* is an unranked tree t such that every node $v \in \mathrm{dom}(t)$ has $\mathrm{rk}(t(v))$ many children. We denote the set of all ranked and unranked trees over Σ by $\mathcal{T}_\Sigma$ and $\mathcal{U}_\Sigma$, respectively. We call subsets of $\mathcal{T}_\Sigma$ and $\mathcal{U}_\Sigma$ *tree languages*. An example of a tree and the corresponding tree domain is illustrated in Figure 2.1.

Tree-regular languages A *nondeterministic (top-down) tree automaton* (NTA) over the ranked alphabet Σ is a tuple $\mathcal{A} = (Q, \Sigma, \Delta, q_0)$ where Q is a finite set of states, $q_0 \in Q$ is an initial state, and $\Delta \subseteq \bigcup_{a \in \Sigma} Q \times \{a\} \times Q^{\mathrm{rk}(a)}$ is a transition relation. A *run* of $\mathcal{A}$ on a tree $t \in \mathcal{T}_\Sigma$ is a tree $\rho \in \mathcal{U}_Q$ with $\mathrm{dom}(\rho) = \mathrm{dom}(t)$ such that $(\rho(u), t(u), \rho(u1), \ldots, \rho(ur)) \in \Delta$ for all nodes $u \in \mathrm{dom}(\rho)$ with $\mathrm{rk}(t(u)) = r$. The

run ρ is *accepting* if $\rho(\varepsilon) = q_0$. As before, $L(\mathcal{A})$ denotes the tree language of ranked trees accepted by $\mathcal{A}$, i.e. the set of all trees $t \in \mathcal{T}_\Sigma$ such that there exists an accepting run of $\mathcal{A}$ on t. A tree language is called *tree-regular* if it is accepted by some NTA.

We define a *deterministic top-down tree automaton* (D↓TA) as an NTA where the transition relation Δ defines a partial function from $Q \times \Sigma$ to $\bigcup_{r \geq 0} Q^r$. However, D↓TAs are strictly less expressive than NTAs. For example, consider the tree-regular language L consisting of two trees $a(b, c)$ and $a(c, b)$. Any D↓TA accepting L would also accept the trees $a(b, b)$ and $a(c, c)$, a contradiction. To be able to accept all tree-regular languages with a deterministic model, we need to process the input tree bottom-up. A *deterministic bottom-up tree automaton* (D↑TA) over the ranked alphabet Σ is a tuple $\mathcal{A} = (Q, \Sigma, \Delta, F)$ where Q is a finite set of states, $F \subseteq Q$ is a set of final states, and $\Delta \subseteq \bigcup_{a \in \Sigma} Q^{\mathrm{rk}(a)} \times \{a\} \times Q$ is a transition relation that defines a function from $(\bigcup_{r \geq 0} Q^r) \times \Sigma$ to Q. A *run* of $\mathcal{A}$ on a tree $t \in \mathcal{T}_\Sigma$ is a tree $\rho \in \mathcal{T}_Q$ with $\mathrm{dom}(\rho) = \mathrm{dom}(t)$ such that $(\rho(u1), \ldots, \rho(ur), t(u), \rho(u)) \in \Delta$ for all nodes $u \in \mathrm{dom}(\rho)$ with $\mathrm{rk}(t(u)) = r$. The run ρ is *accepting* if $\rho(\varepsilon) \in F$. Note that similar to DFAs, we technically assume that the transition relation of a D↑TA defines a function with respect to the implicitly given subalphabet.

It is easy to see that every D↑TA can be converted to an NTA accepting the same language by reverting the transitions and adding a new single initial state that simulates all final states of the D↑TA.

Proposition 2.3.13. *Given a D↑TA $\mathcal{A}$, one can compute in logspace an NTA $\mathcal{A}'$ such that $L(\mathcal{A}) = L(\mathcal{A}')$.*

For the reverse direction we can use a subset construction similarly as in the conversion from NFA to DFA. This, however, results in an exponential blow-up. We refer to [Com+08] for more details.

Proposition 2.3.14. *Given an NTA $\mathcal{A}$, one can compute in polynomial space a D↑TA $\mathcal{A}'$ such that $L(\mathcal{A}) = L(\mathcal{A}')$.*

Note that there is also the notion of nondeterministic bottom-up tree automata that is equivalent to NTAs by reverting the transitions.

The class of tree-regular relations is closed under Boolean operations. The proof is similar to the one in the word case, i.e. it uses a product construction for union and intersection and Proposition 2.3.14 for the complement.

Proposition 2.3.15. *Given NTAs (resp. D↑TAs) $\mathcal{A}_1$ and $\mathcal{A}_2$, one can compute*

1) in logspace an NTA (resp. D↑TA) $\mathcal{A}$ such that $L(\mathcal{A}) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$.

2) in logspace an NTA (resp. D↑TA) $\mathcal{A}$ such that $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.

3) in polynomial space (resp. logspace) a D↑TA $\mathcal{A}$ such that $L(\mathcal{A}) = \overline{L(\mathcal{A}_1)}$.

As over words, the complement is with respect to the implicitly given alphabet. We remark that the class of tree languages accepted by D↓TAs is closed under intersection

(using the product construction) but not under union and complement. To see this, we again consider the tree-regular language L consisting of the trees $a(b,c)$ and $a(c,b)$. As observed above, L cannot be accepted by any D↓TA. However, L is the union of two singleton tree languages that can be accepted by D↓TAs. Moreover, by closure under intersection, this implies that the class of languages accepted by D↓TAs can neither be closed under complement. However, we can efficiently compute an NTA for the complement.

Proposition 2.3.16. *Given a D↓TA $\mathcal{A}$, one can compute in logspace an NTA $\mathcal{A}'$ such that $L(\mathcal{A}') = \overline{L(\mathcal{A})}$.*

Proof. Let $\mathcal{A} = (Q, \Sigma, \Delta, q_0)$ be a D↓TA and assume that Σ is the implicitly given alphabet. We construct an NTA $\mathcal{A}' = (Q', \Sigma, \Delta', q_0)$ that nondeterministically guesses one branch of the tree and verifies that it is not accepted by $\mathcal{A}$. We define $Q' := Q \cup \{q_{\text{acc}}\}$ where $q_{\text{acc}} \notin Q$ is a fresh state from which every tree will be accepted. For all $q \xrightarrow{a} (q_1, \dots, q_r)$ in Δ with $r > 0$ we let $q \xrightarrow{a} (q_{\text{acc}}, \dots, q_{\text{acc}}, q_i, q_{\text{acc}}, \dots, q_{\text{acc}})$ be in Δ'. Furthermore, for all $q \in Q$ and $a \in \Sigma$ with $\text{rk}(a) = r \geq 0$ such that there do not exist $q_1, \dots, q_r \in Q$ with $q \xrightarrow{a} (q_1, \dots, q_r)$ in Δ we let $q \xrightarrow{a} (q_{\text{acc}}, \dots, q_{\text{acc}})$ be in Δ'. Finally, for all $a \in \Sigma$ we let $q_{\text{acc}} \xrightarrow{a} (q_{\text{acc}}, \dots, q_{\text{acc}})$ be in Δ'. □

The *non-emptiness problem* for tree-regular languages has the same complexity for NTAs, D↑TAs, and D↓TAs.

Proposition 2.3.17. *Given a tree-regular language L by an NTA, D↑TA, or D↓TAs, it is* P*-complete to decide whether $L \neq \emptyset$.*

Proof. Let $\mathcal{A} = (Q, \Sigma, \Delta, q_0)$ be an NTA and $S \subseteq Q$ be initially empty. We will iteratively add states to S from which there exists a run of $\mathcal{A}$ on some tree. First, we add for every transition $(q, a) \in \Delta$ the state q to S. Then we check for every transition $(q, a, q_1, \dots, q_r) \in \Delta$ whether $q_1, \dots, q_r \in S$ and if so, we add q to S. We repeat this until either q_0 is added to S, in which case $L(\mathcal{A}) \neq \emptyset$, or S does not change anymore (and does not contain q_0), in which case $L(\mathcal{A}) = \emptyset$.

The P lower bound already holds for D↑TAs and D↓TAs (see [Com+08]). □

Similar to the word case, the complexity of the *intersection non-emptiness problem* over tree-regular languages is much higher than the complexity of the non-emptiness problem since the number of given tree automata is part of the input.

Proposition 2.3.18. *Given a sequence $\mathcal{A}_1, \dots, \mathcal{A}_n$ of NTAs (resp. D↑TAs, D↓TAs), it is* EXP*-complete to decide whether $L(\mathcal{A}_1) \cap \dots \cap L(\mathcal{A}_n) \neq \emptyset$.*

Proof. For the upper bound we first compute an NTA $\mathcal{A}$ for the intersection and then check non-emptiness in time polynomial in the size of $\mathcal{A}$ using Proposition 2.3.17. Note that this results in an exponential-time algorithm since the number n of given tree automata is part of the input and the product construction from Proposition 2.3.15 for the intersection of n tree automata incurs an exponential blow-up in n.

The lower bound for NTAs was shown by Frühwirth et al. [Frü+91]. It already holds for D↑TAs and D↓TAs as proved in [Sei94]. The idea is to encode the computation of a linear-space bounded alternating Turing machine as a tree and check with a sequence of tree automata whether a tree corresponds to an accepting computation. Hence, the intersection of the tree langauges accepted by the sequence of tree automata is non-empty if and only if the Turing machine accepts. □

For the *universality problem*, i.e. emptiness of the complement, one can show that the exponential blow-up for the complementation of NTAs is in some sense unavoidable.

Proposition 2.3.19. *Given a tree-regular language L, it is* EXP*-complete if L is given by an NTA and* P*-complete if L is given by a D↑TA or D↓TA to decide whether $\overline{L} = \emptyset$.*

Proof. The upper bounds follow from Proposition 2.3.15 (resp. Proposition 2.3.16 for D↓TAs) by computing an automaton for the complement and then checking with Proposition 2.3.17 whether the complement is empty. P-hardness for D↑TAs and D↓TAs follows by a logspace reduction using Propositions 2.3.15 and 2.3.16 from the emptiness problem, which is P-hard by Proposition 2.3.17. The EXP-lower bound for NTAs can be shown by a reduction from non-acceptance of linear-space bounded alternating Turing machines. □

An *alternating tree automaton* (ATA) over the ranked alphabet Σ is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0)$ where Q is a finite set of states, $q_0 \in Q$ is an initial state, and $\delta\colon Q \times \Sigma \to \mathcal{B}^+(Q \times \mathbb{N})$ is a transition function with $\delta(q, a) \in \mathcal{B}^+(Q \times \{1, \ldots, \mathrm{rk}(a)\})$ for all $q \in Q$ and $a \in \Sigma$. Here, $\mathcal{B}^+(Q \times \mathbb{N})$ denotes the set of positive propositional formulas (i.e. only using $\wedge$ and $\vee$) over the set of variables $Q \times \mathbb{N}$. A formula without any variables is either true, denoted by $\top$, or false, denoted by $\bot$. For a set $S \subseteq Q \times \mathbb{N}$ of variables and formula φ we denote by $S \models \varphi$ that if the variables in S are set to true and the variables not in S are set to false, then φ is satisfied. For this, the semantics of propositional formulas is defined as usual.

A *run* of $\mathcal{A}$ on a tree $t \in \mathcal{T}_\Sigma$ is an unranked tree ρ over the alphabet $Q \times \mathrm{dom}(t)$ such that $\rho(\varepsilon) = (q, \varepsilon)$ for some $q \in Q$ and for each node $u \in \mathrm{dom}(\rho)$ with $r \geq 0$ children and $\rho(u) = (q, w)$

- there is a set $S = \{(q_1, c_1), \ldots, (q_r, c_r)\} \subseteq Q \times \{1, \ldots, \mathrm{rk}(t(w))\}$ such that
- $S \models \delta(q, t(w))$ and
- $\rho(ui) = (q_i, wc_i)$ for all $i \in [1, r]$.

The run ρ is *accepting* if $\rho(\varepsilon) = (q_0, \varepsilon)$. Again, the tree language $L(\mathcal{A})$ accepted by $\mathcal{A}$ consists of all trees on which there is an accepting run of $\mathcal{A}$.

Intuitively, an ATA can create at every node of the input tree multiple copies of itself and start them in some state at some child. Then it checks whether the set of copies whose runs were accepting satisfies the formula given by the transition. Note that if w is a leaf, then the run checks for the corresponding run node (q, w) whether $\delta(q, t(w)) = \top$.

An NTA can be seen as a special ATA where for all $q \in Q$ and $a \in \Sigma$ the transition formula $\delta(q, a)$ is a disjunction of conjunctions $\bigwedge_{i=1}^{\mathrm{rk}(a)}(q_i, i)$. Conversely, every ATA can be converted to an NTA, i.e. ATAs accept precisely the tree-regular languages. However, ATAs are more succinct.

Proposition 2.3.20. *Given an ATA $\mathcal{A}$, one can compute in polynomial space a D↑TA $\mathcal{A}'$ such that $L(\mathcal{A}) = L(\mathcal{A}')$.*

Proof. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0)$. We use a subset construction to define the D↑TA $\mathcal{A}' := (2^Q, \Sigma, \delta', F')$ with $F' := \{S \in 2^Q \mid q_0 \in S\}$ and

$$\delta'(S_1, \ldots, S_r, a) := \{q \in Q \mid S_1 \times \{1\} \cup \cdots \cup S_r \times \{r\} \models \delta(q, a)\}$$

for all $a \in \Sigma$ with $r := \mathrm{rk}(a)$ and $S_1, \ldots, S_r \in 2^Q$. We refer to [Com+08] for the correctness proof by induction. □

We consider ATAs due to their efficient closure under Boolean operations even if the number of given automata is part of the input.

Proposition 2.3.21. *Given ATAs $\mathcal{A}_1, \ldots, \mathcal{A}_n$, one can compute in linear time ATAs for $\bigcup_{i=1}^n L(\mathcal{A}_i)$, $\bigcap_{i=1}^n L(\mathcal{A}_i)$, and $\overline{L(\mathcal{A}_1)}$.*

Proof. For every $i \in [1, n]$ let $\mathcal{A}_i = (Q_i, \Sigma, \delta_i, q_0^i)$, where we assume that $\bigcap_{i=1}^n Q_i = \emptyset$. For union and intersection we define the ATA $\mathcal{A} := (Q, \Sigma, \delta, q_0)$ where $Q := \{q_0\} \cup \bigcup_{i=1}^n Q_i$ for a fresh initial state $q_0 \notin \bigcup_{i=1}^n Q_i$ and for all $a \in \Sigma$ let $\delta(q_i, a) := \delta_i(q_i, a)$ for all $i \in [1, n]$ and $q_i \in Q_i$ and $\delta(q_0, a) := \delta_1(q_0^1, a) * \cdots * \delta_n(q_0^n, a)$, where $* := \vee$ for union and $* := \wedge$ for intersection.

To accept the complement of $L(\mathcal{A}_1)$, we exchange $\wedge$ and $\vee$ (resp. $\top$ and $\bot$) in the transitions of $\mathcal{A}_1$. □

2.3.4 Languages over Infinite Trees

We define *infinite unranked trees* and *infinite ranked trees* as in the finite case but with infinite domains, i.e. as functions $t\colon \mathrm{dom}(t) \to \Sigma$ for an infinite tree domain $\mathrm{dom}(t)$ and an (ranked) alphabet Σ. Recall that by the definition of tree domains, every node, even in unranked infinite trees, can only have finitely many children (a.k.a. *finitely branching*). We denote the set of all finite and infinite unranked trees over the alphabet Σ by $\mathcal{U}_\Sigma^\infty$ and the set of all finite and infinite ranked trees over the ranked alphabet Σ by $\mathcal{T}_\Sigma^\infty$. We will use the tree version of a statement shown by Kőnig [Kőn27].

Lemma 2.3.22 (Kőnig's Lemma)**.** *Every finitely branching infinite tree contains an infinite path.*

A *nondeterministic Büchi tree automaton* (NBTA) over the ranked alphabet Σ is a tuple $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ where Q, Δ, and q_0 are as in the definition of an NTA and $F \subseteq Q$ is a set of final states. A *run* of $\mathcal{A}$ on a finite or infinite ranked tree $t \in \mathcal{T}_\Sigma^\infty$ is a tree $\rho \in \mathcal{T}_Q^\infty$ with $\mathrm{dom}(\rho) = \mathrm{dom}(t)$ such that $(\rho(u), t(u), \rho(u1), \ldots, \rho(ur)) \in \Delta$

for all nodes $u \in \mathrm{dom}(\rho)$ with $\mathrm{rk}(t(u)) = r$. The run ρ is *accepting* if $\rho(\varepsilon) = q_0$ and every infinite path of ρ contains infinitely many nodes u with $\rho(u) \in F$. As usual, $L(\mathcal{A})$ denotes the tree language accepted by the NBTA $\mathcal{A}$.

Note that in the literature the acceptance condition of NBTAs is sometimes defined such that *every* path contains infinitely many nodes with labels in F (see e.g. [Tho90]). In this case only trees where every path is infinite can be accepted. Clearly, our definition is a generalization since we also allow trees with finite paths. Conversely, every set of finite or infinite trees can be transformed to a set of trees with only infinite paths by padding every finite path with an infinite chain of nodes labeled with a fresh padding symbol. Thus, the two models can simulate each other.

The class of tree languages accepted by NBTAs is closed under union and intersection [Rab70].

Proposition 2.3.23. *Given NBTAs $\mathcal{A}_1$ and $\mathcal{A}_2$, one can compute in logspace an NBTA for $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$ and $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.*

Proof. For the union we can just add a new initial state that combines the transitions of the initial states of $\mathcal{A}_1$ and $\mathcal{A}_2$. Here, we assume that the state sets of $\mathcal{A}_1$ and $\mathcal{A}_2$ are disjoint.

For the intersection we use a product construction where we additionally check that on infinite paths final states of both $\mathcal{A}_1$ and $\mathcal{A}_2$ are visited infinitely often. Let $\mathcal{A}_1 = (Q_1, \Sigma, \Delta_1, q_0^1, F_1)$ and $\mathcal{A}_2 = (Q_2, \Sigma, \Delta_2, q_0^2, F_2)$. We construct $\mathcal{A} := (Q, \Sigma, \Delta, q_0, F)$ for $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ with $Q := Q_1 \times Q_2 \times \{1, 2\}$, $q_0 := (q_0^1, q_0^2, 1)$, and $F := Q_1 \times F_2 \times \{2\}$ and Δ contains for all $a \in \Sigma$ with $r := \mathrm{rk}(a)$ the transitions

- $(p, q, 1) \xrightarrow{a} ((p_1, q_1, 1), \ldots, (p_r, q_r, 1))$ for all $(p, a, p_1, \ldots, p_r) \in \Delta_1$ with $p \notin F_1$ and $(q, a, q_1, \ldots, q_r) \in \Delta_2$,
- $(p, q, 1) \xrightarrow{a} ((p_1, q_1, 2), \ldots, (p_r, q_r, 2))$ for all $(p, a, p_1, \ldots, p_r) \in \Delta_1$ with $p \in F_1$ and $(q, a, q_1, \ldots, q_r) \in \Delta_2$,
- $(p, q, 2) \xrightarrow{a} ((p_1, q_1, 2), \ldots, (p_r, q_r, 2))$ for all $(p, a, p_1, \ldots, p_r) \in \Delta_1$ and $(q, a, q_1, \ldots, q_r) \in \Delta_2$ with $q \notin F_2$,
- $(p, q, 2) \xrightarrow{a} ((p_1, q_1, 1), \ldots, (p_r, q_r, 1))$ for all $(p, a, p_1, \ldots, p_r) \in \Delta_1$ and $(q, a, q_1, \ldots, q_r) \in \Delta_2$ with $q \in F_2$.

Intuitively, the third component indicates whether we want to see a final state of $\mathcal{A}_1$ or of $\mathcal{A}_2$ next. □

Note that, as shown by Rabin [Rab70], the class of tree languages accepted by NBTAs is not closed under complement. For example, the language of trees containing a path with infinitely many a, for a fixed symbol a, can be accepted by an NBTA, but its complement, i.e. the language of trees where all paths contain only finitely many a, cannot be accepted by any NBTA.

As in the finite tree case, the *non-emptiness problem* for tree languages given by NBTAs is P-complete.

Proposition 2.3.24. *Given an NBTA $\mathcal{A}$, it is* P*-complete to decide whether* $L(\mathcal{A}) \neq \emptyset$.

Proof. Let $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ be an NBTA. For a subset $Q' \subseteq Q$ and state $q \in Q'$ we define the NTA $\mathcal{A}'_{Q',q} := (Q', \Sigma_\#, \Delta', q)$ where $\Sigma_\# := \Sigma \cup \{\#\}$ with $\# \notin \Sigma$ and $\mathrm{rk}(\#) := 0$ and $\Delta' := \Delta \cup \{(f, \#) \mid f \in F \cap Q'\}$. The algorithm proceeds iteratively. At the beginning let $Q' := Q$. In each step use Proposition 2.3.17 to check for all $q \in Q'$ whether $L(\mathcal{A}'_{Q',q}) \neq \emptyset$. Then the new Q' consists of all states for which this check was successful. Repeat this process until Q' does not change anymore. It is easy to verify that at the end $q_0 \in Q'$ if and only if $L(\mathcal{A}) \neq \emptyset$. Moreover, the algorithm clearly runs in polynomial time using Proposition 2.3.17. We refer to [Rab70] for more details.

The lower bound is inherited from the non-emptiness problem for tree-regular languages given by NTAs, which by Proposition 2.3.17 is P-complete. □

2.3.5 Relations over Finite Words

Let Σ be a finite alphabet. The product of k free monoids $(\Sigma^*)^k$ forms a monoid with componentwise multiplication $(u_1, \ldots, u_k)(v_1, \ldots, v_k) = (u_1 v_1, \ldots, u_k v_k)$. We often denote word tuples by boldface letters $\boldsymbol{u}$ and denote its i-th entry by u_i. As usual, we identify a pair of tuples $(\boldsymbol{u}, \boldsymbol{v})$ with the concatenation of $\boldsymbol{u}$ and $\boldsymbol{v}$. Furthermore, $\boldsymbol{\varepsilon} = (\varepsilon, \ldots, \varepsilon)$ denotes a tuple of empty words of appropriate dimension. The *length* of a word tuple $\|\boldsymbol{u}\| = \sum_{i=1}^{k} |u_i|$ is the combined length of its entries.

Rational relations A *k-tape automaton* $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ over the alphabet Σ consists of a finite state set Q, an initial state q_0, a set $F \subseteq Q$ of final states, and a finite set of transitions $\Delta \subseteq Q \times (\Sigma^*)^k \times Q$.

A run of $\mathcal{A}$ on a tuple $\boldsymbol{w} \in (\Sigma^*)^k$ is a non-empty sequence of transitions

$$(p_0, \boldsymbol{w}_1, p_1)(p_1, \boldsymbol{w}_2, p_2) \ldots (p_{n-1}, \boldsymbol{w}_n, p_n) \in \Delta^*,$$

written as $p_0 \xrightarrow{\boldsymbol{w}} p_n$, such that $\boldsymbol{w} = \boldsymbol{w}_1 \ldots \boldsymbol{w}_n$. If $p_0 = p_n$, we always call $p_0 \xrightarrow{\boldsymbol{\varepsilon}} p_n$ a run on the tuple of empty words. The relation $R(\mathcal{A})$ accepted by $\mathcal{A}$ consists of all tuples $\boldsymbol{w} \in (\Sigma^*)^k$ such that $\mathcal{A}$ has an accepting run on $\boldsymbol{w}$, i.e. a run from the initial to a final state. Relations accepted by k-tape automata are called *rational* and we denote the class of rational relations by **Rat**.

Note that 1-tape automata are a generalization of NFAs where the automaton is allowed to read more than one symbol at a time. It is easy to see that those two models are logspace equivalent.

Intuitively, a k-tape automaton (or multitape automaton if the number of tapes is not relevant) has access to multiple tapes through a designated read head for each tape. Those heads can move in an arbitrary, *asynchronous* way but only from left to right (i.e. from the beginning to the end of a word).

It was shown by Elgot and Mezei [EM65] that the class of rational relations is closed under union but not under intersection and complement. However, the intersection of a rational relation with a recognizable relation (defined below) is again rational. It is easy to see that **Rat** is also closed under projection.

Proposition 2.3.25. *The class* **Rat** *is closed under union, projection, and intersection with recognizable relations.*

Deterministic rational relations For k-tape automata over the alphabet Σ we define the sets $H_1, \dots, H_k$ by $H_i = \{\varepsilon\}^{i-1} \times \Sigma \times \{\varepsilon\}^{k-i}$. A k-tape automaton $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ is *deterministic* if (i) Q is equipped with a partition into sets $Q = \bigcup_{i=1}^{k} Q_i$, (ii) the transition relation has the form $\Delta \subseteq \bigcup_{i=1}^{k} Q_i \times H_i \times Q$, and (iii) for every $(p, h) \in Q_i \times H_i$ there exists exactly one transition $(p, h, q) \in \Delta$. For convenience, we represent Δ as a transition function $\delta \colon Q \times \Sigma \to Q$ instead.

Intuitively, a state of a deterministic k-tape automaton already determines from which tape the next symbol is read and together with the symbol it determines the succeeding state. Observe that deterministic 1-tape automata are precisely DFAs.

A relation $R \subseteq (\Sigma^*)^k$ is *deterministic rational* if there exists a deterministic k-tape automaton $\mathcal{A}$ such that $R(\mathcal{A}) = \{(w_1\dashv, \dots, w_k\dashv) \mid (w_1, \dots, w_k) \in R\}$ where $\dashv \notin \Sigma$ is a fresh endmarker. We let **DRat** denote the class of all deterministic rational relations. The endmarker is used to allow the deterministic multitape automaton to recognize the end of its tapes. For example, the relation $R = \{(a, \varepsilon), (\varepsilon, b)\}$ would not be deterministic rational without the endmarker since the initial state of any deterministic 2-tape automaton already determines whether the first symbol is read from the first or the second tape, meaning that it cannot accept both (a, ε) and (ε, b). However, if we allow the endmarker, then a deterministic 2-tape automaton for $\{(a\dashv, \dashv), (\dashv, b\dashv)\}$ can easily be constructed.

Deterministic rational relations were introduced by Rabin and Scott [RS59], who showed that the class of deterministic rational relations is closed under complement but not under union and intersection.

Regular relations Let $\perp \notin \Sigma$ be a fresh padding symbol and $\Sigma_\perp := \Sigma \cup \{\perp\}$. For words $w_1, \dots, w_k \in \Sigma^*$ with $w_i = a_{i,1} \dots a_{i,n_i}$ we define the *convolution* $w_1 \otimes \cdots \otimes w_k$ of length $n := \max\{n_1, \dots, n_k\}$ by

$$w_1 \otimes \cdots \otimes w_k := \begin{bmatrix} w_1 \\ \vdots \\ w_k \end{bmatrix} := \begin{pmatrix} a'_{1,1} \\ \vdots \\ a'_{k,1} \end{pmatrix} \dots \begin{pmatrix} a'_{1,n} \\ \vdots \\ a'_{k,n} \end{pmatrix} \in \left((\Sigma_\perp)^k\right)^*$$

where $a'_{i,j} = a_{i,j}$ if $j \leq n_i$ and $a'_{i,j} = \perp$ otherwise. A relation $R \subseteq (\Sigma^*)^k$ over words is *regular* (a.k.a. synchronous, automatic) if $\otimes R := \{w_1 \otimes \cdots \otimes w_k \mid (w_1, \dots, w_k) \in R\}$ is a regular language. A regular relation R is always given by a finite automaton for the language $\otimes R$. The class of regular relations is denoted by **Reg**. Clearly, the regular relations of arity 1 coincide with the regular languages.

Example 2.3.26. Note that convolution is not associative. For example, we have

$$(a \otimes bb) \otimes ccc = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \begin{pmatrix} \perp \\ b \\ c \end{pmatrix} \begin{pmatrix} \perp \\ c \end{pmatrix} \neq \begin{pmatrix} a \\ b \\ c \end{pmatrix} \begin{pmatrix} \perp \\ b \\ c \end{pmatrix} \begin{pmatrix} \perp \\ \perp \\ c \end{pmatrix} = a \otimes bb \otimes ccc$$

where $(a \otimes bb) \otimes ccc$ is a word over the alphabet $((\Sigma_\perp)^2 \cup \Sigma_\perp)^2$ rather than $(\Sigma_\perp)^3$.

In terms of multitape automata, a regular relation can be described by a multitape automaton that in each step reads exactly one symbol from every tape or ε, but then it cannot read a symbol from this tape again. This means that the heads move *synchronously* until the end of some tape is reached at which point the other heads are allowed to make further steps. Such a multitape automaton can easily be made deterministic, so that the regular relations form a subclass of the deterministic rational relations. Note that without padding symbol (or allowing other heads to continue reading) regular relations could only relate words of the same length.

By definition, the class of regular relations inherits closure under Boolean operations from regular languages (Proposition 2.3.4). Recall that for a k-ary relation R and a set $I \subseteq [1,k]$ we write $\pi_I(R)$ for the projection of R to the components in I. The following proposition shows that the class of regular relations is closed under projection. Note, however, that even if we start with a deterministic automaton, the automaton for the projection may be nondeterministic.

Proposition 2.3.27. *Given an NFA $\mathcal{A}$ for a regular relation $R \subseteq (\Sigma^*)^k$ with $k \geq 1$ and a set $I \subseteq [1,k]$, one can compute in logspace an NFA $\mathcal{A}'$ for the relation $\pi_I(R)$.*

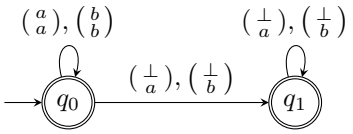
Proof. Let $I = \{i_1, \ldots, i_\ell\}$ with $i_1 < \cdots < i_\ell$. We can construct $\mathcal{A}'$ from $\mathcal{A}$ by replacing every transition $(p, (a_1, \ldots, a_k), q)$ with $(p, (a_{i_1}, \ldots, a_{i_\ell}), q)$ if $a_{i_j} \neq \perp$ for some $j \in [1,\ell]$ and with (p, ε, q) otherwise. □

Recognizable relations A k-ary relation R on words is *recognizable* if it is a finite union $R = \bigcup_{i=1}^n L_{i,1} \times \cdots \times L_{i,k}$ of Cartesian products of regular languages $L_{i,j}$. We denote the class of recognizable relations by **Rec**. Note that recognizability is originally an algebraic notion that for relations on words coincides with the above definition. We refer to [Sak09] for more details on the algebraic point of view.

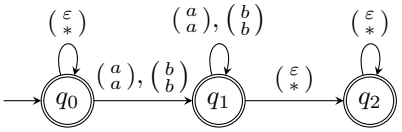
Equivalently, recognizable relations are those that are accepted by independent multitape automata. An *independent k-tape automaton* is a tuple $\mathcal{I} = (\mathcal{A}_1, \ldots, \mathcal{A}_k, F)$ consisting of DFAs $\mathcal{A}_i$ without final states and a set of state tuples $F \subseteq Q_1 \times \cdots \times Q_k$ where Q_i is the state set of $\mathcal{A}_i$. The relation $R(\mathcal{I})$ accepted by $\mathcal{I}$ is the set of all word-tuples $(w_1, \ldots, w_k)$ such that for each $i \in [1,k]$ the unique run of $\mathcal{A}_i$ on w_i ends in a state $q_i \in Q_i$ and it holds that $(q_1, \ldots, q_k) \in F$. Intuitively, an independent multitape automaton consists of a separate automaton for each tape and they can only communicate via the states reached at the end of the runs. Clearly, an independent multitape automaton can be simulated by a single DFA with padding using a product construction (although the representation as independent multitape automaton is more succinct). Thus, recognizable relations are special regular relations.

Using standard set theoretic identities and the closure properties of regular languages (Proposition 2.3.4), it is straightforward to show that the class of recognizable relations is closed under Boolean operations and projection.

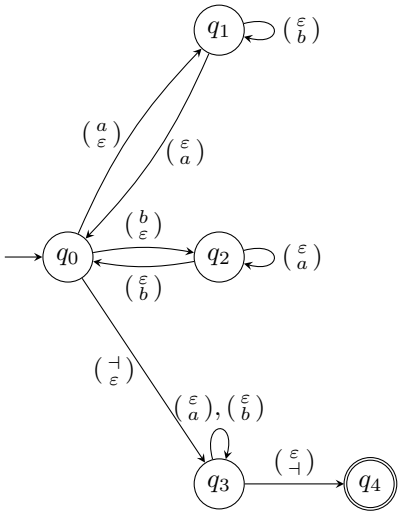
Proposition 2.3.28. *The class* **Rec** *is closed under union, intersection, complement, and projection.*



(a) NFA for the regular relation $R_2 := \{(x, y) \mid x \text{ is a prefix of } y\}$.



(c) 2-tape automaton for the rational relation $R_4 := \{(x, y) \mid x \text{ is an infix of } y\}$, where $*$ is any symbol from Σ.



(b) Deterministic 2-tape automaton for the deterministic rational relation $R_3 := \{(x, y) \mid x \text{ is a subword of } y\}$, where the error state is omitted.

Figure 2.2: Examples of regular, deterministic rational, and rational relations over the alphabet $\Sigma = \{a, b\}$.

By definition or as argued above, the introduced classes form a hierarchy

$$\textbf{Rec} \subsetneq \textbf{Reg} \subsetneq \textbf{DRat} \subsetneq \textbf{Rat}$$

where strictness of the inclusions is illustrated by the following examples.

Example 2.3.29. The relation $R_1 := \{(x, y) \mid |x| + |y| \geq 2\}$ over any alphabet Σ is recognizable since it can be written as the union of the Cartesian products $\Sigma^{\geq 2} \times \Sigma^*$, $\Sigma^{\geq 1} \times \Sigma^{\geq 1}$, and $\Sigma^* \times \Sigma^{\geq 2}$, where $\Sigma^{\geq \ell}$ contains all words of length at least ℓ. The prefix relation $R_2 := \{(x, y) \mid x \text{ is a prefix of } y\}$ is regular as shown in Figure 2.2a, but clearly not recognizable. The relation $R_3 := \{(x, y) \mid x \text{ is a subword of } y\}$ is deterministic rational, where the deterministic automaton depicted in Figure 2.2b greedily embeds x into y, but not regular. The relation $R_4 := \{(x, y) \mid x \text{ is an infix of } y\}$ is rational as shown in Figure 2.2c but not deterministic rational.

2.3.6 Relations over Infinite Words

A similar hierarchy of classes of relations can be defined over infinite words. Here, we extend the models over finite words to infinite words by utilizing the Büchi or parity acceptance condition.

ω-rational relations We extend the notion of a run of a k-tape automaton $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ on a tuple of infinite words $\boldsymbol{w} \in (\Sigma^\omega)^k$ as an infinite sequence of transitions

$$(p_0, \boldsymbol{w}_1, p_1)(p_1, \boldsymbol{w}_2, p_2) \ldots \in \Delta^\omega$$

such that $\boldsymbol{w} = \boldsymbol{w}_1 \boldsymbol{w}_2 \ldots$. We use the Büchi acceptance condition, i.e. the above run is accepting if $p_0 = q_0$ and $p_i \in F$ for infinitely many $i \geq 0$. The automaton model with the above semantics is called *k-tape Büchi automaton*. Relations on infinite words accepted by k-tape Büchi automata are called *ω-rational*. We denote the class of ω-rational relations by ω-**Rat**. Note that the ω-rational relations of arity 1 are exactly the ω-regular languages, i.e. those accepted by NBAs.

Deterministic ω-rational relations As already mentioned in Section 2.3.2, deterministic Büchi automata are strictly less expressive than their nondeterministic version. For this reason, we use the parity acceptance condition to define the notion of deterministic ω-rational relations. A *deterministic k-tape parity automaton* is a tuple $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F, c)$ that extends a deterministic k-tape automaton with a coloring function $c\colon Q \to \{1, \ldots, \ell\}$ for $\ell \in \mathbb{N}$. Then a run of $\mathcal{A}$ on a tuple of infinite words is accepting if it starts with q_0 and the minimal color of states occurring infinitely often in the run is even. We call the relations accepted by deterministic k-tape parity automata *deterministic ω-rational* and denote the class of deterministic ω-rational relations by ω-**DRat**. Again, note that deterministic ω-rational relations of arity 1 coincide with the languages accepted by DPAs.

ω-regular relations The *convolution* on infinite words $w_1, \ldots, w_k \in \Sigma^\omega$ with $w_i = a_{i,1} a_{i,2} \ldots$ is defined similarly as in the finite-word case as

$$w_1 \otimes \cdots \otimes w_k := \begin{bmatrix} w_1 \\ \vdots \\ w_k \end{bmatrix} := \begin{pmatrix} a_{1,1} \\ \vdots \\ a_{k,1} \end{pmatrix} \begin{pmatrix} a_{1,2} \\ \vdots \\ a_{k,2} \end{pmatrix} \ldots \in (\Sigma^k)^\omega$$

where we do not need a padding symbol. A relation $R \subseteq (\Sigma^\omega)^k$ on infinite words is *ω-regular* if $\otimes R := \{w_1 \otimes \cdots \otimes w_k \mid (w_1, \ldots, w_k) \in R\}$ is an ω-regular language. We always assume that an ω-regular relation R is given by an automaton for $\otimes R$. The class of ω-regular relations is denoted by ω-**Reg**.

The class of ω-regular relations enjoys by definition the same closure properties as ω-regular languages (Proposition 2.3.10). Moreover, with the same construction as in Proposition 2.3.27, we can show that ω-regular relations are also closed under projection.

Proposition 2.3.30. *Given an NBA $\mathcal{A}$ for an ω-regular relation $R \subseteq (\Sigma^\omega)^k$ with $k \geq 1$ and a set $I \subseteq [1, k]$, one can compute in logspace an NBA $\mathcal{A}'$ for the relation $\pi_I(R)$.*

In Lemma 2.3.11 we saw that every non-empty ω-regular language contains an ultimately periodic word. This observation can easily be extended to ω-regular relations: Let $R \subseteq (\Sigma^\omega)^k$ be a non-empty ω-regular relation. By Lemma 2.3.11, there exists an

ultimately periodic word $uv^\omega \in \otimes R$ with $u = u_1 \otimes \cdots \otimes u_k$ and $v = v_1 \otimes \cdots \otimes v_k$ such that $|u_1| = \cdots = |u_k|$ and $|v_1| = \cdots = |v_k|$. This ultimately periodic word corresponds to a tuple of ultimately periodic words $(u_1 v_1^\omega, \ldots, u_k v_k^\omega) \in R$. Using the techniques from [LS19b], we can show a similar statement for the equivalence classes of ω-regular equivalence relations.

Lemma 2.3.31. *Let $E \subseteq (\Sigma^\omega)^k \times (\Sigma^\omega)^k$ be an ω-regular equivalence relation. If E has finite index, then every equivalence class of E contains a tuple of ultimately periodic words. If E has infinite index, then there exist infinitely many equivalence classes, each of which contains a tuple of ultimately periodic words.*

Proof. By the above observation, we can pick a tuple $\boldsymbol{u}_1$ of ultimately periodic words from $(\Sigma^\omega)^k$. By the closure properties of ω-regular relations, $(\Sigma^\omega)^k \setminus [\boldsymbol{u}_1]$ is again ω-regular, where $[\boldsymbol{u}_1]$ denotes the E-equivalence class of $\boldsymbol{u}_1$. We repeat this process for $(\Sigma^\omega)^k \setminus [\boldsymbol{u}_1]$ as long as the set is non-empty. If E has finite index, then we will eventually pick tuples of ultimately periodic words for all finitely many equivalence classes. If E has infinite index, we obtain an infinite set of tuples of ultimately periodic words from pairwise distinct equivalence classes in the limit. □

ω-recognizable relations The smallest class of relations over infinite words that we consider is the class of ω-recognizable relations, denoted by ω-**Rec**. A k-ary relation R on infinite words is *ω-recognizable* if it is a finite union $R = \bigcup_{i=1}^{n} L_{i,1} \times \cdots \times L_{i,k}$ of direct products of ω-regular languages $L_{i,j}$. Using Proposition 2.3.10, it is easy to show:

Proposition 2.3.32. *The class ω-**Rec** is closed under union, intersection, complement, and projection.*

As in the finite-word case, the introduced classes of relations on infinite words form a strict hierarchy.

$$\omega\text{-}\mathbf{Rec} \subsetneq \omega\text{-}\mathbf{Reg} \subsetneq \omega\text{-}\mathbf{DRat} \subsetneq \omega\text{-}\mathbf{Rat}$$

Examples of relations for each of the four classes can be obtained from Example 2.3.29 by adding an infinite padding to words.

2.3.7 Relations over Finite Trees

Rational relations can also be defined over trees (see [Rao97; Rad08]), but in this thesis we only need the subclasses consisting of regular and recognizable relations over trees.

Tree-regular relations The intuition for tree-regular relations is the same as over words: A tree automaton reads multiple trees *synchronously*. For the formal definition we need again the notion of convolution. For an alphabet Σ we set again $\Sigma_\perp := \Sigma \cup \{\perp\}$ where $\perp \notin \Sigma$ is a fresh padding symbol. Let ε be the *empty tree* with $\mathrm{dom}(\varepsilon) := \emptyset$. Given k

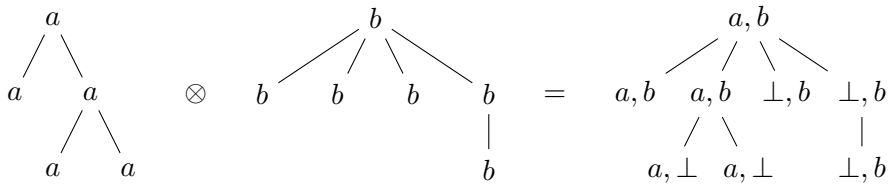


Figure 2.3: An example illustrating the convolution of two trees.

trees $t_1, \dots, t_k \in \mathcal{U}_\Sigma \cup \{\varepsilon\}$, we define their convolution

$$t_1 \otimes \cdots \otimes t_k := \begin{bmatrix} t_1 \\ \vdots \\ t_k \end{bmatrix} := t \in \mathcal{U}_{(\Sigma_\perp)^k} \cup \{\varepsilon\}$$

where $\mathrm{dom}(t) := \bigcup_{i=1}^k \mathrm{dom}(t_i)$ and $t(v) := (t'_1(v), \dots, t'_k(v))$ with $t'_i(v) := t_i(v)$ if $v \in \mathrm{dom}(t_i)$ and $t'_i(v) := \perp$ otherwise. We refer to Figure 2.3 for an example. Observe that the degree of a node v in $t_1 \otimes \cdots \otimes t_k$ is the maximum degree of v in a tree t_i such that $v \in \mathrm{dom}(t_i)$. If all t_i are ranked trees, then also $t_1 \otimes \cdots \otimes t_k$ is a ranked tree with $\mathrm{rk}(a_1, \dots, a_k) := \max\{\mathrm{rk}(a_i) \mid 1 \le i \le k\}$ for all $(a_1, \dots, a_k) \in (\Sigma_\perp)^k$ where $\mathrm{rk}(\perp) := 0$.

A relation $R \subseteq (\mathcal{T}_\Sigma)^k$ is *tree-regular* if the tree language $\otimes R := \{t_1 \otimes \cdots \otimes t_k \mid (t_1, \dots, t_k) \in R\}$ is tree-regular. In particular, tree-regular relations of arity 1 coincide with the tree-regular languages. We always assume that tree-regular relations R are given by a tree automaton for $\otimes R$.

As over words, the class of tree-regular relations is closed under projection. However, for the proof we cannot just project the transitions of the automaton to the respective components. We additionally need to check non-emptiness starting from states that are projected away.

Proposition 2.3.33. *Given an NTA (resp. ATA) $\mathcal{A}$ for a tree-regular relation $R \subseteq (\mathcal{T}_\Sigma)^k$ with $k \ge 1$ and a set $I \subseteq [1, k]$, one can compute in polynomial (resp. exponential) time an NTA $\mathcal{A}'$ for the relation $\pi_I(R)$.*

Proof. Let $I = \{i_1, \dots, i_\ell\}$ for $i_1 < \cdots < i_\ell$. We start with construction of an NTA $\mathcal{A}'$ for the projection $\pi_I(R)$ in case that $\mathcal{A}$ is an NTA. For every transition $(q, (a_1, \dots, a_k), q_1, \dots, q_r)$ of $\mathcal{A}$ and $r' := \mathrm{rk}(a_{i_1}, \dots, a_{i_\ell})$ we do the following: We check for all $r' < i \le r$ whether the tree language accepted by $\mathcal{A}$ from state q_i is non-empty. By Proposition 2.3.17, these non-emptiness checks can be performed in polynomial time. If all non-emptiness checks are successful, we add the transition $(q, (a_{i_1}, \dots, a_{i_\ell}), q_1, \dots, q_{r'})$ to $\mathcal{A}'$. Otherwise, the transition is discarded.

If $\mathcal{A}$ is an ATA, we first use Proposition 2.3.20 followed by Proposition 2.3.13 to transform $\mathcal{A}$ into an NTA in exponential time and then apply the above construction. □

Tree-recognizable relations We define *tree-recognizable relations* analogously to the word case as the class of relations that can be written as a finite union $\bigcup_{i=1}^n L_{i,1} \times \cdots \times L_{i,k}$

of Cartesian products of tree-regular languages $L_{i,j}$. Again, tree-recognizable relations form a strict subclass of tree-regular relations. Closure of tree-recognizable relations under Boolean operations and projection follows by the same reasoning as in the word case.

Proposition 2.3.34. *The class of tree-recognizable relations is closed under union, intersection, complement, and projection.*

2.4 Logic

We now introduce notions from mathematical logic that we require in this thesis. For a deeper introduction to the topic we refer to [End01; Sch08; Ben12].

2.4.1 First-Order Logic

Since we mainly deal with first-order logic in this thesis, we only formally define first-order logic in the following. Note that *propositional logic* can be seen as a special case, not allowing quantification, where the domain is $\{0, 1\}$ and atoms are of the form $x = 1$, which is just written as x.

Syntax We assume an infinite set of first-order *variables*. A *signature* is a tuple of relation and function symbols associated with arities. We call function symbols of arity 0 *constants*. A *term* over a signature σ is either a first-order variable or a function application $f(t_1, \dots, t_n)$ where f is an n-ary function symbol and $t_1, \dots, t_n$ are terms. A term that does not contain any variables, i.e. only consisting of function applications and constants, is called *ground term*. An *atom* over σ is either a relation constraint $R(t_1, \dots, t_n)$ for an n-ary relation symbol R and terms $t_1, \dots, t_n$ or an equality constraint $t_1 = t_2$ for terms t_1, t_2. A (first-order) *formula* over σ is either an atom or if φ_1, φ_2 are formulas, then also the negation $\neg\varphi_1$, conjunction $\varphi_1 \wedge \varphi_2$, disjunction $\varphi_1 \vee \varphi_2$, and existential quantification $\exists x\colon \varphi_1$ for a variable x are formulas. As a shorthand, we will write implication $\varphi_1 \to \varphi_2$ for $(\neg\varphi_1) \vee \varphi_2$, equivalence $\varphi_1 \leftrightarrow \varphi_2$ for $(\varphi_1 \to \varphi_2) \wedge (\varphi_2 \to \varphi_1)$, and universal quantification $\forall x\colon \varphi_1$ for $\neg\exists x\colon \neg\varphi_1$. To reduce the usage of parentheses, we assume the binding strengths of logical operators to be $\neg, \wedge, \vee, \to, \leftrightarrow$ in decreasing order and quantifiers bind the weakest. Occurrences of variables in the scope of a quantifier that quantifies the variable are called *bound*. All other variable occurrences are called *free*. We say that a formula in which every variable occurrence is bound is *closed* or a *sentence*. If all occurrences of a variable in a formula are bound (resp. free), we also say that the variable is bound (resp. free). We will often denote a tuple $(x_1, \dots, x_n)$ of variables by a boldface letter $\boldsymbol{x}$. If φ is a formula, we write $\varphi(\boldsymbol{x})$ to indicate that $\boldsymbol{x}$ are the variables with free occurrences in φ. For the sake of brevity, we compress a block of quantifiers $Qx_1\colon \dots Qx_n\colon \varphi$ with $Q \in \{\exists, \forall\}$ by $Q\boldsymbol{x}\colon \varphi$.

Semantics For a given signature σ we define a (σ-)*structure* as a pair $\mathfrak{A} = (A, I)$ consisting of a non-empty set A, called *domain*, and an *interpretation function* I mapping

relation symbols R of arity n to a relation $I(R) \subseteq A^n$ and function symbols f of arity n to a function $I(f)\colon A^n \to A$. We often assume that a structure implicitly contains its signature and will therefore just say that a formula is defined over a structure (instead of the signature of the structure). If the interpretation of the relation symbols R_i and function symbols f_j is clear, we also write a structure as $\langle A; (R_i)_{i\in N}, (f_j)_{j\in M}\rangle$ where $N, M \subseteq \mathbb{N}$. We may also just define relations R_i and functions f_j over some domain A and implicitly assume that formulas are over the structure $\langle A; (R_i)_{i\in N}, (f_j)_{j\in M}\rangle$, where the identifiers R_i (resp. f_j) are overloaded and stand for both the relation (resp. function) symbol and its interpretation. Given a structure $\mathfrak{A} = (A, I)$ and a *valuation* ν mapping variables to values in A, we define $\mathrm{val}(x) := \nu(x)$ for a variable x and $\mathrm{val}(f(t_1, \dots, t_n)) := I(f)(\mathrm{val}(t_1), \dots, \mathrm{val}(t_n))$ for an n-ary function symbol f and terms $t_1, \dots, t_n$. Then we say that $\mathfrak{A}$ and ν *satisfy* a formula φ, written $\mathfrak{A}, \nu \models \varphi$, if

1) $\varphi = R(t_1, \dots, t_n)$ for an n-ary relation symbol R and terms $t_1, \dots, t_n$ such that $(\mathrm{val}(t_1), \dots, \mathrm{val}(t_n)) \in I(R)$,

2) $\varphi = (t_1 = t_2)$ for terms t_1, t_2 such that $\mathrm{val}(t_1) = \mathrm{val}(t_2)$,

3) $\varphi = \neg\psi$ for a formula ψ such that $\mathfrak{A}, \nu \not\models \psi$,

4) $\varphi = \psi_1 \wedge \psi_2$ for formulas ψ_1, ψ_2 such that $\mathfrak{A}, \nu \models \psi_1$ and $\mathfrak{A}, \nu \models \psi_2$,

5) $\varphi = \psi_1 \vee \psi_2$ for formulas ψ_1, ψ_2 such that $\mathfrak{A}, \nu \models \psi_1$ or $\mathfrak{A}, \nu \models \psi_2$, or

6) $\varphi = \exists x\colon \psi$ for a formula ψ such that there exists $a \in A$ such that $\mathfrak{A}, \nu' \models \psi$ where $\nu'(x) := a$ and $\nu'(y) := \nu(y)$ for every variable $y \neq x$.

We say that two formulas φ_1 and φ_2 over a signature σ are *equivalent*, written $\varphi_1 \equiv \varphi_2$, if $\mathfrak{A}, \nu \models \varphi_1$ if and only if $\mathfrak{A}, \nu \models \varphi_2$ for every σ-structure $\mathfrak{A}$ and valuation ν. If the structure $\mathfrak{A}$ is fixed, we also write $\varphi_1 \equiv_{\mathfrak{A}} \varphi_2$ (or just $\varphi_1 \equiv \varphi_2$ if $\mathfrak{A}$ is clear from the context) in case that $\mathfrak{A}, \nu \models \varphi_1$ if and only if $\mathfrak{A}, \nu \models \varphi_2$ for every valuation ν.

Clearly, the satisfaction of a formula only depends on the valuation of the variables that actually occur in the formula. In other words, if φ is a formula over the signature of the structure $\mathfrak{A}$ and ν_1 and ν_2 are valuations with $\nu_1(x) = \nu_2(x)$ for all variables x in φ, then $\mathfrak{A}, \nu_1 \models \varphi$ if and only if $\mathfrak{A}, \nu_2 \models \varphi$. Thus, if $\varphi(\boldsymbol{x})$ is a formula with free occurrences of variables $\boldsymbol{x} = (x_1, \dots, x_n)$, we can write $\mathfrak{A} \models \varphi(\boldsymbol{a})$ if $\mathfrak{A}, \nu \models \varphi$ for some valuation ν with $\nu(x_i) = a_i$ for all $i \in [1, n]$. In particular, this means that if φ is a sentence, we simply write $\mathfrak{A} \models \varphi$ if $\mathfrak{A}$ satisfies φ. Furthermore, for a formula $\varphi(\boldsymbol{x})$ over the signature of a structure $\mathfrak{A} = (A, I)$ we let $[\![\varphi]\!]_{\mathfrak{A}}$ (or just $[\![\varphi]\!]$) denote the relation $\{\boldsymbol{a} \in A^{|\boldsymbol{x}|} \mid \mathfrak{A} \models \varphi(\boldsymbol{a})\}$. Here, we also allow to fix the valuation for some of the free variables, i.e. for $\varphi(\boldsymbol{x}, \boldsymbol{y})$ and $\boldsymbol{a} \in A^{|\boldsymbol{x}|}$ we can write $[\![\varphi(\boldsymbol{a}, \boldsymbol{y})]\!]_{\mathfrak{A}} := \{\boldsymbol{b} \in A^{|\boldsymbol{y}|} \mid \mathfrak{A} \models \varphi(\boldsymbol{a}, \boldsymbol{b})\}$.

The *theory* of a structure $\mathfrak{A}$, denoted by $\mathrm{Th}(\mathfrak{A})$, is the set of all sentences φ such that $\mathfrak{A} \models \varphi$. Note that for a fixed structure $\mathfrak{A}$, deciding whether a sentence over $\mathfrak{A}$ is contained in $\mathrm{Th}(\mathfrak{A})$ is equivalent to the problem of checking satisfiability of a (not necessarily closed) formula with respect to $\mathfrak{A}$. Below we will give some examples of first-order theories.

Normal forms A formula in *negation normal form* (NNF) is a positive Boolean combination of possibly negated atoms. A formula is in *disjunctive normal form* (DNF) if it is a disjunction of conjunctions of possibly negated atoms. Similarly, a formula in *conjunctive normal form* (CNF) is a conjunction of disjunctions. Note that every quantifier-free formula can be converted into a formula in DNF/CNF (with an exponential blow-up in size). Using standard equivalences and renaming of variables, every quantified formula can be converted into *prenex normal form* (PNF), i.e. having the quantifiers only at the front.

2.4.2 Presburger Arithmetic

Linear integer arithmetic (LIA) is defined as the theory of the structure $\langle \mathbb{Z}; <, +, 0, 1 \rangle$. LIA is also called *Presburger arithmetic* and we will use these terms interchangeably also to identify the structure instead of the theory. For computational purposes it is often convenient to allow terms of the form

$$a_0 + a_1 x_1 + \cdots + a_n x_n$$

where $x_1, \ldots, x_n$ are variables and $a_0, \ldots, a_n \in \mathbb{Z}$ are integer constants. That is, formally we assume that the signature of Presburger arithmetic contains all integer constants and we allow multiplication of variables with constants. Thus, a formula in Presburger arithmetic is a quantified Boolean combination of linear (in)equalities with integer coefficients. Note that any such atom can be transformed into an atom over the original structure. We define the *size* of a formula by the length of its usual encoding, where we assume that every variable occurrence has length one and coefficients are encoded in binary.

In the *existential fragment* of Presburger arithmetic we assume formulas of the form $\exists \boldsymbol{x} \colon \varphi(\boldsymbol{x}, \boldsymbol{z})$, where the variables in $\boldsymbol{x}$ are bound by the existential quantifiers and $\boldsymbol{z}$ is a vector of free variables. It follows from [BT76; GS78] that the problem of checking whether a given formula in the existential fragment of Presburger arithmetic is satisfiable is in NP.

Proposition 2.4.1. *Satisfiability of existential Presburger formulas is* NP*-complete.*

Here, NP-hardness follows from a reduction from SAT (satisfiability of propositional formulas) since a propositional formula can easily be converted to a Presburger formula by replacing each occurrence of a propositional variable x with the atom $x = 1$ and adding the constraints that every variable is either equal to 0 or 1.

We say that a structure $\mathfrak{A}$ *admits quantifier elimination* if for every formula φ over $\mathfrak{A}$ there exits a quantifier-free formula φ' over $\mathfrak{A}$ such that $\varphi \equiv_{\mathfrak{A}} \varphi'$. Presburger arithmetic with the above signature does not admit quantifier elimination. To get this property, one has to enrich the signature with modulo constraints. A modulo constraint is a binary predicate $\equiv_e$ with $e \in \mathbb{N}$ such that for Presburger terms s, t we have that $s \equiv_e t$ is satisfied if and only if $e \mid s - t$. Note that modulo constraints are definable over $\langle \mathbb{Z}; <, +, 0, 1 \rangle$ using existential quantifiers, which means that the structure $\langle \mathbb{Z}; <, (\equiv_e)_{e \in \mathbb{N}}, +, 0, 1 \rangle$ still defines Presburger arithmetic. In fact, the existential fragments even coincide since also

a negated modulo constraint $s \not\equiv_e t$ can be written equivalently as existential formula $\exists x, y\colon s = t + ex + y \wedge 0 < y \wedge y < e$ over $\langle \mathbb{Z}; <, +, 0, 1\rangle$ where x, y are fresh variables. It was shown by Presburger [Pre29] that Presburger arithmetic with the enriched signature admits quantifier elimination.

Proposition 2.4.2. *The structure $\langle \mathbb{Z}; <, (\equiv_e)_{e\in\mathbb{N}}, +, 0, 1\rangle$ admits quantifier elimination.*

Proof. First observe that it suffices to show how to eliminate a single existential quantifier from a formula of the form $\exists x\colon \varphi$, where φ is quantifier-free, since we can bring the formula into prenex normal form, eliminate the quantifiers from the innermost to the outermost, and replace universal quantifiers $\forall x\colon \psi$ with negated existential quantifiers $\neg\exists x\colon \neg\psi$. Moreover, by bringing φ into disjunctive normal form, where we move the negations into the atoms (and replace negated modulo constraints with disjunctions of modulo constraints), and distributing the existential quantifier over the disjunctions, we can assume that φ is a conjunction of atoms.

We first ensure that every atom in φ has the form $ax \sim t$ for $a \in \mathbb{N}$ and a term t that does not involve x, where $\sim \in \{<, >, =\} \cup \{\equiv_e |\ e \in \mathbb{N}\}$. Let $a_1, \dots, a_n \in \mathbb{N}$ be the coefficients of x in atoms of φ and $N := \operatorname{lcm}\{a_1, \dots, a_n\}$. Now, replace every (in)equality of the form $a_i x \sim t$ for $\sim \in \{<, >, =\}$ with $x \sim \frac{N}{a_i} \cdot t$ and every modulo constraint of the form $a_i x \equiv_e t$ with $x \equiv_{\frac{N}{a_i}\cdot e} \frac{N}{a_i} \cdot t$. Additionally, we add the conjunct $x \equiv_N 0$ to φ. Thus, every atom of φ is now in solved form for x.

If φ contains an equation of the form $x = t$, then we can replace every occurrence of x in φ with t, which eliminates x from φ. Therefore, we can assume that φ does not contain any equality involving x. Assume φ contains modulo constraints $x \equiv_{e_i} t_i$ for $i \in [1, n]$ and inequalities $x > t_j$ for $j \in [1, m]$. Let $N := \operatorname{lcm}\{e_1, \dots, e_n\}$. By the Chinese remainder theorem, if there is a solution for x that satisfies all modulo constraints, then there is an $r \in [0, N-1]$ such that $r + k \cdot N$ is a solution for every $k \in \mathbb{Z}$. Hence, we have that

$$\exists x\colon \varphi \equiv \bigvee_{j=1}^{m} \bigvee_{r=1}^{N} \varphi[t_j + r/x]$$

where $\varphi[t_j + r/x]$ denotes the formula φ where every occurrence of x is replaced with $t_j + r$. If only modulo constraints and upper bounds on x are present in φ, the elimination works analogously. If φ only contains modulo constraints on x, then x can be eliminated by $\bigvee_{r=0}^{N-1} \varphi[r/x]$. It remains to consider the case where all atoms of φ involving x are inequalities and there is at least one lower and one upper bounds on x. Note that all other cases can be handled by simply replacing all atoms involving x with true. Let $x > \ell_i$ for $i \in [1, n]$ be the lower bounds and $x < u_j$ for $j \in [1, m]$ be the upper bounds on x contained in φ. Then x can be eliminated by removing all inequalities involving x and adding the conjuncts

$$\bigwedge_{i=1}^{n} \bigwedge_{j=1}^{m} \ell_i + 1 < u_j.$$

□

Note that the above proof even shows that there is an effective quantifier elimination procedure. For complexity considerations we refer to [Wei97]. Here it should be noted that the inherent double exponential blow-up when eliminating a single block of existential quantifiers claimed in [Wei97] is not correct as shown independently in [Haa+24] and [CMS24], where a single exponential-time elimination procedure is provided.

2.4.3 Linear Real Arithmetic

We define *linear real arithmetic* (LRA) as the theory of the structure $\langle \mathbb{R}; <, +, 0, 1 \rangle$. Again, we also use LRA to refer to the structure. As for Presburger arithmetic, we allow multiplication with constants. Here, constants range over the rational numbers. That is, terms in LRA are of the form

$$a_0 + a_1 x_1 + \cdots + a_n x_n$$

for variables $x_1, \ldots, x_n$ and constants $a_0, \ldots, a_n \in \mathbb{Q}$. Note that any atom over such terms can be transformed into an atom over the original structure $\langle \mathbb{R}; <, +, 0, 1 \rangle$.

As over the integers, the existential fragment of LRA is NP-complete [Son85].

Proposition 2.4.3. *Satisfiability of existential formulas in LRA is* NP*-complete.*

Moreover, LRA with the signature $\langle \mathbb{R}; <, +, 0, 1 \rangle$ already admits quantifier elimination. This goes back to Fourier [Fou26] and was rediscovered several times thereafter most famously by Motzkin [Mot36]. For this reason, the method is often called *Fourier-Motzkin elimination.*

Proposition 2.4.4. *The structure* $\langle \mathbb{R}; <, +, 0, 1 \rangle$ *admits quantifier elimination.*

Proof. The elimination is simpler than in the case of Presburger arithmetic (Proposition 2.4.2). Again, it suffices to consider formulas of the form $\exists x \colon \varphi$ where φ is a conjuction of atoms. Moreover, we can assume that every atom of φ involving x is already solved for x, i.e. is of the form $x \sim t$ for a term t not involving x and $\sim \in \{<, >, =\}$.

If φ contains an equation $x = t$, then $\exists x \colon \varphi$ is equivalent to $\varphi[t/x]$. If φ either contains only lower bounds or only upper bounds on x, we can replace every inequality involving x with true. Therefore, we can assume that φ contains at least one lower and one upper bound on x. Let $x > \ell_i$ for $i \in [1, n]$ be the lower bounds and $x < u_j$ for $j \in [1, m]$ be the upper bounds on x contained in φ. Now, x can be eliminated by removing all inequalities involving x and adding the conjuncts $\bigwedge_{i=1}^{n} \bigwedge_{j=1}^{m} \ell_i < u_j$. □

2.4.4 Linear Integer Real Arithmetic

We call the theory of the structure $\langle \mathbb{R}; <, \lfloor . \rfloor, +, 0, 1 \rangle$ *linear integer real arithmetic* (LIRA). Here, $\lfloor r \rfloor$ returns the greatest integer smaller than or equal to $r \in \mathbb{R}$. LIRA can be seen as the mixture of LIA and LRA and in terms of full first-order logic it is equally expressive as the structure $\langle \mathbb{R}; \mathbb{Z}, <, +, 0, 1 \rangle$ since we can check whether a variable x is an integer using $x = \lfloor x \rfloor$ and conversely, $\lfloor x \rfloor$ can be replaced with y where y, z are fresh

existentially quantified variables with $y \in \mathbb{Z}$, $0 \leq z < 1$, and $x = y + z$. Note that for the latter direction we introduced new existential quantifiers. In fact, Weispfenning [Wei99] showed that even if extended with modulo constraints (where $r \equiv_e s$ for $r, s \in \mathbb{R}$ if and only if $e \mid r - s$), the structure with integer test does not admit quantifier elimination. Whereas, if we extend $\langle \mathbb{R}; <, \lfloor . \rfloor, +, 0, 1 \rangle$ with modulo constraints, the structure admits quantifier elimination. In particular, this also implies that the existential fragment of LIRA is expressively complete, i.e. for every formula in LIRA there is an equivalent existential formula.

Using $x = \lfloor x \rfloor$, we can extend LIRA to allow two sorts of variables: real and integer variables. For a vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ of variables let $\boldsymbol{x}^{\mathrm{i/r}}$ denote the vector $(\boldsymbol{x}^{\mathrm{int}}, \boldsymbol{x}^{\mathrm{real}})$ where $\boldsymbol{x}^{\mathrm{int}} = (x_1^{\mathrm{int}}, \ldots, x_n^{\mathrm{int}})$ is a vector of integer variables and $\boldsymbol{x}^{\mathrm{real}} = (x_1^{\mathrm{real}}, \ldots, x_n^{\mathrm{real}})$ is a vector of real variables. Two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ of dimension n are said to have the same *type* if for all $i \in [1, n]$ we have that x_i and y_i are both real or integer variables. The *separation* of an existential formula $\exists x_1, \ldots, x_n \colon \varphi(x_1, \ldots, x_n, z_1, \ldots, z_m)$ in LIRA is defined as

$$\exists \boldsymbol{x}^{\mathrm{i/r}} \colon \varphi(\boldsymbol{x}^{\mathrm{int}} + \boldsymbol{x}^{\mathrm{real}}, \boldsymbol{z}^{\mathrm{int}} + \boldsymbol{z}^{\mathrm{real}}) \wedge \mathbf{0} \leq \boldsymbol{x}^{\mathrm{real}} \ll \mathbf{1} \wedge \mathbf{0} \leq \boldsymbol{z}^{\mathrm{real}} \ll \mathbf{1}$$

where x_i^{int}, z_j^{int} are fresh integer variables and x_i^{real}, z_j^{real} are fresh real variables that express the integer and real part of x_i and z_j. If x_i (resp. z_j) is an integer variable, we add the constraint $x_i^{\mathrm{real}} = 0$ (resp. $z_j^{\mathrm{real}} = 0$) to the separation. We say that an existential formula in LIRA is *decomposable* if its separation can be written as an existentially quantified Boolean combination of Presburger and LRA formulas, called *decomposition*.

Lemma 2.4.5. *Every existential formula in LIRA is decomposable. Moreover, its decomposition is of linear size and can be computed in polynomial time.*

Proof. Let $\psi = \exists x_1, \ldots, x_n \colon \varphi(x_1, \ldots, x_n, z_1, \ldots, z_m)$ be an existential formula in LIRA. By introducing new existentially quantified variables, we can assume that every atom of φ is of one of the following forms:

$$\text{(i) } x = 0, \qquad \text{(ii) } x = 1, \qquad \text{(iii) } x + y = z, \qquad \text{(iv) } x < 0, \qquad \text{(v) } x = \lfloor y \rfloor.$$

Note that the size of the formula is still linear, even if the coefficients are given in binary. To see this, let ax be a term where $a \in \mathbb{N}$ has the binary expansion $b_0 \ldots b_n$ with least significant bit left. For every $i \in [0, n]$ introduce a fresh existentially quantified variable y_i with the constraint $y_i = y_{i-1} + y_{i-1}$ if $i \in [1, n]$ and $y_0 = x$. Then ax can be replaced with the sum of all y_i where $b_i = 1$, which in turn can be transformed into the above form by introducing new existentially quantified variables.

Let $\varphi'(\boldsymbol{x}^{\mathrm{i/r}}, \boldsymbol{z}^{\mathrm{i/r}})$ be the formula obtained from $\varphi(\boldsymbol{x}^{\mathrm{int}} + \boldsymbol{x}^{\mathrm{real}}, \boldsymbol{z}^{\mathrm{int}} + \boldsymbol{z}^{\mathrm{real}})$ by replacing every atom of the form

(i) $x^{\mathrm{int}} + x^{\mathrm{real}} = 0$ with $x^{\mathrm{int}} = 0 \wedge x^{\mathrm{real}} = 0$,

(ii) $x^{\mathrm{int}} + x^{\mathrm{real}} = 1$ with $x^{\mathrm{int}} = 1 \wedge x^{\mathrm{real}} = 0$,

(iii) $x^{\text{int}} + x^{\text{real}} + y^{\text{int}} + y^{\text{real}} = z^{\text{int}} + z^{\text{real}}$ with

$$(x^{\text{real}} + y^{\text{real}} < 1 \to x^{\text{int}} + y^{\text{int}} = z^{\text{int}} \wedge x^{\text{real}} + y^{\text{real}} = z^{\text{real}}) \wedge$$
$$(x^{\text{real}} + y^{\text{real}} \geq 1 \to x^{\text{int}} + y^{\text{int}} + 1 = z^{\text{int}} \wedge x^{\text{real}} + y^{\text{real}} - 1 = z^{\text{real}}),$$

(iv) $x^{\text{int}} + x^{\text{real}} < 0$ with $x^{\text{int}} < 0$, and

(v) $x^{\text{int}} + x^{\text{real}} = \lfloor y^{\text{int}} + y^{\text{real}} \rfloor$ with $x^{\text{real}} = 0 \wedge x^{\text{int}} = y^{\text{int}}$

where $x^{\text{int}}, y^{\text{int}}, z^{\text{int}}$ refer to an x_i^{int} or z_i^{int} and $x^{\text{real}}, y^{\text{real}}, z^{\text{real}}$ refer to an x_i^{real} or z_i^{real}. Thus, φ' is a Boolean combination of formulas that either only involve integer variables or real variables. Now, the separation of ψ is equivalent to

$$\exists \boldsymbol{x}^{\text{i/r}} \colon \varphi'(\boldsymbol{x}^{\text{i/r}}, \boldsymbol{z}^{\text{i/r}}) \wedge \mathbf{0} \leq \boldsymbol{x}^{\text{real}} \ll \mathbf{1} \wedge \mathbf{0} \leq \boldsymbol{z}^{\text{real}} \ll \mathbf{1}$$

where we add $x_i^{\text{real}} = 0$ if x_i is an integer variable and $z_j^{\text{real}} = 0$ if z_j is an integer variable, which is a linear sized decomposition. □

Using Lemma 2.4.5, we can show satisfiability of existential formulas in LIRA has the same complexity as for LIA and LRA. This was already shown in [QSW17] under the assumption that the LIRA formula is already decomposed.

Proposition 2.4.6. *Satisfiability of existential formulas in LIRA is* NP*-complete.*

Proof. The NP lower bound is inherited from the Presburger (Proposition 2.4.1) and LRA (Proposition 2.4.3) case. For the upper bound let φ be an existential formula in LIRA. We first apply Lemma 2.4.5 to compute a decomposition ψ of φ in polynomial time. Then we guess truth values for the Presburger and LRA subformulas of ψ and verify the guesses in NP using Propositions 2.4.1 and 2.4.3. Since φ and its decomposition ψ are equisatisfiable, it remains to check whether the truth values satisfy ψ in order to decide satisfiability of φ. □

2.4.5 Automatic Structures

The notion of automatic structures first appeared in the PhD thesis of Hodgson [Hod76] and was later rediscovered by Khoussainov and Nerode [KN94]. Their systematic study started with Blumensath and Grädel [BG00; BG04]. We refer to the survey by Grädel [Grä20] for a more detailed overview. Intuitively, automatic structures are (possibly infinite) structures that admit a finite representation in terms of automata. Loosely speaking, this means that both the domain and the relations are regular and given by finite automata.

Before we give a formal definition of automatic structures, we argue that we can focus on *relational structures*, i.e. structures that only consist of relations. Let $\sigma = ((R_i)_{i \in N}, (f_j)_{j \in M})$ be a signature with relation symbols R_i and function symbols f_j of arity r_j and let $\mathfrak{A} = (A, I)$ be a σ-structure. We define the signature $\sigma' := ((R_i)_{i \in N}, (F_j)_{j \in M})$ with relation symbols F_j of arity $r_j + 1$ and the relational σ'-structure $\mathfrak{A}' := (A, I')$ where $I'(R_i) := I(R_i)$ for all $i \in N$ and $I'(F_j) :=$

$\{(a_1, \ldots, a_{r_j+1}) \in A^{r_j+1} \mid f_j(a_1, \ldots, a_{r_j}) = a_{r_j+1}\}$ for all $j \in M$. Now we can transform a formula φ over σ to a formula φ' over σ' by replacing each function application of the form $f_j(t_1, \ldots, t_{r_j})$ with a fresh existentially quantified variable x with the constraint $F_j(t_1, \ldots, t_{r_j}, x)$. Then we have that $\mathfrak{A}, \nu \models \varphi$ if and only if $\mathfrak{A}', \nu \models \varphi'$.

A relational structure $\mathfrak{A}$ over the domain A is *(tree-)automatic* if there exist a (tree-)regular language L_A and a surjective function $\alpha\colon L_A \to A$ such that the relation

$$L_= := \{(v, w) \in L_A \times L_A \mid \alpha(v) = \alpha(w)\}$$

and the relations

$$L_R := \{\boldsymbol{a} \in (L_A)^r \mid (\alpha(a_1), \ldots, \alpha(a_r)) \in R\}$$

for all r-ary relations R of $\mathfrak{A}$ are (tree-)regular. We also say that non-relational structures are (tree-)automatic if their relational versions are.

A *presentation* of an (tree-)automatic structure $\mathfrak{A} = \langle A; (R_i)_{i \in N}\rangle$ is a tuple $\mathfrak{p} = (\mathcal{A}_A, \mathcal{A}_=, (\mathcal{A}_{R_i})_{i \in N})$ of (tree) automata for L_A, $L_=$, and L_{R_i}, respectively. We will in the following always assume that (tree-)automatic structures are given by presentations (i.e. up to isomorphism). For a formula $\varphi(x_1, \ldots, x_n)$ over $\mathfrak{A}$ we write

$$[\![\varphi]\!]_\mathfrak{p} := \{(w_1, \ldots, w_n) \in (L_A)^n \mid \mathfrak{A} \models \varphi(\alpha(w_1), \ldots, \alpha(w_n))\}.$$

We say that $\mathfrak{p}$ is *injective* if $L_= = \{(w, w) \mid w \in L_A\}$, which implies that $\alpha\colon L_A \to A$ is injective. It was shown by Khoussainov and Nerode [KN94] over finite words and Colcombet and Löding [CL07] over finite trees that injective presentations can capture all (tree-)automatic structures:

Proposition 2.4.7. *Every (tree-)automatic structure admits an injective presentation.*

Automatic structures can also be defined over infinite words. An *ω-automatic structure* is defined similarly as an automatic structure by replacing regular languages and relations with ω-regular languages and relations. Likewise, a presentation of an ω-automatic structure is a tuple of NBAs. Note that the analogue of Proposition 2.4.7 for ω-automatic structures does not hold [Hjo+08].

A nice property of (ω-/tree-)automatic structures is that their theories are decidable.

Proposition 2.4.8. *Let $\mathfrak{A}$ be an (ω-/tree-)automatic structure given by a presentation $\mathfrak{p}$. Then for any formula φ over $\mathfrak{A}$ one can effectively construct an NFA (resp. NBA/NTA) for the relation $[\![\varphi]\!]_\mathfrak{p}$. In particular,* $\mathrm{Th}(\mathfrak{A})$ *is decidable.*

Proof. Let $\mathfrak{A}$ be given by the presentation $\mathfrak{p} = (\mathcal{A}_A, \mathcal{A}_=, (\mathcal{A}_{R_i})_{i \in N})$ and φ be a formula over $\mathfrak{A}$. We construct the NFA (resp. NBA/NTA) $\mathcal{A}$ for $[\![\varphi]\!]_\mathfrak{p}$ inductively. Assume $\psi_1(\boldsymbol{x})$ and $\psi_2(\boldsymbol{x})$ are formulas over $\mathfrak{A}$ and $\mathcal{A}_1$ and $\mathcal{A}_2$ are NFAs (resp. NBAs/NTAs) for $[\![\psi_1]\!]_\mathfrak{p}$ and $[\![\psi_1]\!]_\mathfrak{p}$, respectively. Note that w.l.o.g. we can assume that ψ_1 and ψ_2 have the same vector of free variables since otherwise we can extend the automata $\mathcal{A}_1$ and $\mathcal{A}_2$ with additional components.

1) If $\varphi = R_i(\boldsymbol{x})$ for some $i \in N$ and variables $\boldsymbol{x}$, then let $\mathcal{A} := \mathcal{A}_{R_i}$.

2) If $\varphi = (x_1 = x_2)$ for variables x_1, x_2, then let $\mathcal{A} := \mathcal{A}_=$.

3) If $\varphi = \psi_1 \wedge \psi_2$, then construct $\mathcal{A}$ with $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2) = \otimes[\![\varphi]\!]_\mathfrak{p}$ from $\mathcal{A}_1$ and $\mathcal{A}_2$ using Proposition 2.3.4 (resp. Propositions 2.3.10 and 2.3.15).

4) If $\varphi = \psi_1 \vee \psi_2$, then construct $\mathcal{A}$ with $L(\mathcal{A}) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2) = \otimes[\![\varphi]\!]_\mathfrak{p}$ from $\mathcal{A}_1$ and $\mathcal{A}_2$ using Proposition 2.3.4 (resp. Propositions 2.3.10 and 2.3.15).

5) If $\varphi = \neg\psi_1$, then construct $\mathcal{A}$ with $L(\mathcal{A}) = \overline{L(\mathcal{A}_1)} \cap (\otimes A^{|\boldsymbol{x}|}) = \otimes[\![\varphi]\!]_\mathfrak{p}$ from $\mathcal{A}_1$ using Proposition 2.3.4 (resp. Propositions 2.3.10 and 2.3.15), where an automaton for $A^{|\boldsymbol{x}|}$ can be constructed using $\mathcal{A}_A$.

6) If $\varphi = \exists x_i \colon \psi_1(x_1, \ldots, x_k)$ for variables $x_1, \ldots, x_k$ and $i \in [1, k]$, then construct $\mathcal{A}$ with $L(\mathcal{A}) = \pi_{[1,k]\setminus\{i\}}(L(\mathcal{A}_1)) = \otimes[\![\varphi]\!]_\mathfrak{p}$ from $\mathcal{A}_1$ using Proposition 2.3.27 (resp. Propositions 2.3.30 and 2.3.33).

Now, φ is satisfiable if and only if $L(\mathcal{A}) \neq \emptyset$, which can be checked using Proposition 2.3.5 (resp. Propositions 2.3.12 and 2.3.17). □

Example 2.4.9. A famous example of an automatic structure is Presburger arithmetic. For simplicity, let us assume that Presburger arithmetic is the structure $\langle \mathbb{N}_0; + \rangle$. It is easy to see that also the full structure $\langle \mathbb{Z}; <, +, 0, 1 \rangle$ is automatic. The idea is to encode numbers in binary with least significant bit first. Let $L_{\mathbb{N}_0} := \{0, 1\}^+$ and $\alpha \colon L_{\mathbb{N}_0} \to \mathbb{N}_0$ be the function that maps a binary string to the natural number it encodes. Then the relations $L_= := \{(v, w) \in L_{\mathbb{N}_0} \times L_{\mathbb{N}_0} \mid \alpha(v) = \alpha(w)\}$ and $L_+ := \{(u, v, w) \in (L_{\mathbb{N}_0})^3 \mid \alpha(u) + \alpha(v) = \alpha(w)\}$ are regular, where the NFA for L_+ proceeds like the standard algorithm for addition of binary numbers by storing a carry bit in its states. Note that we can make α injective by removing trailing zeros.

We can extend Presburger arithmetic with the unary function V_p for some $p \geq 2$, where $V_p(x)$ is defined as the greatest power of p dividing x, and remain automatic (by encoding numbers in base p). The structure $\langle \mathbb{N}_0; +, V_p \rangle$, called *Büchi arithmetic*, forms a so-called *universal automatic structure*, i.e. every automatic structure is first-order interpretable in it.

An example of a tree-automatic structure that is not word-automatic is *Skolem arithmetic* $\langle \mathbb{N}_0; \cdot \rangle$. Thus, this example witnesses that the word-automatic structures form a strict subclass of tree-automatic structures. For the tree-automatic presentation of $\langle \mathbb{N}_0; \cdot \rangle$ one can encode a natural number as a tree where branches encode the exponents of prime powers in the prime factorization. Then multiplication reduces to addition of the exponents.

Examples of ω-automatic structures are LRA and LIRA. Here, real numbers can be encoded as an infinite word of pairs of bits where the first component encodes the integral part and the second component encodes the fractional part. The NBA for addition guesses at the beginning whether the sum of the fractional parts produces a carry and then verifies the sum of the first components and the second components separately. For the floor function of LIRA the NBA can for non-negative numbers simply check whether the result has the same integral part and only zeros in the fractional part (for negative

numbers it subtracts one from the integral part if the fractional part is non-zero). Note that LRA and LIRA are clearly not (tree-)automatic structures (over finite words/trees) since their domain is uncountable, meaning that there is no surjective function from a (tree-)regular language onto it.

3 Ramsey Quantifiers and Their Applications

In this chapter we introduce the central notion of this thesis, the Ramsey quantifier. After the formal definition in Section 3.1, we introduce two major applications of Ramsey quantifiers in Sections 3.2 and 3.3, respectively: liveness verification over infinite-state transition systems and recognizability and the related concept of monadic decomposability.

3.1 Ramsey Quantifiers

Intuitively, Ramsey quantifiers state the existence of an infinite clique in the graph defined by a formula. They generalize the infinity quantifier in the sense that they allow to not only check for an infinite set in which every element satisfies a condition, but even that a condition holds between every pair (or n-tuple in general) over the set. Ramsey quantifiers were introduced by Magidor and Malitz [MM77], which is why they are also known as *Magidor-Malitz quantifiers*.

Let $\mathfrak{A}$ be a structure with domain A and $\varphi(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, \boldsymbol{z})$ with $n \geq 1$ be a formula over $\mathfrak{A}$ such that there is some $d \geq 1$ with $|\boldsymbol{x}_i| = d$ for all $i \in [1, n]$. The *n-Ramsey quantifier* $\exists^{n\text{-ram}}$ of dimension d is defined such that $\mathfrak{A} \models \exists^{n\text{-ram}} \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \colon \varphi(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, \boldsymbol{c})$ for some $\boldsymbol{c} \in A^{|\boldsymbol{z}|}$ if and only if there exists an infinite sequence $(\boldsymbol{a}_i)_{i \geq 1}$ of pairwise distinct tuples $\boldsymbol{a}_i \in A^d$ such that $\mathfrak{A} \models \varphi(\boldsymbol{a}_{i_1}, \ldots, \boldsymbol{a}_{i_n}, \boldsymbol{c})$ for all $1 \leq i_1 < \cdots < i_n$. For every $n \geq 1$ we denote by $\mathrm{Th}^{\exists^{n\text{-ram}}}(\mathfrak{A})$ the theory of $\mathfrak{A}$ under first-order logic enriched with $\exists^{n\text{-ram}}$ of all dimensions $d \geq 1$.

We will mostly only deal with the 2-Ramsey quantifier (just called Ramsey quantifier in the following), which we denote by $\exists^{\mathrm{ram}}$. In that case, the formula $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ defines for every valuation of $\boldsymbol{z}$ the edge relation of a directed graph. Then we call a sequence $(\boldsymbol{a}_i)_{i \geq 1}$ witnessing $\mathfrak{A} \models \exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c})$ an *infinite (directed) clique* of φ with respect to $\boldsymbol{c}$. The *infinite clique problem* asks whether $\mathfrak{A} \models \exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y})$ for a given formula $\varphi(\boldsymbol{x}, \boldsymbol{y})$ over some fixed structure $\mathfrak{A}$ or in other words, whether $\varphi(\boldsymbol{x}, \boldsymbol{y})$ contains an infinite clique. For (ω-/tree-)regular relations the infinite clique problem refers to the problem of checking whether a given (ω-/tree-)regular relation $R \subseteq A^{2d}$ satisfies $\mathfrak{A} \models \exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R(\boldsymbol{x}, \boldsymbol{y})$, where $\mathfrak{A} := \langle A; R \rangle$ is the structure implicitly defined by R.

Ramsey quantifiers are named after Frank P. Ramsey due to his seminal work on cliques. The infinite version of Ramsey's theorem [Ram30] can be formulated over graphs as follows:

Theorem 3.1.1 (Ramsey's theorem)**.** *Any complete infinite undirected graph whose edges are colored with finitely many colors contains an infinite monochromatic clique.*

Proof. Let $G = (V, E)$ be a complete infinite undirected graph, i.e. with an infinite set of vertices V and edges $E := \{\{u, v\} \mid u, v \in V, u \neq v\}$. Let $c\colon E \to \{1, \dots, k\}$ be a coloring of the edges with $k \geq 1$ colors. Take an arbitrary vertex $v_1 \in V$. By the pigeonhole principle, there exists an infinite subset $V_1 \subseteq V \setminus \{v_1\}$ and color $r_1 \in [1, k]$ such that $c(\{v_1, w\}) = r_1$ for all $w \in V_1$. Now take an arbitrary vertex $v_2 \in V_1$ and obtain the infinite set $V_2 \subseteq V_1 \setminus \{v_2\}$ and color $r_2 \in [1, k]$ with the same argument as for v_1. Inductively, we obtain an infinite sequence $(v_i)_{i \geq 1}$ of vertices such that $c(\{v_i, v_j\}) = r_i$ for all $1 \leq i < j$. Again, by the pigeonhole principle there exists an infinite subsequence $(v_{i_j})_{j \geq 1}$ and color $r \in [1, k]$ such that $r_{i_j} = r$ for all $j \geq 1$. Thus, $(v_{i_j})_{j \geq 1}$ forms a monochromatic clique in G. □

We will encounter similar proof strategies with repeated applications of pigeonhole principle and Ramsey's theorem in Chapter 4. Note that by induction, Ramsey's theorem can also be generalized to hypergraphs, which will be used in Section 4.6. We will often use the fact that by Ramsey's theorem, $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y}\colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \vee \psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ is equivalent to $(\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y}\colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})) \vee (\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y}\colon \psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}))$.

Note that we deviate from the undirected definition of the Ramsey quantifier found in the literature (see e.g. [MM77]), which we denote by $\exists^{\mathrm{ram}'}$, requiring $\mathfrak{A} \models \varphi(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{c})$ for all $i \neq j$ (instead of $i < j$) in the definition above. This definition corresponds to an undirected clique notion. Clearly, the undirected version is expressible in our definition of the directed version by enforcing a clique in both directions. If an infinite well-partial-ordering is definable in the structure, then also the reverse direction can easily be shown.

Proposition 3.1.2. *Let $\mathfrak{A}$ be a structure. Given a formula $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ over $\mathfrak{A}$, one can compute in logspace a formula $\psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ of linear size such that*

$$\exists^{\mathrm{ram}'} \boldsymbol{x}, \boldsymbol{y}\colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \equiv \exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y}\colon \psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}).$$

Conversely, if an infinite well-partial-ordering is definable in $\mathfrak{A}$, then given a formula $\psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ over $\mathfrak{A}$, one can compute in logspace a formula $\varphi(\boldsymbol{x}', \boldsymbol{y}', \boldsymbol{z})$ of linear size such that

$$\exists^{\mathrm{ram}'} \boldsymbol{x}', \boldsymbol{y}'\colon \varphi(\boldsymbol{x}', \boldsymbol{y}', \boldsymbol{z}) \equiv \exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y}\colon \psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}).$$

Proof. For the first part we construct from the given formula $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ the formula $\psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) := \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \wedge \varphi(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{z})$. Let $\boldsymbol{c} \in A^{|z|}$ where A is the domain of $\mathfrak{A}$. It is easy to see that there is an infinite undirected clique in φ with respect to $\boldsymbol{c}$ if and only if there is an infinite directed clique in ψ with respect to $\boldsymbol{c}$.

For the second part of the proposition assume that $(B, \leq)$ is an infinite well-partial-ordering definable in $\mathfrak{A} = (A, I)$ and denote by $<$ the corresponding strict ordering, i.e. where $a < b$ if and only if $a \leq b$ and $b \not\leq a$ for all $a, b \in A$. We construct from the given formula $\psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ the formula

$$\varphi(\boldsymbol{x}', \boldsymbol{y}', \boldsymbol{z}) := \boldsymbol{x} \neq \boldsymbol{y} \wedge (\psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \wedge r < s \vee \psi(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{z}) \wedge s < r)$$

where $\boldsymbol{x}' = (\boldsymbol{x}, r)$ and $\boldsymbol{y}' = (\boldsymbol{y}, s)$ for variables r, s. Let $\boldsymbol{c} \in A^{|z|}$. If ψ has an infinite directed clique $(\boldsymbol{a}_i)_{i\geq 1}$ with respect to $\boldsymbol{c}$, then we can number the elements with some $b_1 < b_2 < \dots$ from B and get an infinite undirected clique $((\boldsymbol{a}_i, b_i))_{i\geq 1}$ of φ with respect to $\boldsymbol{c}$. Note that every infinite well-partial-ordering contains such an infinite strictly increasing sequence $(b_i)_{i\geq 1}$. Reversely, if φ has an infinite undirected clique $((\boldsymbol{a}_i, b_i))_{i\geq 1}$, then $(\boldsymbol{a}_{i_j})_{j\geq 1}$ with $b_{i_j} < b_{i_{j'}}$ for all $1 \leq j < j'$ is an infinite directed clique of ψ with respect to $\boldsymbol{c}$. Again, since $(B, \leq)$ is a well-partial-ordering, the infinite sequence $(b_i)_{i\geq 1}$ of pairwise distinct elements of B contains a strictly increasing subsequence $(b_{i_j})_{j\geq 1}$. □

Thus, complexity upper bounds that are obtained for the directed version of the Ramsey quantifier always directly carry over to the undirected version. For structures like Presburger arithmetic or LIRA, Proposition 3.1.2 can also be applied to transfer lower bounds from the directed to the undirected version since there the well-partial-ordering $(\mathbb{N}, \leq)$ is definable. But it is easy to see that also for LRA, the lower bounds established in this thesis already hold for the undirected version. Over (ω-/tree-)automatic structures we observe that we can always construct an automaton going from the directed to the undirected version and vice versa. This implies that all complexity results that we obtain for the directed version also hold true in the undirected case.

Proposition 3.1.3. *Given an NFA (resp. NBA/NTA) for a $(2d + k)$-ary (ω-/tree-)regular relation R, one can compute in logspace an NFA (resp. NBA/NTA) for a $(2d + k)$-ary relation S such that*

$$[\![\exists^{\mathrm{ram}'} \boldsymbol{x}, \boldsymbol{y} \colon R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!] = [\![\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon S(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!]$$

where $|\boldsymbol{x}| = |\boldsymbol{y}| = d$ and $|\boldsymbol{z}| = k$. Conversely, given an NFA (resp. NBA/NTA) for a $(2d + k)$-ary (ω-/tree-)regular relation S, one can compute in logspace an NFA (resp. NBA/NTA) for a $(2(d + 1) + k)$-ary relation R such that

$$[\![\exists^{\mathrm{ram}'} \boldsymbol{x}', \boldsymbol{y}' \colon R(\boldsymbol{x}', \boldsymbol{y}', \boldsymbol{z})]\!] = [\![\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon S(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!]$$

where $|\boldsymbol{x}| = |\boldsymbol{y}| = d$, $|\boldsymbol{x}'| = |\boldsymbol{y}'| = d + 1$, and $|\boldsymbol{z}| = k$.

Proof. We proceed as in the proof of Proposition 3.1.2. For the first part let R be given by an NFA (resp. NBA/NTA) $\mathcal{A}$. We define $S := \{(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \mid (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \in R \wedge (\boldsymbol{b}, \boldsymbol{a}, \boldsymbol{c}) \in R\}$. Clearly, S is a (ω-/tree-)regular relation and an NFA (resp. NBA/NTA) for S can be constructed in logspace from $\mathcal{A}$ using Proposition 2.3.4 (resp. Propositions 2.3.10 and 2.3.15).

For the second part let S be given by an NFA (resp. NBA/NTA) $\mathcal{A}$. Then we define

$$R := \{(\boldsymbol{a}, i, \boldsymbol{b}, j, \boldsymbol{c}) \mid i, j \in \mathbb{N} \wedge \boldsymbol{a} \neq \boldsymbol{b} \wedge ((\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \in S \wedge i < j \vee (\boldsymbol{b}, \boldsymbol{a}, \boldsymbol{c}) \in S \wedge j < i)\}.$$

It is easy to see that R can be encoded as a (ω-/tree-)regular relation and an NFA (resp. NBA/NTA) recognizing this relation can be constructed in logspace from $\mathcal{A}$ using Proposition 2.3.4 (resp. Propositions 2.3.10 and 2.3.15). Here, note that the automaton for $\boldsymbol{a} \neq \boldsymbol{b}$ is not too large since we can restrict it to the alphabet implicitly given by $\mathcal{A}$. □

Over (ω-/tree-)regular relations we can use the convolution operation to reduce the evaluation of the Ramsey quantifier for a $(2d+k)$-ary relation to the case where $d = 1$.

Proposition 3.1.4. *Given an automaton for a $(2d + k)$-ary (ω-/tree-)regular relation R, one can compute in logspace an automaton for a $(k+2)$-ary (ω-/tree-)regular relation R' such that*

$$[\![\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!] = [\![\exists^{\mathrm{ram}} x, y \colon R'(x, y, \boldsymbol{z})]\!]$$

where $|\boldsymbol{x}| = |\boldsymbol{y}| = d$ and $|\boldsymbol{z}| = k$. Moreover, the construction preserves (co-)transitivity of relations and determinism of automata.

Proof. For the sake of an easier notation, we only show the finite-word case and note that the infinite-word case and the tree case are similar. Let $R \subseteq (\Sigma^*)^{2d+k}$ be given by an NFA $\mathcal{A}$. The idea is to summarize the components of $\boldsymbol{x}$ (resp. $\boldsymbol{y}$) in just one component using the convolution operation. For a tuple $\boldsymbol{w} = (w_1, \ldots, w_n)$ of words let $\otimes \boldsymbol{w} := w_1 \otimes \cdots \otimes w_n$ be the convolution of all of its components. We define the $(k + 2)$-ary relation

$$R' := \big\{(\otimes \boldsymbol{u}, \otimes \boldsymbol{v}, \boldsymbol{w}) \mid \boldsymbol{u}, \boldsymbol{v} \in (\Sigma^*)^d, \boldsymbol{w} \in (\Sigma^*)^k, (\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}) \in R\big\} \subseteq \big((\Sigma')^*\big)^{k+2}$$

where $\Sigma' := (\Sigma_\perp)^d \cup \Sigma$. We can construct an NFA $\mathcal{A}'$ for R' from $\mathcal{A}$ in logspace by replacing every transition label of the form $(a_1, \ldots, a_d, b_1, \ldots, b_d, c_1, \ldots, c_k) \in (\Sigma_\perp)^{2d+k}$ with $(a', b', c_1, \ldots, c_k) \in (\Sigma'_\perp)^{k+2}$ where $a' := \perp$ if $a_1 = \cdots = a_d = \perp$ and $a' := (a_1, \ldots, a_d)$ otherwise and similarly for b'. Note that $\mathcal{A}'$ accepts the relation $\otimes R'$ that first takes the convolution of the components of $\boldsymbol{x}$ and $\boldsymbol{y}$ separately and then takes the convolution of the results with the components of $\boldsymbol{z}$. This is not the same as $\otimes R$ since the convolution operation is not associative as illustrated in Example 2.3.26. It remains to show that

$$[\![\exists^{\mathrm{ram}} x, y \colon R'(x, y, \boldsymbol{z})]\!] = [\![\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!].$$

Let $\boldsymbol{w} \in (\Sigma^*)^k$. If $(u_i)_{i\geq 1}$ with $u_i \in \big((\Sigma_\perp)^d\big)^*$ is a clique of R' with respect to $\boldsymbol{w}$, then $(\boldsymbol{v}_i)_{i\geq 1}$ with $\boldsymbol{v}_i \in (\Sigma^*)^d$ such that $u_i = \otimes \boldsymbol{v}_i$ is a clique of R with respect to $\boldsymbol{w}$. Conversely, if $(\boldsymbol{v}_i)_{i\geq 1}$ with $\boldsymbol{v}_i \in (\Sigma^*)^d$ is a clique of R with respect to $\boldsymbol{w}$, then $(u_i)_{i\geq 1}$ with $u_i = \otimes \boldsymbol{v}_i$ is a clique of R' with respect to $\boldsymbol{w}$.

Clearly, if R is (co-)transitive, then so is R'. Moreover, if the automaton for R is deterministic, then so is the automaton for R'. In the tree case we further remark that also the direction (bottom-up or top-down) of the automaton is preserved. □

3.2 Liveness Verification

Loosely speaking, liveness verification amounts to checking whether in a system "something good eventually happens". The reason why liveness properties are regarded as more difficult to check than safety properties, which essentially can be reduced to reachability via finite paths, is that liveness requires to reason about infinite paths since a counterexample for a liveness condition is an infinite run in the system where good behavior is never observed. Since Ramsey quantifiers (in their directed semantics) allow

the reasoning about infinite (transitive) paths, they can be applied to liveness verification whenever the reachability relation, i.e. the transitive closure of the transition relation, of the system can be defined in some logic that admits evaluation of the Ramsey quantifier. We already saw in Algorithm 1 a concrete example of a program where the Ramsey quantifier can be used for proving non-termination since the reachability relation is expressible in LIRA.

Let us make this precise. As our definition of systems we take the well-studied notion of transition systems. A *transition system* is a pair $(C, \to)$ where C is a set of *configurations* and $\to \subseteq C \times C$ is a *step relation* (a.k.a. *transition relation*). We call the transitive closure $\to^+$ of $\to$ the *reachability relation*, whose reflexive closure is denoted by $\to^*$. Note that the set of configurations C is potentially infinite, in which case the transition system is said to be *infinite-state*. We will encounter several examples of infinite-state transition systems that have finite representations.

3.2.1 Recurrent Reachability

For a binary relation $R \subseteq A^2$ we call an infinite sequence $(a_i)_{i \geq 1}$ of elements in A a *transitive path* in R if $(a_i, a_j) \in R$ for all $1 \leq i < j$. Note that the only difference to an infinite clique is that the elements need not to be pairwise distinct. Given a binary relation $R \subseteq A^2$ and sets $L_1, \ldots, L_k \subseteq A$, we write $Rec(L_1, \ldots, L_k)[R]$ for the set of all initial elements a_1 of transitive paths $(a_i)_{i \geq 1}$ in R that visit each set L_j infinitely often. *Recurrent reachability with generalized Büchi condition* is the problem of testing whether $a_1 \in Rec(L_1, \ldots, L_k)[R]$ for given relation $R \subseteq A^2$, sets $L_1, \ldots, L_k \subseteq A$, and initial element $a_1 \in A$. If $k = 1$, this problem is simply called *recurrent reachability*.

If R is the reachability relation of a transition system, then recurrent reachability (with generalized Büchi condition) corresponds to the liveness problem of checking whether a given set (resp. list of sets) of configurations is reachable infinitely often via some run starting in a given initial configuration. We also consider the problem where instead of the reachability relation only a representation of the transition system is given. As already motivated in the introduction, LTL model checking, in which various liveness properties are expressible, can for some classes of systems (e.g. pushdown systems [BEM97]) be reduced to recurrent reachability [VW86].

In Section 4.4.1 we consider recurrent reachability with generalized Büchi condition for transition systems where the reachability relation is (tree-)regular, i.e. finitely representable using (tree) automata. Concrete examples of such systems are pushdown systems [Esp+00], ground tree rewriting systems [Löd06], and PA-processes [LS05]. Since (tree) automata for the reachability relations of all of these systems can be computed in polynomial time given a description of the transition system, it suffices to consider recurrent reachability with generalized Büchi condition over (tree-)regular relations. More precisely, in the above definition we assume $R \subseteq A^{2d}$ and $L_1, \ldots, L_k \subseteq A^d$ to be (tree-)regular relations given by (tree) automata. Note that, unlike in the case when R is a reachability relation, we do not assume R to be transitive. In the liveness setting this corresponds to the situation where only an (not necessarily transitive) approximation of the reachability relation is provided as an input. We remark that in this

setting the problem definition with transitive paths is crucial since for (not necessarily transitive) regular relations the reachability problem via not necessarily transitive paths is already undecidable [BG04]. To be able to apply our results developed in Chapter 4, we show that recurrent reachability and the infinite clique problem over (tree-)regular relations are interreducible.

Proposition 3.2.1. *The infinite clique problem and recurrent reachability are logspace equivalent over (tree-)regular relations. Moreover, the logspace reduction from recurrent reachability to the infinite clique problem produces a binary relation and preserves transitivity of relations and determinism of automata.*

Proof. We start with the reduction from the infinite clique problem to recurrent reachability. Let $R \subseteq A^{2d}$ be a (tree-)regular relation given by an NFA (resp. NTA) $\mathcal{A}$. Fix some vector $\boldsymbol{a}_1 \in A^d$. We define the relation $R' \subseteq A^{2d}$ such that

(i) $(\boldsymbol{a}_1, \boldsymbol{a}) \in R'$ for all $\boldsymbol{a} \in A^d$ and

(ii) $(\boldsymbol{a}, \boldsymbol{b}) \in R'$ for all $(\boldsymbol{a}, \boldsymbol{b}) \in R$ with $\boldsymbol{a} \neq \boldsymbol{b}$.

Clearly, the relation R' is (tree-)regular and an NFA (resp. NTA) for R' is logspace computable from $\mathcal{A}$. Since the additional condition $\boldsymbol{a} \neq \boldsymbol{b}$ in (ii) ensures that the vectors of a transitive path in R' are pairwise distinct, it holds that R has an infinite clique if and only if $\boldsymbol{a}_1 \in \mathit{Rec}(A^d)[R']$.

For the reverse reduction let $R \subseteq A^{2d}$ be a (tree-)regular relation given by an NFA (resp. NTA) $\mathcal{A}$ and $L \subseteq A^d$ be a (tree-)regular relation given by an NFA (resp. NTA) $\mathcal{B}$. Furthermore, let $\boldsymbol{a}_1 \in A^d$ be the initial vector. We define the relation $R' \subseteq (A^d \times \mathbb{N}) \times (A^d \times \mathbb{N})$ such that for all $\boldsymbol{a}, \boldsymbol{b} \in A^d$ and $m, n \in \mathbb{N}$ we have $((\boldsymbol{a}, m), (\boldsymbol{b}, n)) \in R'$ if and only if

(i) $(\boldsymbol{a}_1, \boldsymbol{a}) \in R$,

(ii) $(\boldsymbol{a}, \boldsymbol{b}) \in R$, and

(iii) $\boldsymbol{a} \in L$.

Intuitively, we create infinitely many copies of every vector by taking the direct product with the natural numbers. This allows the infinite clique to visit an original vector several times. Furthermore, in R' we only consider the vectors that are in relation with $\boldsymbol{a}_1$ to ensure that $\boldsymbol{a}_1$ can be appended to the beginning of every infinite clique. With the third condition we ensure that every vector of the infinite clique is contained in L. Thus, $\boldsymbol{a}_1 \in \mathit{Rec}(L)[R]$ if and only if R' has an infinite clique.

Note that R' can be viewed as a subset of $A^{2(d+1)}$ by representing the natural numbers in unary as words (resp. paths). Then R' is (tree-)regular and an automaton for it can easily be constructed in logspace from $\mathcal{A}$ and $\mathcal{B}$ using the product construction from Proposition 2.3.4 (resp. Proposition 2.3.15). Moreover, if R is transitive, then so is R' and if $\mathcal{A}$ and $\mathcal{B}$ are deterministic, then so is the automaton for R'. In the tree case we further remark that also the direction (bottom-up or top-down) of the automata is preserved. Furthermore, by Proposition 3.1.4, R' can be transformed into a binary relation. □

3.2.2 Linear Liveness

In Section 5.6.3 we consider a similar liveness problem as recurrent reachability for transition systems whose reachability relation can be expressed by a formula in linear integer/real arithmetic. More specifically, the systems that we consider will involve counters and/or clocks. Configurations of such systems will be tuples of integers representing the counter values and reals representing the clock values. The difference to recurrent reachability is that instead of just visiting a set of configurations infinitely often, we require that between every pair of infinitely many configurations on the run a formula in linear integer/real arithmetic holds. The formula may even have additional components whose valuation has to be the same for every pair.

Let us make this more precise. In the transition systems that we consider, configurations are elements of $C := Q \times \mathbb{Z}^k \times D^\ell$ where Q is a finite set of control states and D is either $\mathbb{R}$ or $\mathbb{Q}$. We are interested in transition systems with step relation $\to \subseteq C \times C$ such that for all $p, q \in Q$ one can effectively construct an existential formula $\varphi_{p,q}(\boldsymbol{x}, \boldsymbol{y})$ in LIRA for its reachability relation $\to^+$ restricted to initial state p and target state q. This means $\varphi_{p,q}(\boldsymbol{a}, \boldsymbol{b})$ if and only if $(p, \boldsymbol{a}) \to^+ (q, \boldsymbol{b})$ for all $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}^k \times D^\ell$. Concrete examples of classes of transition systems that satisfy this property include timed automata [CJ99], succinct one-counter automata [Li+20], and continuous vector addition systems with states [BH17]. We will discuss these and more examples in detail in Section 5.6.3. Observe that by assuming that $Q = \{1, \ldots, |Q|\}$, we can construct a formula φ for the reachability relation that includes the state component of the configurations:

$$\varphi(x_0, \boldsymbol{x}, y_0, \boldsymbol{y}) := \bigvee_{p,q \in Q} x_0 = p \wedge y_0 = q \wedge \varphi_{p,q}(\boldsymbol{x}, \boldsymbol{y})$$

Then for all configurations $(p, \boldsymbol{a}), (q, \boldsymbol{b}) \in C$ we have that $\varphi(p, \boldsymbol{a}, q, \boldsymbol{b})$ if and only if $(p, \boldsymbol{a}) \to^+ (q, \boldsymbol{b})$. Thus, in the following we just combine the state and the vector of counter/clock values.

We define the *linear liveness problem* for transition systems with the above properties as follows:

Given A description of a transition system and existential LIRA formulas $\varepsilon(\boldsymbol{x}_0)$ and $\psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$.

Question Is there an infinite sequence $\boldsymbol{a}_0, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots$ of configurations and a vector $\boldsymbol{c}$ such that

- $\varepsilon(\boldsymbol{a}_0)$,
- $\boldsymbol{a}_i \to^+ \boldsymbol{a}_j$ for all $0 \leq i < j$, and
- $\psi(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{c})$ for all $1 \leq i < j$?

Here, ε defines a set of initial configurations. A simplified form of linear liveness is when ψ only depends on $\boldsymbol{x}$, i.e. it states an LIRA condition that must be fulfilled separately by infinitely many configuration of a run. This version of the problem was studied by Dang and Ibarra [DI02]. But ψ can also relate pairs of configurations while involving some

vector of free variables. For example, ψ could require that between $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$ the values in some components have increased by at least some positive value in $\boldsymbol{c}$. With this one can express that real clock values grow unboundedly rather than being convergent.

Given the formulas φ, ε, and ψ, the Ramsey quantifier can be used to express linear liveness as follows:

$$\exists \boldsymbol{x}_0, \boldsymbol{z} \colon \varepsilon(\boldsymbol{x}_0) \wedge \Big(\big(\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}_0, \boldsymbol{x}) \wedge \varphi(\boldsymbol{x}, \boldsymbol{y}) \wedge \psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\big) \vee \big(\exists \boldsymbol{x} \colon \varphi(\boldsymbol{x}_0, \boldsymbol{x}) \wedge \varphi(\boldsymbol{x}, \boldsymbol{x}) \wedge \psi(\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{z})\big)\Big)$$

Here, the first disjunct states the existence of a sequence of pairwise distinct configurations, whereas the second disjunct expresses the existence of a lasso. In Chapter 5 we will see how the Ramsey quantifier in the first disjunct can be eliminated, yielding an existential formula in LIRA. This will result in new complexity bounds for linear liveness for various classes of systems.

3.3 Recognizability and Monadic Decomposability

In this section we introduce the notion of *monadic decomposability* and discuss its connection to the problem of *recognizability*. Loosely speaking, a formula is monadically decomposable if its free variables can be decoupled, i.e. the formula is equivalent to a Boolean combination of formulas that only depend on one free variable. The connection to recognizability will become apparent when considering the index of certain equivalence relations. We will see that in some cases, the problems of recognizability and monadic decomposability reduce to checking whether these equivalence relations have finite index. This characterization is interesting for us since it allows us to apply results on infinite cliques. Here, the main idea is that an equivalence relation has infinite index if and only if its complement contains an infinite clique.

3.3.1 Recognizability via Finite-Index Equivalence Relations

In Sections 2.3.5 and 2.3.6 we defined a hierarchy of classes of relations over (infinite) words. A natural question one can ask is whether a given relation R from some class $\mathbf{C}_1$ also belongs to a strict subclass $\mathbf{C}_2$. We call this a *membership problem*. It was already observed by Fischer and Rosenberg [FR68] and Lisovik [Lis79] that all membership problems over this hierarchy are undecidable if $\mathbf{C}_1$ is the class of (ω-)rational relations. The decidability and complexity status of all other problems is discussed in Chapter 6. In this thesis we will mainly focus on the problem of *(ω-)recognizability*, i.e. the membership problem where $\mathbf{C}_2$ is the class of (ω-)recognizable relations. Here, the key observation, which connects recognizability with Ramsey quantifiers and infinite cliques, is a characterization of recognizability in terms of finite-index equivalence relations.

Let $R \subseteq A^k$ be a k-ary relation. For $I \subseteq \{1, \dots, k\}$ and two tuples $\boldsymbol{a} \in A^{|I|}$ and $\boldsymbol{b} \in A^{k-|I|}$ we define $\boldsymbol{a} \odot_I \boldsymbol{b} \in A^k$ to be the unique k-tuple whose projection to I is $\boldsymbol{a}$

and whose projection to $\{1,\dots,k\} \setminus I$ is $\boldsymbol{b}$. Define the equivalence relation $\approx_I^R$ on $A^{|I|}$ such that

$$\boldsymbol{a} \approx_I^R \boldsymbol{b} \quad :\iff \quad \forall \boldsymbol{c} \in A^{k-|I|} \colon (\boldsymbol{a} \odot_I \boldsymbol{c} \in R \iff \boldsymbol{b} \odot_I \boldsymbol{c} \in R).$$

If R is clear from the context, we will simply write $\approx_I$. If $I = \{j\}$ is a singleton set, we also write $\odot_j$ and $\approx_j$ instead of $\odot_I$ and $\approx_I$.

Example 3.3.1. Let $R_1 := \{(x,y) \mid |x| + |y| \geq 2\}$ over some alphabet Σ. Here, $\approx_1^{R_1}$ has three equivalence classes $\Sigma^{\geq 2}$, Σ, and $\{\varepsilon\}$. We have already seen in Example 2.3.29 that R_1 is recognizable. In Proposition 3.3.3 we show that recognizability of R_1 also follows from finiteness of the index of $\approx_1^{R_1}$.

For a set A we say that a set of sets $T \subseteq 2^A$ is *finer* than a set of sets $S \subseteq 2^A$ if every set in S is a union of sets in T.

Lemma 3.3.2. *Let A be some set and $S = \{S_1,\dots,S_n\} \subseteq 2^A$ be a finite subset of the power set of A. Then the coarsest partition P of $\bigcup_{i=1}^n S_i$ that is finer than S is finite. Moreover, every block of P can be written as a finite Boolean combination of sets in S.*

Proof. For every $I \subseteq [1,n]$ we define the set $B_I := \bigcap_{i \in I} S_i \cap \bigcap_{j \in [1,n] \setminus I} \overline{S_j}$. Clearly, $B_I \cap B_{I'} = \emptyset$ if $I \neq I'$. Now, the set $P := \{B_I \mid I \subseteq [1,n], B_I \neq \emptyset\}$ is the desired partition of $\bigcup_{i=1}^n S_i$. □

The following proposition was shown by Carton, Choffrut, and Grigorieff [CCG06] for rational relations and adapted by Löding and Spinrath [LS19b] to ω-regular relations.

Proposition 3.3.3. *Let $R \subseteq A^k$ be a rational, ω-regular, or tree-regular relation. Then R is recognizable (resp. ω-recognizable or tree-recognizable) if and only if $\approx_{[1,j]}^R$ has finite index for all $j \in [1,k-1]$.*

Proof. Let $R \subseteq (\Sigma^*)^k$ be a rational relation. For the "only if" direction assume that $R = \bigcup_{i=1}^n L_{i,1} \times \cdots \times L_{i,k}$ is recognizable. We show that $\approx_{[1,j]}^R$ has finite index for any $j \in [1,k-1]$. Let $S_i := L_{i,1} \times \cdots \times L_{i,j}$ for all $i \in [1,n]$ and B_I as in Lemma 3.3.2 for all $I \subseteq [1,n]$. We have that

$$R = \bigcup_{I \subseteq [1,n] \colon B_I \neq \emptyset} B_I \times \big(\bigcup_{i \in I} L_{i,j+1} \times \cdots \times L_{i,k}\big).$$

Hence, since the B_I are pairwise disjoint, the tuples in each B_I are $\approx_{[1,j]}^R$-equivalent. Now, any tuple in $(\Sigma^*)^j$ is either $\approx_{[1,j]}^R$-equivalent to the tuples in one of the B_I or belongs to the equivalence class $(\Sigma^*)^j \setminus \bigcup_{I \subseteq [1,n]} B_I$. Thus, $\approx_{[1,j]}^R$ has finite index.

For the "if" direction assume that $\approx_{[1,j]}$ has finite index for all $j \in [1,k-1]$. For words $u_1,\dots,u_i \in \Sigma^*$ with $i \in [0,k-1]$ let $R|_{u_1,\dots,u_i} := \{(u_{i+1},\dots,u_k) \in (\Sigma^*)^{k-i} \mid (u_1,\dots,u_k) \in R\}$ be the restriction of R to $u_1,\dots,u_i$. Since $S_{u_1,\dots,u_i} := \{u_1\} \times \cdots \times \{u_i\} \times (\Sigma^*)^{k-i}$ is recognizable and R is rational, it follows from Proposition 2.3.25 that

$R|_{u_1,\dots,u_i} = S_{u_1,\dots,u_i} \cap R$ is rational. We show that for every $u_1, \dots, u_{j-1} \in \Sigma^*$ and $j \in [1, k-1]$ the restriction $R|_{u_1,\dots,u_{j-1}}$ is recognizable if $R|_{v_1,\dots,v_j}$ is recognizable for all $v_1, \dots, v_j \in \Sigma^*$. Then the claim follows by induction since $R|_{v_1,\dots,v_{k-1}}$ is trivially recognizable. Let $R' := R|_{u_1,\dots,u_{j-1}}$ for some arbitrary words $u_1, \dots, u_{j-1} \in \Sigma^*$ and $j \in [1, k-1]$. Since $\approx^R_{[1,j]}$ has finite index, also $\approx^{R'}_1$ as finite index. Let $\{w_1, \dots, w_n\}$ be a complete set of representatives of the $\approx^{R'}_1$-equivalence classes. Let $\{Y_1, \dots, Y_m\}$ be the coarsest partition finer than $\{R'|_{w_1}, \dots, R'|_{w_n}\}$ from Lemma 3.3.2. Since by assumption $R'|_{w_i}$ is recognizable for all $i \in [1, n]$, the closure properties of recognizable relations (Proposition 2.3.28) imply that also Y_ℓ is recognizable for all $\ell \in [1, m]$. Hence, by Proposition 2.3.25, the language $X_\ell := \{w \in \Sigma^* \mid R'|_w \cap Y_\ell \neq \emptyset\}$ is regular for all $\ell \in [1, m]$. Thus, $R' = \bigcup_{\ell=1}^m X_\ell \times Y_\ell$ is recognizable.

If R is tree-regular, we can take the same proof as above. For ω-regular R we consider in the "if" direction only restrictions of the form $R|_{u_1,\dots,u_j}$ where $u_1, \dots, u_j$ are ultimately periodic words. This suffices since we can always pick ultimately periodic representatives of the $\approx^{R'}_1$-equivalence classes: First observe that $\approx^{R'}_1$ is ω-regular due to the closure properties of ω-regular relations (Propositions 2.3.10 and 2.3.30). Now, pick an ultimately periodic word $w_1 \in \Sigma^\omega$ and since its equivalence class $[w_1] := \{w \in \Sigma^\omega \mid w_1 \approx^{R'}_1 w\}$ is ω-regular, Lemma 2.3.11 implies that we can pick an ultimately periodic $w_2 \in \Sigma^\omega \setminus [w_1]$. This can be continued until we picked $w_1, \dots, w_n$, where n is the (finite) index of $\approx^{R'}_1$. □

We say that $R \subseteq A^k$ is an *I-relation*, for some $I \subseteq [1, k]$, if there exists a relation $S \subseteq A^{|I|}$ such that $R = \{\boldsymbol{a} \odot_I \boldsymbol{b} \mid \boldsymbol{a} \in S, \boldsymbol{b} \in A^{k-|I|}\}$. In this case we say that R is an I-relation with respect to S. Intuitively, this means that the components from I are controlled by S and the remaining components can take arbitrary elements. Let P be a partition of $\{1, \dots, k\}$. Note that the class of I-relations is closed under Boolean operations. We say that $R \subseteq A^k$ *conforms* to P if R is a finite Boolean combination of relations $R_1, \dots, R_n$ where each R_i is an I-relation for some $I \in P$. In particular, R conforms to the *discrete partition* $\{\{1\}, \dots, \{k\}\}$ if and only if R is a finite (positive) Boolean combination of direct products $L_1 \times \cdots \times L_k$ of sets $L_i \subseteq A$.

Lemma 3.3.4. *Let $R \subseteq A^k$ be some relation. The equivalence relation $\approx^R_I$ has finite index if and only if R conforms to $\{I, [1, k] \setminus I\}$.*

Proof. For the "only if" direction assume that $\{\boldsymbol{a}_1, \dots, \boldsymbol{a}_n\}$ is a complete set of representatives of the equivalence classes of $\approx^R_I$. For every $i \in [1, n]$ we define the I-relation $X_i := \{\boldsymbol{a} \in A^{|I|} \mid \boldsymbol{a}_i \approx^R_I \boldsymbol{a}\}$ and the $([1, k] \setminus I)$-relation $Y_i := \{\boldsymbol{b} \in A^{k-|I|} \mid \boldsymbol{a}_i \odot_I \boldsymbol{b} \in R\}$. Then we have that $R = \bigcup_{i=1}^n X_i \cap Y_i$ as required.

For the "if" direction we can assume that R is of the form $R = \bigcup_{i=1}^n X_i \cap Y_i$ for some I-relations X_i with respect to S_i and $([1, k] \setminus I)$-relations Y_i. Moreover, by Lemma 3.3.2, we can assume that the X_i are pairwise disjoint, which implies that also the S_i are pairwise disjoint. Thus, the tuples in each S_i are $\approx^R_I$-equivalent. This means that every tuple $\boldsymbol{a} \in A^{|I|}$ is either $\approx^R_I$-equivalent to the tuples in one of the S_i or belongs to the equivalence class $A^{|I|} \setminus \left(\bigcup_{i=1}^n S_i\right)$. Therefore, $\approx^R_I$ has finite index. □

If R conforms to a partition P, then clearly R conforms to any partition P' that is coarser than P. The *coarsest refinement* $P_1 \sqcap P_2$ of two partitions P_1, P_2 of the same set is the set of all non-empty intersections $I_1 \cap I_2$ where $I_1 \in P_1$ and $I_2 \in P_2$. The following theorem was shown (in a slightly different setting) by Cosmadakis, Kuper, and Libkin [CKL01].

Theorem 3.3.5. *If a relation $R \subseteq A^k$ conforms to two partitions P_1, P_2 of $[1,k]$, then also to their coarsest refinement $P_1 \sqcap P_2$.*

The partition of $[1,k]$ *generated* by subsets $I_1, \dots, I_n \subseteq [1,k]$ is the coarsest refinement $P_1 \sqcap \cdots \sqcap P_n$ of the partitions $P_j = \{I_j, [1,k] \setminus I_j\}$. For example, the discrete partition of $[1,k]$ is clearly generated by the singleton sets $\{1\}, \dots, \{k-1\}$. It is also generated by all intervals $[1,j]$ for $j \in [1,k-1]$.

Lemma 3.3.6. *Let P be a partition of $[1,k]$ generated by $I_1, \dots, I_n \subseteq [1,k]$. Then $R \subseteq A^k$ conforms to P if and only if $\approx^R_{I_j}$ has finite index for all $j \in [1,n]$.*

Proof. By Theorem 3.3.5, R conforms to P if and only if R conforms to $\{I_j, [1,k] \setminus I_j\}$ for all $j \in [1,n]$. By Lemma 3.3.4, this is equivalent to finite index of $\approx^R_{I_j}$ for all $j \in [1,n]$. □

Thus, by choosing the discrete partition of $[1,k]$ for P in Lemma 3.3.6, we obtain for any $I_1, \dots, I_n \subseteq [1,k]$ generating the discrete partition of $[1,k]$ that $\approx^R_{I_j}$ has finite index for all $j \in [1,n]$ if and only if $\approx^R_{[1,j]}$ has finite index for all $j \in [1,k-1]$. Using Proposition 3.3.3, this gives rise to the following proposition.

Proposition 3.3.7. *Let $R \subseteq A^k$ be a rational, ω-regular, or tree-regular relation and $I_1, \dots, I_n \subseteq [1,k]$ be subsets that generate the discrete partition of $[1,k]$. Then R is recognizable (resp. ω-recognizable or tree-recognizable) if and only if $\approx^R_{I_j}$ has finite index for all $j \in [1,n]$.*

In particular, Proposition 3.3.7 implies that recognizability of $R \subseteq A^k$ is equivalent to finiteness of the indexes of $\approx^R_j$ for all $j \in [1,k-1]$.

3.3.2 Monadic Decomposability

A formula $\varphi(x_1, \dots, x_k)$ over a structure $\mathfrak{A}$ is *monadically decomposable* if it is $\mathfrak{A}$-equivalent to a Boolean combination of *monadic formulas* over $\mathfrak{A}$, i.e. formulas with only a single free variable [Vea+17].

Example 3.3.8. Consider the formula $\varphi(x,y) := x+y \geq 2 \wedge x \geq 0 \wedge y \geq 0$ in Presburger arithmetic. Since φ contains an atom with more than one free variable, it is not a monadic decomposition. However, φ is monadically decomposable since it is equivalent to

$$(x \geq 2 \wedge y \geq 0) \vee (x \geq 1 \wedge y \geq 1) \vee (x \geq 0 \wedge y \geq 2),$$

where each atom is a monadic formula.

On the other hand, it is not hard to see that the formula $x = y$ is not monadically decomposable over any structure with an infinite domain.

In general, the problem of monadic decomposability, i.e. given a formula φ over some fixed structure $\mathfrak{A}$, check whether φ is monadically decomposable, is undecidable. However, Libkin [Lib03] provides sufficient conditions on the structure $\mathfrak{A}$ under which the problem becomes decidable. In particular, it is shown that under these assumptions, monadic decomposability can be characterized via finite-index equivalence relations.

Let $\mathfrak{A}$ be a structure with domain A. We say that $\mathfrak{A}$ has *effective test for algebraicity* if for every monadic formula $\varphi(x)$ over $\mathfrak{A}$ it is decidable whether φ is algebraic, i.e. whether $[\![\varphi]\!]_{\mathfrak{A}}$ is finite. A function $f\colon A^k \to A^\ell$ is *definable* over $\mathfrak{A}$ if there exists a formula $\varphi(\boldsymbol{x}, \boldsymbol{y})$ with $|\boldsymbol{x}| = k$ and $|\boldsymbol{y}| = \ell$ over $\mathfrak{A}$ such that $f(\boldsymbol{a}) = \boldsymbol{b}$ if and only if $\mathfrak{A} \models \varphi(\boldsymbol{a}, \boldsymbol{b})$ for all $\boldsymbol{a} \in A^k$ and $\boldsymbol{b} \in A^\ell$. Finally, we say that $\mathfrak{A}$ has *definable invariant Skolem functions* if for every formula $\varphi(\boldsymbol{x}, \boldsymbol{y})$ over $\mathfrak{A}$ there exists a definable function $f_\varphi\colon A^{|\boldsymbol{x}|} \to A^{|\boldsymbol{y}|}$ over $\mathfrak{A}$ such that $\mathfrak{A} \models \forall \boldsymbol{x}\colon (\exists \boldsymbol{y}\colon \varphi(\boldsymbol{x}, \boldsymbol{y})) \to \varphi(\boldsymbol{x}, f_\varphi(\boldsymbol{x}))$ and if $[\![\varphi(\boldsymbol{a}_1, \boldsymbol{y})]\!]_{\mathfrak{A}} = [\![\varphi(\boldsymbol{a}_2, \boldsymbol{y})]\!]_{\mathfrak{A}}$, then $f_\varphi(\boldsymbol{a}_1) = f_\varphi(\boldsymbol{a}_2)$ for all $\boldsymbol{a}_1, \boldsymbol{a}_2 \in A^{|\boldsymbol{x}|}$. Intuitively, $f_\varphi(\boldsymbol{x})$ selects an element in the image of $\boldsymbol{x}$ under φ under the assumption that the image is non-empty and this element is unique in the sense that if $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ have the same image under φ, then f_φ selects the same element for both $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$. The following theorem is shown in [Lib03].

Theorem 3.3.9. *If $\mathfrak{A}$ is a structure such that*

(a) $\mathrm{Th}(\mathfrak{A})$ *is decidable,*

(b) $\mathfrak{A}$ has effective test for algebraicity, and

(c) $\mathfrak{A}$ has definable invariant Skolem functions,

then monadic decomposability is decidable over $\mathfrak{A}$. In particular, if a structure $\mathfrak{A}$ satisfies (c), then a formula $\varphi(x_1, \ldots, x_k)$ over $\mathfrak{A}$ is monadically decomposable if and only if $\approx^{[\![\varphi]\!]_{\mathfrak{A}}}_{[1,j]}$ has finite index for all $j \in [1, k-1]$.

Examples of structure that satisfy the conditions from Theorem 3.3.9 are Presburger arithmetic, linear real arithmetic, and the real field $\langle \mathbb{R}; <, +, \cdot, 0, 1 \rangle$. Thus, monadic decomposability is decidable for all of these examples. Note that since Presburger arithmetic and linear real arithmetic are (ω-)automatic structures, Theorem 3.3.9 and Proposition 3.3.7 imply that over those structures a formula φ is monadically decomposable if and only if $[\![\varphi]\!]_{\mathfrak{p}}$ is recognizable, where $\mathfrak{p}$ is an (ω-)automatic presentation.

Veanes et al. [Vea+17] considered the problem of monadic decomposability restricted to the quantifier-free fragment of fist-order logic. We say that a quantifier-free formula φ over some structure $\mathfrak{A}$ is monadically decomposable in the quantifier-free fragment of $\mathfrak{A}$ if and only if φ is $\mathfrak{A}$-equivalent to a Boolean combination of quantifier-free monadic formulas over $\mathfrak{A}$. Veanes et al. [Vea+17] prove a theorem similar to Theorem 3.3.9 (although they only give a semi-decision procedure for binary formulas). We adapt the techniques from [Vea+17] to obtain a sufficient condition for the characterization of monadic decomposability in the quantifier-free fragment via finite-index equivalence relations.

We say that a tuple $(a_1, \ldots, a_n) \in A^n$ is definable by ground terms over a structure $\mathfrak{A} = (A, I)$ if the element a_i is definable by ground terms over $\mathfrak{A}$ for all $i \in [1, n]$.

Proposition 3.3.10. *Let $\mathfrak{A} = (A, I)$ be a structure such that for every finite-index equivalence relation $E \subseteq A^n \times A^n$ definable over $\mathfrak{A}$ each equivalence class contains a representative that is definable by ground terms over $\mathfrak{A}$. Then a quantifier-free formula $\varphi(x_1, \ldots, x_k)$ is monadically decomposable in the quantifier-free fragment of $\mathfrak{A}$ if and only if $\approx_j^{[\![\varphi]\!]_\mathfrak{A}}$ has finite index for all $j \in [1, k-1]$.*

Proof. For the "only if" direction we can assume that φ is in DNF

$$\varphi(\boldsymbol{x}) = \bigvee_{i=1}^{n} \psi_{i,1}(x_1) \wedge \cdots \wedge \psi_{i,k}(x_k).$$

For all $i \in [1, n]$ and $j \in [1, k]$ let $S_{i,j} := A^{j-1} \times [\![\psi_{i,j}]\!] \times A^{k-j}$. Then $[\![\varphi]\!] = \bigcup_{i=1}^{n} S_{i,1} \cap \cdots \cap S_{i,k}$ conforms to the discrete partition of $[1, k]$, which by Lemma 3.3.6 implies that $\approx_j^{[\![\varphi]\!]}$ has finite index for all $j \in [1, k-1]$.

For the "if" direction assume that $\approx_j^{[\![\varphi]\!]}$ has finite index for all $j \in [1, k-1]$ (and therefore for all $j \in [1, k]$), which by Lemma 3.3.6 implies that also $\approx_{[1,k]\setminus\{j\}}^{[\![\varphi]\!]}$ has finite index for all $j \in [1, k]$. By assumption, for every $j \in [1, k]$ there exist complete sets V_j and W_j of representatives of the equivalence classes of $\approx_j^{[\![\varphi]\!]}$ and $\approx_{[1,k]\setminus\{j\}}^{[\![\varphi]\!]}$, respectively, such that every element in V_j and every tuple in W_j is definable by ground terms over $\mathfrak{A}$. Thus, we can define a quantifier-free formula for $\approx_j^{[\![\varphi]\!]}$ as

$$x \approx_j y := \bigwedge_{\boldsymbol{b} \in W_j} \varphi(x \odot_j \boldsymbol{b}) \leftrightarrow \varphi(y \odot_j \boldsymbol{b})$$

for all $j \in [1, k]$. Now, φ is equivalent to the following finite Boolean combination of quantifier-free monadic formulas:

$$\bigvee_{\boldsymbol{a} \in (V_1 \times \cdots \times V_k) \cap [\![\varphi]\!]} \bigwedge_{j=1}^{k} x_j \approx_j a_j$$

□

Example 3.3.11. Clearly, Presburger arithmetic satisfies the condition of Proposition 3.3.10 since, as discussed in Section 2.4.2, we assume that we have access to all constants in $\mathbb{Z}$. Using the fact that LRA and LIRA are ω-automatic structures, one can show that they also satisfy the condition of Proposition 3.3.10: By Lemma 2.3.31, every equivalence class of an ω-regular equivalence relation with finite index contains a tuple of ultimately periodic words, which in case of the standard presentation for LRA and LIRA (see Example 2.4.9) are always rational numbers and therefore definable by ground terms (since we assume that terms can use rational constants). Thus, also in case of LIRA, the correspondence between monadic decomposability of a formula φ and recognizability of the relation $[\![\varphi]\!]_\mathfrak{p}$ holds.

4 Ramsey Quantifiers in Automatic Structures

Parts of this chapter were published by the author in [Ber+22].

As already motivated in the introduction, Ramsey quantifiers in automatic structures, on the one hand, provide a general framework for liveness verification in regular model checking and, on the other hand, have applications to recognizability. Recall that regular model checking describes the problem of checking whether a transition system with (tree-)regular transition relation satisfies some specification given in some logical language. More specifically, we consider the case where even the reachability relation $\to^+$, i.e. the transitive closure of the transition relation $\to$, is (tree-)regular and provided as an input in form of an (tree) automaton. This assumption can, for example, be made for pushdown systems [Esp+00], ground tree rewriting systems [Löd06], and PA-processes [LS05], where (tree) automata for the reachability relations can even be computed in polynomial time. Then recurrent reachability of a set L from an initial configuration a can be reduced to checking satisfiability of

$$\exists^{\mathrm{ram}} x, y \colon a \to^+ x \wedge x \to^+ y \wedge x \in L \qquad \vee \qquad \exists x \colon a \to^+ x \wedge x \to^+ x \wedge x \in L.$$

Here, the disjunct on the left checks for an infinite run of pairwise distinct configurations that is reachable from a and visits configurations in L infinitely often. The right disjunct checks the existence of a lasso, i.e. a cycle containing a configuration in L that is reachable from a. Note that the right disjunct can easily be checked due to the closure properties of (tree-)regular languages/relations. To check satisfiability of the left disjunct, we have to solve the infinite clique problem for (tree-)regular relations. Since the reachability relation is transitive, for this application it suffice to solve the infinite clique problem restricted to transitive relations. We show that for regular relations the complexity of the infinite clique problem is independent from transitivity, while for tree-regular relations transitivity can result in an exponential speedup (for certain models of tree automata).

A similar situation arises in the application to recognizability. Here, the Ramsey quantifier is used to check whether certain (tree-)regular equivalence relations have finite index. To this end, we construct (tree) automata for the complements of these equivalence relations and check for an infinite clique. Thus, in this case it suffices to solve the infinite clique problem for co-transitive (tree-)regular relations. We show that while over words co-transitivity does not effect the complexity, over trees co-transitivity again exponentially decreases the complexity of the infinite clique problem for certain models of tree automata.

More generally, to evaluate the Ramsey quantifier over regular relations, we first observe that every infinite clique in a regular relation contains an infinite subclique wit-

	regular relations	tree-regular relations
Evaluation of the Ramsey quantifier	logspace	polynomial time for transitive relations or D↑TA
Recurrent reachability & infinite clique	NL-complete*	EXP-complete for NTA, D↓TA P-complete for transitive* or co-transitive relations or D↑TA
Recurrent reachability with generalized Büchi condition	PSPACE-complete	EXP-complete
Recognizability	NL-complete for DFA* PSPACE-complete for NFA*	P-complete for D↑TA, D↓TA EXP-complete for NTA

Table 4.1: An overview of the complexity results obtained in this chapter. Those marked with * were known, but we provide simpler proofs. The other results are new.

nessed by accepting runs that form a so-called *comb of combs* structure. This structure makes it possible for an alternating Büchi automaton to verify that the encoding of an infinite sequence forms a clique. To obtain a polynomially-sized Büchi automaton, we further observe that the accepting runs can be "merged", yielding an even easier structure. In the case of tree-regular relations, the strategy is similar. The first observation, that the witnessing runs form a comb of combs structure, can be extended to trees. The merging, however, is impossible over general tree-regular relations given as nondeterministic tree automata since we prove the infinite clique problem to be EXP-hard in this case. The EXP upper bound follows from the fact that an alternating Büchi tree automaton can simulate the comb of combs. However, for deterministic bottom-up tree automata the complexity is much lower. Due to bottom-up determinism, merging becomes possible again, which yields existence of a polynomially-sized Büchi automaton that accepts encodings of infinite cliques.

4.1 Main Results

Before we dive into the proofs, let us provide a detailed summary of the main results of this chapter. An overview can be found in Table 4.1. Unless otherwise specified, the completeness results mentioned in this section (and Table 4.1) hold for NFAs and DFAs in the word case and NTAs, D↑TAs, and D↓TAs in the tree case.

Ramsey quantifiers in automatic structures It follows from [Bar+19] that the infinite clique problem over word-regular relations is NL-complete. In Section 4.2 we provide a much simpler proof by considering a slightly more general setting. Instead of the infinite clique problem we consider the *evaluation* of the Ramsey quantifier on a $(2d + k)$-ary (tree-)regular relation $R \subseteq A^{2d+k}$, i.e. compute an automaton for

$[\![\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!]$. For this recall that R implicitly defines a structure $\mathfrak{A} := \langle A; R \rangle$ and $[\![\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!] = \{\boldsymbol{c} \in A^k \mid \mathfrak{A} \models \exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c})\}$.

Theorem 4.2.1. *Given a regular relation $R \subseteq (\Sigma^*)^{2d+k}$ by an NFA, one can construct in logspace an NFA for the relation $[\![\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!]$, where $|\boldsymbol{x}| = |\boldsymbol{y}| = d$ and $|\boldsymbol{z}| = k$.*

In particular, Theorem 4.2.1 implies that the infinite clique problem over regular relations is in NL.

Corollary 4.2.2. *The infinite clique problem over regular relations $R \subseteq (\Sigma^*)^{2d}$ is* NL*-complete.*

Theorem 4.2.1 shows that Ramsey quantifiers can be effectively evaluated in automatic structures and can be reformulated as follows.

Corollary 4.2.3. *Given an automatic structure $\mathfrak{A}$ by a presentation $\mathfrak{p}$ and a formula $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ over $\mathfrak{A}$ with $|\boldsymbol{x}| = |\boldsymbol{y}|$, one can effectively compute an NFA for the relation $[\![\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!]_{\mathfrak{p}}$. Moreover, if an NFA for $[\![\varphi]\!]_{\mathfrak{p}}$ is given, the construction requires only logspace.*

Ramsey quantifiers in tree-automatic structures In Section 4.3 we show that the complexity of the infinite clique problem increases from NL to EXP when considered over tree-regular relations given by NTAs or D↓TAs and to P for D↑TAs.

Theorem 4.3.1. *The infinite clique problem over tree-regular relations $R \subseteq (\mathcal{T}_\Sigma)^{2d}$ is* EXP*-complete if R is given as NTA or D↓TA and* P*-complete if R is given as D↑TA.*

For the exponential lower bound we present a reduction from intersection non-emptiness for NTAs and D↓TAs. This is surprising because an analogous reduction in the word case does not exist: This would yield a PSPACE lower bound for the infinite clique problem over words but the latter belongs to NL. The P lower bound for D↑TAs follows from Proposition 3.2.1 since recurrent reachability is P-hard if the relation is given by a D↑TA (Corollary 4.4.1).

The exponential upper bound for NTAs was already stated by To [To10] in the context of recurrent reachability, which is logspace equivalent to the infinite clique problem (Proposition 3.2.1). We show that it is even solvable in polynomial time if the relation is given by a D↑TA by proving the tree analogue of Theorem 4.2.1. Using Proposition 2.3.20, this implies that the exponential upper bound also holds for ATAs, which allows us to apply it to recurrent reachability with *generalized Büchi condition* (see Section 4.4).

Theorem 4.3.2. *Given a D↑TA (resp. ATA) for a tree-regular relation $R \subseteq (\mathcal{T}_\Sigma)^{2d+k}$, one can construct in polynomial (resp. exponential) time an NTA for the relation $[\![\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!]$, where $|\boldsymbol{x}| = |\boldsymbol{y}| = d$ and $|\boldsymbol{z}| = k$.*

As in the word case, we can reformulate Theorem 4.3.2 for tree-automatic structures.

Corollary 4.3.3. *Given a tree-automatic structure $\mathfrak{A}$ by a presentation $\mathfrak{p}$ and a formula $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ over $\mathfrak{A}$ with $|\boldsymbol{x}| = |\boldsymbol{y}|$, one can effectively compute an NTA for the relation $[\![\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!]_{\mathfrak{p}}$. Moreover, if a D↑TA for $[\![\varphi]\!]_{\mathfrak{p}}$ is given, the construction requires only polynomial time.*

If we make further assumptions on the relation R, we obtain a better complexity for NTAs. We say that a $(2d + k)$-ary relation R over A is *transitive* if the binary relation $\{(\boldsymbol{a}, \boldsymbol{b}) \in A^d \times A^d \mid (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \in R\}$ is transitive for all $\boldsymbol{c} \in A^k$. Recurrent reachability for transitive tree-regular relations was already shown to be decidable in polynomial time by To and Libkin [TL08].

Theorem 4.3.9. *Given an NTA for a transitive tree-regular relation $R \subseteq (\mathcal{T}_\Sigma)^{2d+k}$, one can construct in polynomial time an NTA for the relation $[\![\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!]$, where $|\boldsymbol{x}| = |\boldsymbol{y}| = d$ and $|\boldsymbol{z}| = k$. In particular, the infinite clique problem over transitive tree-regular relations $R \subseteq (\mathcal{T}_\Sigma)^{2d}$ is* P*-complete.*

We show that the complexity of the infinite clique problem is the same if instead of transitive relations we consider complements thereof. A relation R is *co-transitive* if its complement $\overline{R}$ is a transitive relation.

Theorem 4.3.13. *The infinite clique problem over co-transitive tree-regular relations $R \subseteq (\mathcal{T}_\Sigma)^{2d}$ is* P*-complete.*

Theorem 4.3.13 enables us to apply our results on infinite cliques to the recognizability problem of tree-regular relations while obtaining precise complexity (see Section 4.4).

In Section 4.5 we show by a reduction that the Ramsey quantifier can be evaluated over unranked tree-regular relations with the same complexity as in the ranked case.

Recurrent reachability Recall that the problem of recurrent reachability over (tree-)regular relations is to check whether in a given (tree-)regular relation $R \subseteq A^{2d}$ there exists a transitive path, i.e. an infinite sequence $(\boldsymbol{a}_i)_{i \geq 1}$ in A^d with $(\boldsymbol{a}_i, \boldsymbol{a}_j) \in R$ for all $1 \leq i < j$, that visits a given (tree-)regular relation $L \subseteq A^d$ infinitely often. In Section 4.4.1 we use Proposition 3.2.1 to show that we can transfer the upper bounds in Corollary 4.2.2 and Theorems 4.3.1 and 4.3.9 for the infinite clique problem directly to recurrent reachability.

Corollary 4.4.1. *Recurrent reachability is* NL*-complete over regular relations. It is* EXP*-complete over tree-regular relations given by NTAs or D↓TAs and* P*-complete if the tree-regular relations are transitive or given by D↑TAs.*

If instead of a single L multiple given relations $L_1, \ldots, L_k$ must be visited infinitely often, we call the problem recurrent reachability with generalized Büchi condition.

Theorem 4.4.3. *Recurrent reachability with generalized Büchi condition is* PSPACE*-complete over regular relations and* EXP*-complete over tree-regular relations.*

The lower bounds already hold if R is transitive and $L_1, \ldots, L_k$ are languages. We show that we can even compute an automaton for the set of all initial vectors of transitive paths in R that satisfy the generalized Büchi condition.

Recognizability Recall that a relation $R \subseteq A^k$ is *(tree-)recognizable* if it is of the form $R = \bigcup_{i=1}^{n} A_{i,1} \times \cdots \times A_{i,k}$ for some (tree-)regular languages $A_{i,j}$. The traditional approach to deciding recognizability [GS66b; Lib03; CCG06; LS19b; Bar+19] is to associate with R the equivalence relations $\approx_{[1,j]}$ for all $j \in [1, k-1]$ as defined in Section 3.3. As seen in Proposition 3.3.7, R is (tree-)recognizable if and only if each $\approx_{[1,j]}$ has finite index. An equivalence relation has infinite index if and only if there exist infinitely many elements that are pairwise in different equivalence classes, which is witnessed by an infinite clique in the complement relation. Therefore, (tree-)recognizability amounts to checking that $\approx_{[1,j]}$'s complement $\not\approx_{[1,j]}$ does not have an infinite clique for any j. If R is given by a DFA (resp. NFA), then one can construct an NFA for each $\approx_{[1,j]}$ in logspace (resp. polynomial space) and thus Theorem 4.2.1 yields a tight upper bound (which was already established in [Bar+19]).

Corollary 4.4.9. *Given a regular relation $R \subseteq (\Sigma^*)^k$ by a DFA (resp. NFA), it is* NL*-complete (resp.* PSPACE*-complete) to decide whether R is recognizable.*

While this approach yields optimal complexity for words, this is, unexpectedly, not the case for trees. For a tree-regular relation given as D↓TA or D↑TA (resp. NTA) one can also construct an NTA for each $\not\approx_{[1,j]}$ in polynomial (resp. exponential) time. Then applying Theorem 4.3.2 would yield an EXP (resp. 2-EXP) algorithm. However, perhaps surprisingly, in Section 4.4.2 we show that tree-recognizability for tree-regular relations has much lower complexity:

Corollary 4.4.10. *Given a tree-regular relation $R \subseteq (\mathcal{T}_\Sigma)^k$ by a D↑TA or D↓TA (resp. NTA), it is* P*-complete (resp.* EXP*-complete) to decide whether R is tree-recognizable.*

To get the P (resp. EXP) algorithm, we exploit the co-transitivity of each $\not\approx_{[1,j]}$ and apply Theorem 4.3.13 instead of Theorem 4.3.2. This shows the importance of the co-transitivity notion: In the word case, recognizability requires only the generic clique detection, but the tree case is more nuanced. Here, we need one algorithm for the general case and a specialized algorithm for co-transitive relations.

4.2 Word-Automatic Structures

We first consider the evaluation of the Ramsey quantifier over word-regular relations. As the main result we show the following:

Theorem 4.2.1. *Given a regular relation $R \subseteq (\Sigma^*)^{2d+k}$ by an NFA, one can construct in logspace an NFA for the relation $[\![\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!]$, where $|\boldsymbol{x}| = |\boldsymbol{y}| = d$ and $|\boldsymbol{z}| = k$.*

An immediate consequence of Theorem 4.2.1 is that the infinite clique problem over regular relations is in NL.

Corollary 4.2.2. *The infinite clique problem over regular relations $R \subseteq (\Sigma^*)^{2d}$ is* NL*-complete.*

Proof. For the upper bound let R be given by an NFA. By Theorem 4.2.1, we can construct an NFA for $[\![\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R(\boldsymbol{x}, \boldsymbol{y})]\!]$ in logspace and then check non-emptiness in NL using Proposition 2.3.5.

The lower bound already holds for binary relations given by a DFA and follows from Proposition 3.2.1 since recurrent reachability is NL-complete by Corollary 4.4.1. □

From the perspective of automatic structures, Theorem 4.2.1 can be rephrased as follows:

Corollary 4.2.3. *Given an automatic structure $\mathfrak{A}$ by a presentation $\mathfrak{p}$ and a formula $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ over $\mathfrak{A}$ with $|\boldsymbol{x}| = |\boldsymbol{y}|$, one can effectively compute an NFA for the relation $[\![\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!]_{\mathfrak{p}}$. Moreover, if an NFA for $[\![\varphi]\!]_{\mathfrak{p}}$ is given, the construction requires only logspace.*

Corollary 4.2.3 implies that over automatic structures one can effectively construct an automaton for the relation defined by a formula in first-order logic enriched with the Ramsey quantifier. Thus, since now satisfiability reduces to checking non-emptiness of the language accepted by the resulting automaton, we obtain the following extension of Proposition 2.4.8, which was already shown by Rubin [Rub08].

Corollary 4.2.4. *The enriched theory* $\mathrm{Th}^{\exists^{\text{ram}}}(\mathfrak{A})$ *of every automatic structure $\mathfrak{A}$ is decidable.*

To prove Theorem 4.2.1, we will in the following first assume that $R \subseteq \Sigma^* \times \Sigma^*$ is a binary relation, i.e. we assume that $d = 1$ and $k = 0$. At the end of this section, when we complete the proof of Theorem 4.2.1, we show that the general case can be reduced to the binary case.

Word combs The first step is to observe that, when looking for infinite cliques in R, one can restrict to combs: An infinite sequence $\boldsymbol{v}$ of words is called a *comb* if there exist infinite sequences $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ of words with $v_i = \beta_1 \dots \beta_{i-1}\alpha_i$ and $1 \leq |\alpha_i| \leq |\beta_i|$ for all $i \geq 1$. The pair $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is called a *generator* of $\boldsymbol{v}$. We remark that the choice of the generator is not unique. Any infinite subsequence of a comb is again a comb. In fact, the following lemma is well-known (see e.g. [KL10]).

Lemma 4.2.5. *Any infinite sequence $\boldsymbol{w}$ of pairwise distinct words w_i over a finite alphabet Σ contains a comb as a subsequence.*

Proof. It suffices to show that there exists a comb over the set $W := \{w_i \mid i \geq 1\}$. Indeed, since $(\mathbb{N}, \leq)$ is a well-quasi-ordering, any sequence $(w_{i_j})_{j \geq 1}$ of pairwise distinct words $w_{i_j} \in W$ contains a subsequence $(w_{i'_j})_{j \geq 1}$ such that $i'_j < i'_k$ for all $j < k$, i.e. a subsequence of $\boldsymbol{w}$. Consider the infinite tree over Σ^* with the edges (w, wa) for all $w \in \Sigma^*$ and $a \in \Sigma$, which is $|\Sigma|$-branching. The set of all prefixes of words in W forms an infinite subtree, which contains an infinite path π by Kőnig's Lemma. Let $u_0 := \varepsilon$ be the root. Then for all $i \geq 1$ let $v_i \in W$ be any word which contains u_{i-1} as a proper prefix and let u_i be the unique node on π with $|u_i| = |v_i|$. Then we can write $u_i = \beta_1 \dots \beta_i$ and $v_i = \beta_1 \dots \beta_{i-1}\alpha_i$ for some generator $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ with $1 \leq |\alpha_i| = |\beta_i|$. □

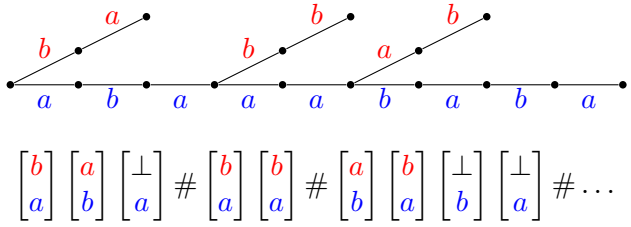


Figure 4.1: An example word comb $ba, ababb, abaaaab, \dots$ and its encoding as an infinite word.

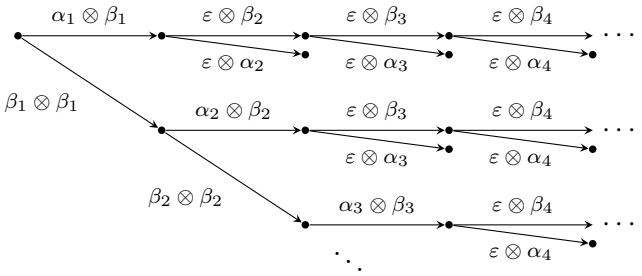


Figure 4.2: If $\boldsymbol{v}$ is a comb of the form $v_i = \beta_1 \dots \beta_{i-1}\alpha_i$, then the convolutions $v_i \otimes v_j$ for all $i < j$ form a *comb of combs*.

In contrast to arbitrary infinite sequences of words, combs can be encoded naturally by infinite words. If $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is a generator, we call the infinite word

$$\operatorname{enc}(\boldsymbol{\alpha}, \boldsymbol{\beta}) := \begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix} \# \begin{bmatrix} \alpha_2 \\ \beta_2 \end{bmatrix} \# \dots \in \left((\Sigma_\bot \times \Sigma) \cup \{\#\}\right)^\omega$$

the *encoding of* $(\boldsymbol{\alpha}, \boldsymbol{\beta})$, or also an encoding of $\boldsymbol{v}$, where $\# \notin \Sigma$ is a fresh delimiter symbol. An example of a comb together with its encoding as an infinite word is depicted in Figure 4.1. Clearly, the set of valid comb encodings is ω-regular and given an alphabet, an NBA for it can be constructed in logspace.

Comb of combs The next goal would be to construct a Büchi automaton which reads an encoding of a comb $\boldsymbol{v}$ and verifies that $v_i \otimes v_j$ has an accepting run $\rho(v_i, v_j)$ for all $1 \leq i < j$. In general, this is challenging since it is not clear how a finite automaton can keep track of infinitely many runs (let alone, an automaton of polynomial size). Instead, we will show that every infinite clique contains an infinite subclique whose accepting runs can be arranged in a dag of constant width and can therefore be recognized by a polynomially-sized Büchi automaton.

Consider a comb $\boldsymbol{v}$ with generator $(\boldsymbol{\alpha}, \boldsymbol{\beta})$. First observe that the convolutions $v_i \otimes v_j$ can be written as

$$\begin{bmatrix} v_i \\ v_j \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_1 \end{bmatrix} \dots \begin{bmatrix} \beta_{i-1} \\ \beta_{i-1} \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} \begin{bmatrix} \varepsilon \\ \beta_{i+1} \end{bmatrix} \dots \begin{bmatrix} \varepsilon \\ \beta_{j-1} \end{bmatrix} \begin{bmatrix} \varepsilon \\ \alpha_j \end{bmatrix} \tag{4.1}$$

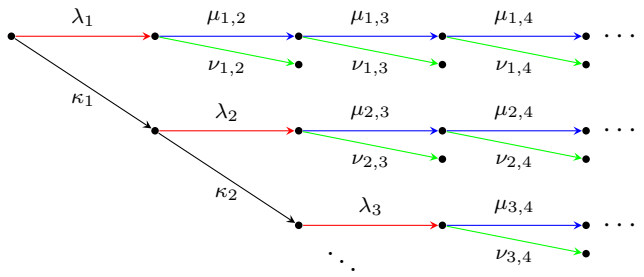


Figure 4.3: One can always find an infinite comb clique $\boldsymbol{v}$ with accepting runs that can be decomposed in the form $\rho(v_i, v_j) = \kappa_1 \dots \kappa_{i-1}\lambda_i\mu_{i,i+1} \dots \mu_{i,j-1}\nu_{i,j}$.

and can hence be arranged in a trie displayed in Figure 4.2, that we call *comb of combs*. The next insight is that we can ensure that the accepting runs $\rho(v_i, v_j)$ on the convolutions $v_i \otimes v_j$ match this comb of combs structure after replacing $\boldsymbol{v}$ with an infinite subsequence. Roughly speaking, the runs look as if the automaton for R would be *deterministic*. For example, all runs $\rho(v_1, v_j)$ for $j > 1$ share a common prefix which is a run on $\alpha_1 \otimes \beta_1$ and all runs $\rho(v_i, v_j)$ for $1 < i < j$ share a common prefix which is a run on $\beta_1 \otimes \beta_1$.

We define a *decomposition* of a word $w \in \Sigma^*$ as $w = u_1 \dots u_n$ where $u_i \in \Sigma^*$. The decomposition in Equation (4.1) is called the $(\boldsymbol{\alpha}, \boldsymbol{\beta})$-decomposition of $v_i \otimes v_j$. We say that a decomposition $\rho = \rho_1 \dots \rho_n$ of a run of an NFA is *compatible* with a decomposition $w = u_1 \dots u_n$ of a word if ρ_i is a run on u_i for all $i \in [1, n]$.

We say that a generator $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of a comb $\boldsymbol{v}$ is *coarser* than a generator $(\boldsymbol{\gamma}, \boldsymbol{\delta})$ of a comb $\boldsymbol{w}$ if there exist indices $k_1 < k_2 < \dots$ such that $v_i = w_{k_i}$ and $\beta_1 \dots \beta_i = \delta_1 \dots \delta_{k_i}$ for all $i \geq 1$. In this case we also say that $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is the coarsening of $(\boldsymbol{\gamma}, \boldsymbol{\delta})$ defined by the subsequence $\boldsymbol{v}$ of $\boldsymbol{w}$.

Lemma 4.2.6. *If $\boldsymbol{w}$ is an infinite clique in a regular relation $R \subseteq \Sigma^* \times \Sigma^*$ given as an NFA $\mathcal{A}$, then there exist a generator $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of a subsequence $\boldsymbol{v}$ of $\boldsymbol{w}$, accepting runs $\rho(v_i, v_j)$ of $\mathcal{A}$ on $v_i \otimes v_j$, and runs $\kappa_i, \lambda_i, \mu_{i,j}, \nu_{i,j}$ such that*

$$\rho(v_i, v_j) = \kappa_1 \dots \kappa_{i-1}\lambda_i\mu_{i,i+1} \dots \mu_{i,j-1}\nu_{i,j}$$

is a decomposition compatible with the $(\boldsymbol{\alpha}, \boldsymbol{\beta})$-decomposition of $v_i \otimes v_j$ for all $i < j$.

Proof. Let $\boldsymbol{w}$ be an infinite clique in R and $\rho(w_i, w_j)$ be an accepting run of $\mathcal{A}$ on $w_i \otimes w_j$ for all $1 \leq i < j$. By Lemma 4.2.5, we can assume that $\boldsymbol{w}$ is a comb. We establish the run structure as illustrated in Figure 4.3 column-wise.

Assume we already defined a subcomb $\boldsymbol{v}$ of $\boldsymbol{w}$ with generator $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ and runs $(\kappa_i)_{i<n}$, $(\lambda_i)_{i<n}$, $(\mu_{i,j})_{i<j<n}$, and $(\nu_{i,j})_{i<j<n}$ for some $n \geq 1$ such that

$$\begin{aligned}\rho(v_i, v_j) &= \kappa_1 \dots \kappa_{i-1}\lambda_i\mu_{i,i+1} \dots \mu_{i,n-1}\tau(v_i, v_j)\\ \rho(v_{i'}, v_{j'}) &= \kappa_1 \dots \kappa_{n-1}\sigma(v_{i'}, v_{j'})\end{aligned}$$

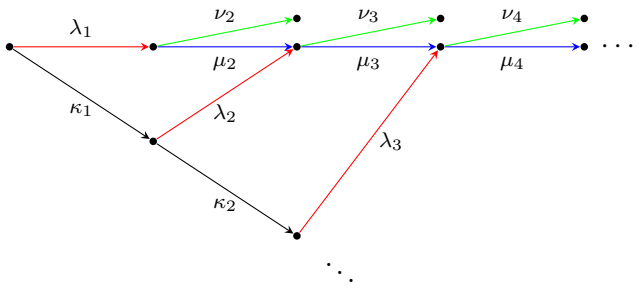


Figure 4.4: One can always find an infinite comb clique $\boldsymbol{v}$ such that the accepting runs in Figure 4.3 can be *merged*, which yields decompositions of the form $\rho(v_i, v_j) = \kappa_1 \dots \kappa_{i-1}\lambda_i\mu_{i+1} \dots \mu_{j-1}\nu_j$.

for runs $\tau(v_i, v_j)$, $\sigma(v_{i'}, v_{j'})$ for all $1 \leq i < n \leq j$ and $n \leq i' < j'$. For all $1 \leq i < n$ we just set $\nu_{i,n} := \tau(v_i, v_n)$.

We now define $\mu_{i,n}$ successively for each $1 \leq i < n$. In step i we apply the pigeonhole principle to get an infinite subsequence $\boldsymbol{u}$ of $\boldsymbol{v}$ starting with $v_1, \dots, v_n$ such that all runs $\tau(u_i, u_j)$ for $j > n$ have a common prefix $\mu_{i,n}$ which is a run on $\varepsilon \otimes \beta_n$. At the end of step i, we replace $\boldsymbol{v}$ with $\boldsymbol{u}$ and we replace $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ with the coarsening defined by $\boldsymbol{u}$.

Next we define λ_n. By the pigeonhole principle, there exists an infinite subsequence $\boldsymbol{u}$ of $\boldsymbol{v}$ starting with $v_1, \dots, v_n$ such that all runs $\sigma(u_n, u_j)$ for $j > n$ have a common prefix λ_n which is a run on $\alpha_n \otimes \beta_n$. Again, we replace $\boldsymbol{v}$ with $\boldsymbol{u}$ and $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ with the coarsening defined by $\boldsymbol{u}$.

Finally, by Ramsey's theorem there is an infinite subsequence $\boldsymbol{u}$ of $\boldsymbol{v}$ starting with $v_1, \dots, v_n$ such that all runs $\sigma(u_i, u_j)$ for $n < i < j$ have a common prefix κ_n which is a run on $\beta_n \otimes \beta_n$. We replace $\boldsymbol{v}$ with $\boldsymbol{u}$ and $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ with the coarsening defined by $\boldsymbol{u}$.

In the limit we obtain a comb $\boldsymbol{v}$ with generator $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ that is a subsequence of $\boldsymbol{w}$ and the desired decomposition of the runs $\rho(v_i, v_j)$ that is compatible with the $(\boldsymbol{\alpha}, \boldsymbol{\beta})$-decomposition of $v_i \otimes v_j$ for all $i < j$. □

It is not hard to see that such a comb of combs structure can be simulated by an alternating Büchi automaton, which would only yield a PSPACE-solution for the infinite clique problem. The following key lemma states that the runs $\mu_{i,j}$, $\nu_{i,j}$ from Lemma 4.2.6 can be chosen independently from i, which reduces the width of the run dag of the alternating automaton to a constant. Intuitively, the runs can be *merged*, which yields an easier run structure as shown in Figure 4.4.

Lemma 4.2.7. *If $\boldsymbol{w}$ is an infinite clique in a regular relation $R \subseteq \Sigma^* \times \Sigma^*$ given as an NFA $\mathcal{A}$, then there exist a generator $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of a subsequence $\boldsymbol{v}$ of $\boldsymbol{w}$, accepting runs $\rho(v_i, v_j)$ of $\mathcal{A}$ on $v_i \otimes v_j$, and runs $\kappa_i, \lambda_i, \mu_j, \nu_j$ such that*

$$\rho(v_i, v_j) = \kappa_1 \dots \kappa_{i-1}\lambda_i\mu_{i+1} \dots \mu_{j-1}\nu_j$$

is a decomposition compatible with the $(\boldsymbol{\alpha}, \boldsymbol{\beta})$-decomposition of $v_i \otimes v_j$ for all $i < j$.

Proof. Suppose that R has an infinite clique. Then there exist an infinite clique $\boldsymbol{w}$ in R generated by $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ and runs $\kappa_i, \lambda_i, \mu_{i,j}, \nu_{i,j}$ for $i < j$ as in Lemma 4.2.6.

It remains to ensure that $\mu_{i,j} = \mu_{i',j}$ and $\nu_{i,j} = \nu_{i',j}$ for all $i < i' < j$. To this end, consider the initial state of the run $\mu_{i,j}$. By Ramsey's theorem, there exist indices $k_1 < k_2 < \dots$ such that all runs μ_{k_i,k_j} have the same initial state. We define $v_i := w_{k_i}$ for all $i \geq 1$ and

$$\tilde{\kappa}_1 := \kappa_1 \dots \kappa_{k_2-1} \qquad \tilde{\lambda}_1 := \kappa_1 \dots \kappa_{k_1-1} \lambda_{k_1} \mu_{k_1,k_1+1} \dots \mu_{k_1,k_2-1}$$
$$\tilde{\kappa}_j := \kappa_{k_j} \dots \kappa_{k_{j+1}-1} \qquad \tilde{\lambda}_j := \lambda_{k_j} \mu_{k_j,k_j+1} \dots \mu_{k_j,k_{j+1}-1}$$
$$\tilde{\mu}_j := \mu_{k_1,k_j} \dots \mu_{k_1,k_{j+1}-1} \qquad \tilde{\nu}_j := \nu_{k_1,k_j}$$

for all $j \geq 2$. Observe that the composition $\tilde{\lambda}_i \tilde{\mu}_{i+1}$ forms a valid run since $\mu_{k_i,k_{i+1}}$ and $\mu_{k_1,k_{i+1}}$ have the same initial state. Then $\tilde{\kappa}_1 \dots \tilde{\kappa}_{i-1} \tilde{\lambda}_i \tilde{\mu}_{i+1} \dots \tilde{\mu}_{j-1} \tilde{\nu}_j$ is an accepting run on $v_i \otimes v_j$. Furthermore, this run decomposition is compatible with the $(\boldsymbol{\gamma}, \boldsymbol{\delta})$-decomposition of $v_i \otimes v_j$, where the generator $(\boldsymbol{\gamma}, \boldsymbol{\delta})$ is defined as

$$\delta_1 := \beta_1 \dots \beta_{k_2-1} \qquad \gamma_1 := \beta_1 \dots \beta_{k_1-1} \alpha_{k_1}$$
$$\delta_j := \beta_{k_j} \dots \beta_{k_{j+1}-1} \qquad \gamma_j := \alpha_{k_j}$$

for all $j \geq 2$. Note that the construction ensures that $|\gamma_i| \leq |\delta_i|$ for all $i \geq 1$. □

Using Lemma 4.2.7, we can now construct a Büchi automaton that simulates runs of the form as in Figure 4.4. This automaton is small since it only has to run one copy of the input automaton for each of the runs $\lambda_i, \kappa_i, \mu_i, \nu_i$ in parallel.

Proposition 4.2.8. *Given an NFA $\mathcal{A}$ for a relation $R \subseteq \Sigma^* \times \Sigma^*$, one can construct in logspace an NBA $\mathcal{B}$ over the alphabet $(\Sigma_\perp \times \Sigma) \cup \{\#\}$ such that:*

- *If $\boldsymbol{w}$ is an infinite clique in R, then $\mathcal{B}$ accepts an encoding of a comb $\boldsymbol{v}$ which is a subsequence of $\boldsymbol{w}$.*
- *If $\mathcal{B}$ accepts an encoding of a comb $\boldsymbol{w}$, then $\boldsymbol{w}$ is an infinite clique in R.*

Proof. Let $\Gamma := (\Sigma_\perp)^2$ and $\Gamma' := (\Sigma_\perp \times \Sigma) \cup \{\#\}$. Given an NFA $\mathcal{A} = (Q, \Gamma, \Delta, q_0, F)$ for R, we add to $\mathcal{A}$ a fresh state $\perp$ and transitions $\perp \xrightarrow{(a,b)} q$ for all $(a, b) \in \Gamma$ and $q \in Q_\perp$, where $Q_\perp := Q \cup \{\perp\}$.

The NBA $\mathcal{B} = ((Q_\perp)^4, \Gamma', \Delta^{\mathcal{B}}, q_0^{\mathcal{B}}, (Q_\perp)^4)$ simulates the runs $\kappa_i, \lambda_i, \mu_i, \nu_i$ from Lemma 4.2.7 in four components. Its initial state is $q_0^{\mathcal{B}} := (q_0, q_0, \perp, \perp)$ and it contains the following transtions:

- $(p, q, r, s) \xrightarrow{(a,b)}_{\mathcal{B}} (p', q', r', s')$ for all transitions $p \xrightarrow{(b,b)}_{\mathcal{A}} p'$, $q \xrightarrow{(a,b)}_{\mathcal{A}} q'$, $r \xrightarrow{(\perp,b)}_{\mathcal{A}} r'$, and either $a = \perp$ and $s = s' \in Q_\perp$ or $a \neq \perp$ and $s \xrightarrow{(\perp,a)}_{\mathcal{A}} s'$ in $\mathcal{A}$,
- $(p, q, q, s) \xrightarrow{\#}_{\mathcal{B}} (p, p, q, q)$ for all $p, q \in Q$ and $s \in F$,
- $(p_1, p_2, p_3, p_4) \xrightarrow{\varepsilon}_{\mathcal{B}} (q_1, q_2, q_3, q_4)$ if $p_i = q_i \in Q_\perp$ or $p_i \xrightarrow{\varepsilon}_{\mathcal{A}} q_i$ for all $i \in [1, 4]$.

The desired NBA is the intersection of $L(\mathcal{B})$ and the ω-regular language of all valid comb encodings, which can be computed using Proposition 2.3.10. □

We are now ready to prove Theorem 4.2.1. Note that to only solve the infinite clique problem, it suffices to check non-emptiness of the NBA constructed in Proposition 4.2.8. To also evaluate the Ramsey quantifier in presence of additional components, we add them to the clique components and enforce that they do not change throughout the clique. For convenience, we repeat the statement of the theorem.

Theorem 4.2.1. *Given a regular relation $R \subseteq (\Sigma^*)^{2d+k}$ by an NFA, one can construct in logspace an NFA for the relation $[\![\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!]$, where $|\boldsymbol{x}| = |\boldsymbol{y}| = d$ and $|\boldsymbol{z}| = k$.*

Proof. First assume that $d = 1$ and $k = 0$, i.e. R is a binary relation on words over Σ. We can construct in logspace an NFA $\mathcal{C}$ over Σ such that (i) for every infinite clique $\boldsymbol{w}$ of R some element w_i is accepted by $\mathcal{C}$ and (ii) if w is accepted by $\mathcal{C}$, then w belongs to an infinite clique of R. To be more precise, $\mathcal{C}$ accepts $\alpha_1 \in \Sigma^*$ if and only if some encoding $\text{enc}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is accepted by the Büchi automaton $\mathcal{B}$ from Proposition 4.2.8. This can be done in logspace as follows: First we construct an NBA $\hat{\mathcal{C}}$ over $\Sigma \cup \{\#\}$ which accepts all words of the form $\alpha_1 \#^\omega$ such that an encoding $\text{enc}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is accepted by $\mathcal{B}$. To this end, $\hat{\mathcal{C}}$ simulates $\mathcal{B}$ until the first # is read and switches to a copy of $\mathcal{B}$ where every transition label is replaced with #. Then $\hat{\mathcal{C}}$ is turned into an NFA $\mathcal{C}$ (which does not read the suffix $\#^\omega$) by replacing #-transitions with ε-transitions. Furthermore, $\mathcal{C}$ tracks the number of final states visited so far and accepts if and only if this number exceeds the number of states in $\hat{\mathcal{C}}$ (i.e. $\hat{\mathcal{C}}$ visits a cycle containing a final state and therefore there is an accepted suffix).

Now assume that $d = 1$ and let $R \subseteq (\Sigma^*)^{k+2}$ be given by an NFA $\mathcal{A}$. We construct an NFA $\mathcal{A}'$ for the regular binary relation

$$R' := \{(u \otimes c_1 \otimes \cdots \otimes c_k, v \otimes c_1 \otimes \cdots \otimes c_k) \mid (u, v, \boldsymbol{c}) \in R\} \subseteq \left((\Sigma_\perp)^{k+1}\right)^* \times \left((\Sigma_\perp)^{k+1}\right)^*.$$

Intuitively, we add the free components that are not bound by the Ramsey quantifier to the clique components using the convolution operation and we ensure that those components do not change throughout the clique. Note that $\mathcal{A}'$ can be constructed in logspace since it is obtained by taking each transition of $\mathcal{A}$ and duplicating the $\boldsymbol{c}$-coordinates, moving the v-coordinate, and replacing the first (resp. last) $k+1$ components with a single $\perp$ if they only consist of $\perp$-symbols. Let $\mathcal{C}'$ be the NFA described above, which accepts at least one word from each infinite R'-clique and only accepts elements of infinite R'-cliques. Projecting away the first component yields the desired NFA for $[\![\exists^{\text{ram}} x, y \colon R(x, y, \boldsymbol{z})]\!]$.

By Proposition 3.1.4, the general case where $R \subseteq (\Sigma^*)^{2d+k}$ can be reduced to the case where $d = 1$. □

4.3 Tree-Automatic Structures

We now turn to Ramsey quantifiers over tree-regular relations. Unlike the word case, the complexity of the infinite clique problem over tree-regular relations depends on how the relation is given.

Theorem 4.3.1. *The infinite clique problem over tree-regular relations $R \subseteq (\mathcal{T}_\Sigma)^{2d}$ is* EXP*-complete if R is given as NTA or D↓TA and* P*-complete if R is given as D↑TA.*

We will start with the surprising EXP lower bound of the infinite clique problem for NTAs and D↓TAs. We then continue with the evaluation of the Ramsey quantifier over (general) tree-regular relations, which proves the upper bounds in Theorem 4.3.1 since it reduces the infinite clique problem to checking non-emptiness.

Theorem 4.3.2. *Given a D↑TA (resp. ATA) for a tree-regular relation $R \subseteq (\mathcal{T}_\Sigma)^{2d+k}$, one can construct in polynomial (resp. exponential) time an NTA for the relation $[\![\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!]$, where $|\boldsymbol{x}| = |\boldsymbol{y}| = d$ and $|\boldsymbol{z}| = k$.*

In terms of tree-automatic structures, we can rephrase Theorem 4.3.2 again as follows:

Corollary 4.3.3. *Given a tree-automatic structure $\mathfrak{A}$ by a presentation $\mathfrak{p}$ and a formula $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ over $\mathfrak{A}$ with $|\boldsymbol{x}| = |\boldsymbol{y}|$, one can effectively compute an NTA for the relation $[\![\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!]_{\mathfrak{p}}$. Moreover, if a D↑TA for $[\![\varphi]\!]_{\mathfrak{p}}$ is given, the construction requires only polynomial time.*

In particular, Corollary 4.3.3 implies that the first-order theory enriched with the Ramsey quantifier of tree-automatic structures is decidable, which was already shown by Kartzow [Kar11].

Corollary 4.3.4. *The enriched theory $\mathrm{Th}^{\exists^{\text{ram}}}(\mathfrak{A})$ of every tree-automatic structure $\mathfrak{A}$ is decidable.*

We then consider the special cases where the given tree-regular relation is transitive or co-transitive.

4.3.1 EXP-Hardness

In this section we prove the exponential-time lower bound from Theorem 4.3.1 of the infinite clique problem over tree-regular relations given as NTA or D↓TA. It already holds for binary relations. This lower bound is surprising since over words the infinite clique problem can be reduced to non-emptiness of (word) Büchi automata, which is NL-complete. This is not the case over trees since non-emptiness of Büchi tree automata is P-complete.

We start with an intuitive explanation of the lower bound. To prove the upper bound in the word case, we used the fact that we can assume cliques $\boldsymbol{v}$ whose runs $\rho(v_i, v_j)$ can be merged into a *single* global run (Lemma 4.2.7). Over tree-regular relations this is not the case anymore. Consider a D↓TA $\mathcal{A}$ which behaves as follows on the convolution

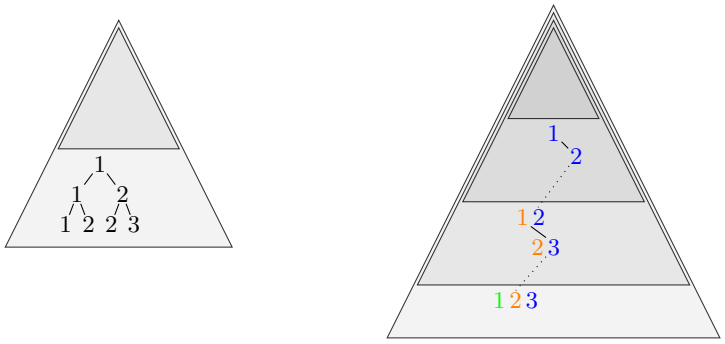


Figure 4.5: Left: Illustration of the tree automaton $\mathcal{A}$ tracking the number of right directions modulo n. Right: A path on which the runs of $\mathcal{A}$ are disjoint.

$t \otimes t'$ of two binary trees t, t' with $\mathrm{dom}(t) \subsetneq \mathrm{dom}(t')$: Starting from every node on the fringe of t, the automaton tracks the number of times it moves to a right child modulo some number n (see Figure 4.5 for a depiction). Now consider an increasing sequence of binary trees $t_1, t_2, \dots$ and the unique runs $\rho_{i,j}$ of $\mathcal{A}$ on $t_i \otimes t_j$. Figure 4.5 illustrates that we can always find a path on which the runs $\rho_{1,n}, \rho_{2,n}, \dots, \rho_{n-1,n}$ are disjoint. This behavior indicates that it is difficult to witness the existence of infinite cliques by a polynomially-sized Büchi tree automaton since the number of copies that have to be simulated in parallel is not constant anymore.

We extend this idea to a reduction from the intersection non-emptiness problem for NTAs, which by Proposition 2.3.18 is EXP-complete. For convenience, we use the following equivalent formulation of the intersection non-emptiness problem: Given an NTA $\mathcal{A} = (Q, \Sigma, \Delta, q_0)$ and states $q_1, \dots, q_n \in Q$, decide whether $\bigcap_{i=1}^{n} L(\mathcal{A}_{q_i})$ is non-empty. Here, $\mathcal{A}_{q_i}$ denotes the NTA $\mathcal{A}$ with initial state q_i.

We construct a relation R on decorated trees, which are obtained from a binary tree by attaching to every inner node u a ranked tree $\delta(u)$ over Σ. Let $\Gamma := \Sigma \cup \{a, c\}$ be a ranked alphabet with $a, c \notin \Sigma$ and $\mathrm{rk}(a) := 3$ and $\mathrm{rk}(c) := 0$. A *decorated tree* is a tree $t \in \mathcal{T}_\Gamma$ such that $t(\varepsilon) \in \{a, c\}$ and for all $u \in \mathrm{dom}(t)$ we have that

- if $t(u) = a$, then $t(u1) = t(u2) \in \{a, c\}$ and $t(u3) \in \Sigma$, and
- if $t(u) \in \Sigma$, then $t(ui) \in \Sigma$ for all $i \in [1, \mathrm{rk}(t(u))]$.

We denote by $a(t) := \{u \in \mathrm{dom}(t) \mid t(u) = a\}$ the nodes of t labeled with a and by $ac(t) := \{u \in \mathrm{dom}(t) \mid t(u) \in \{a, c\}\}$ the nodes labeled with a or c. The *decoration* of t is the function $\delta_t \colon a(t) \to \mathcal{T}_\Sigma$ such that $\delta_t(u) := t_{\downarrow u3}$ for all $u \in a(t)$, where $t_{\downarrow v}$ denotes the subtree of t rooted in $v \in \mathrm{dom}(t)$.

Let $\mathcal{A}' := (Q', \Gamma, \Delta', p_1)$ be the NTA where $Q' := Q \cup \{p_1, \dots, p_n\}$ for fresh states $p_1, \dots, p_n \notin Q$ and Δ' contains all transitions from Δ and the transitions

- $p_i \xrightarrow{a} (p_i, p_{i+1}, q_i)$ for all $i \in [1, n]$ where $p_{n+1} := p_1$ and
- $p_i \xrightarrow{c} ()$ for all $i \in [1, n]$.

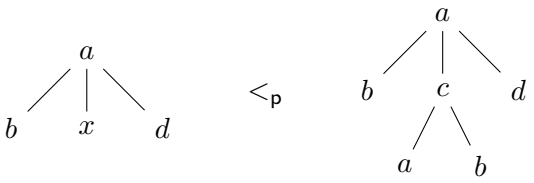


Figure 4.6: The context on the left is a proper prefix of the tree on the right.

We define the tree-regular relation $R \subseteq \mathcal{T}_\Gamma \times \mathcal{T}_\Gamma$ such that $(s,t) \in R$ if and only if s and t are decorated trees with $ac(s) \subseteq a(t)$ and $\mathcal{A}'$ accepts $t_{\downarrow u}$ for all $u \in \min(a(t) \setminus a(s))$. Here, the minimum is defined with respect to prefix ordering. It is easy to construct an NTA for R in logspace.

It remains to show that $\bigcap_{i=1}^n L(\mathcal{A}_{q_i}) \neq \emptyset$ if and only if R contains an infinite clique. For the "only if" direction let $t \in \mathcal{T}_\Sigma$ be a tree that is accepted by $\mathcal{A}_{q_i}$ for all $i \in [1,n]$. For all $i \geq 0$ we define the decorated tree $t_i \in \mathcal{T}_\Gamma$ such that $ac(t_i) = \bigcup_{j=0}^i \{1,2\}^j$ and $\delta_{t_i}(u) = t$ for all $u \in a(t_i)$. It is easy to verify that $(t_i, t_j) \in R$ for all $i < j$.

Conversely, consider a sequence of decorated trees $t_i \in \mathcal{T}_\Gamma$ for $i \geq 0$ with $(t_i, t_j) \in R$ for all $i < j$. We define nodes $v_1, \ldots, v_n$ with

- $v_i \in \min(a(t_i) \setminus a(t_{i-1}))$ for all $i \in [1,n]$ and
- $v_{i+1} = v_i 21^{k_i}$ for all $i \in [1, n-1]$ and some $k_i \geq 0$.

We can choose $v_1 \in \min(a(t_1) \setminus a(t_0))$ arbitrarily, which defines $v_2, \ldots, v_n$ uniquely. Since $(t_{i-1}, t_n) \in R$ and $v_i \in \min(a(t_n) \setminus a(t_{i-1}))$, the subtree $t_{n\downarrow v_i}$ is accepted by $\mathcal{A}'$ for all $i \in [1,n]$. By definition of $\mathcal{A}'$, there exist accepting runs on $t_{n\downarrow v_n}$ starting from $p_1, \ldots, p_n$ since for all $i \in [1, n-1]$ the accepting run on $t_{n\downarrow v_i}$ is in state p_{n-i+1} at node v_n. Therefore, the tree $\delta_{t_n}(v_n) \in \mathcal{T}_\Sigma$ is accepted by $\mathcal{A}$ starting from all states $q_1, \ldots, q_n$.

We note that EXP-hardness already holds if R is given by a D↓TA since the intersection non-emptiness problem is EXP-hard already for D↓TAs (Proposition 2.3.18) and if the automaton $\mathcal{A}$ is a D↓TA, then the constructed relation R from the above proof can also be accepted by a D↓TA. Moreover, the reduction can be adapted to recurrent reachability by setting the target set to $\mathcal{T}_\Sigma$, which proves the exponential lower bound in Corollary 4.4.1.

4.3.2 Tree Combs

To prove the upper bounds stated in Theorem 4.3.2, we extend the notion of combs to the tree case. To this end, we need the definition of contexts that allows us to decompose a tree vertically.

Let $x \notin \Sigma$ be a *variable*. The set $\mathcal{C}_\Sigma$ of all *contexts* over Σ contains all unranked trees t over $\Sigma \cup \{x\}$ such that every node $u \in \mathrm{dom}(t)$ with $t(u) = x$ is a leaf, called *hole*. We partition $\mathrm{dom}(t) = \mathrm{nodes}(t) \cup \mathrm{holes}(t)$ into nodes and holes. The *size* of a context t is $|t| := |\mathrm{nodes}(t)|$ and if $|t| \geq 1$, then t is called *nontrivial*. For contexts $s, t_1, \ldots, t_n \in \mathcal{C}_\Sigma$ with $|\mathrm{holes}(s)| = n$ we denote by $s[t_1, \ldots, t_n]$ the context obtained by replacing the i-th

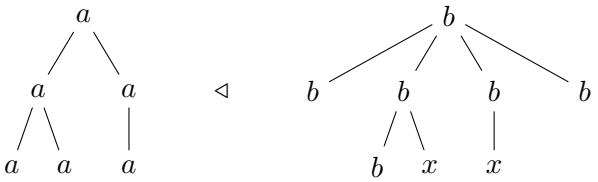


Figure 4.7: An example of a tree and a context that are related by $\triangleleft$.

hole in lexicographic order with t_i. For two contexts $s_1, s_2 \in \mathcal{C}_\Sigma$ we call s_1 a *prefix* of s_2, denoted by $s_1 \leq_\mathsf{p} s_2$, if $s_1[t_1, \ldots, t_n] = s_2$ for some contexts $t_1, \ldots, t_n \in \mathcal{C}_\Sigma$. If $n > 0$ and each context t_i is nontrivial, then s_1 is a *proper prefix* of s_2, denoted by $s_1 <_\mathsf{p} s_2$. An example of a proper prefix is illustrated in Figure 4.6.

A *context forest* of width n over an alphabet Σ is a finite sequence $\tau = (c_i)_{1 \leq i \leq n}$ of contexts $c_i \in \mathcal{C}_\Sigma$. We denote the set of all context forests over Σ by $\mathcal{F}_\Sigma$. Context forests of width 1 are regarded as contexts. We say that τ is *nontrivial* if $n \geq 1$ and $|c_i| \geq 1$ for all $i \in [1, n]$. If the c_i are trees in $\mathcal{U}_\Sigma$, we call τ just a *forest*. We define the concatenation of a context $\tau_1 \in \mathcal{C}_\Sigma$, where $|\mathrm{holes}(\tau_1)| = n$, with a context forest $\tau_2 = (c_1, \ldots, c_n) \in \mathcal{F}_\Sigma$ of width n by $\tau_1\tau_2 := \tau_1[c_1, \ldots, c_n]$. We write $\tau_1\tau_2 \ldots \tau_n$ for a context τ_1 and context forests $\tau_2, \ldots, \tau_n$ assuming left-associativity. Here, we implicitly assume that the width of τ_i matches $|\mathrm{holes}(\tau_1 \ldots \tau_{i-1})|$. If t is a tree and s is a context, we write $t \triangleleft s$ if $\mathrm{dom}(t) \cap \mathrm{holes}(s) = \emptyset$. See Figure 4.7 for an example. For a forest $\alpha = (t_i)_{i \leq n}$ and a context forest $\beta = (c_i)_{i \leq n}$ we also write $\alpha \triangleleft \beta$ if $t_i \triangleleft c_i$ for all $i \in [1, n]$.

We are now ready to formally define the notion of a comb of trees. An infinite sequence $\boldsymbol{t} = (t_i)_{i \geq 1}$ of ranked trees is called a *comb* if there is a sequence of forests $\boldsymbol{\alpha} = (\alpha_i)_{i \geq 1}$ and a sequence of nontrivial context forests $\boldsymbol{\beta} = (\beta_i)_{i \geq 1}$ such that for all $i \geq 1$ we have $t_i = \beta_1 \ldots \beta_{i-1}\alpha_i$ and $\alpha_i \triangleleft \beta_i$. The pair $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is called *generator* of the comb $\boldsymbol{t}$. Since the trees t_i are ranked, also the trees in α_i and the contexts in β_i are ranked. The property $\alpha_i \triangleleft \beta_i$ should be compared to the property $|\alpha_i| \leq |\beta_i|$ in word combs. It ensures that every forest α_i does not touch any context forest α_j, β_j for $j > i$. For an abstract illustration of a tree comb we refer to Figure 4.8.

The following lemma is the tree analogue of Lemma 4.2.5.

Lemma 4.3.5. *Any infinite sequence $\boldsymbol{t}$ of pairwise distinct ranked trees $t_i \in \mathcal{T}_\Sigma$ over a finite ranked alphabet Σ contains a comb as a subsequence.*

Proof. As in the word case, it suffices to show that for any infinite set $T \subseteq \mathcal{T}_\Sigma$ there exists a comb over T. Consider the following finitely branching infinite tree whose nodes are contexts from $\mathcal{C}_\Sigma$. The root is the trivial context only consisting of a hole and no nodes. The children of a context s are the contexts of the form $s[t_1, \ldots, t_n]$ where each t_i is a context of size 1, i.e. consisting of a root node all of whose children are holes. Observe that all trees in $\mathcal{T}_\Sigma$ occur as nodes in the infinite tree. The set of all ancestors of trees in T form an infinite subtree, which contains an infinite path $s_0 <_\mathsf{p} s_1 <_\mathsf{p} s_2 <_\mathsf{p} \ldots$ of contexts by Kőnig's Lemma. For every $i \geq 1$ there exists a tree $t_i \in T$ which contains s_{i-1} as a prefix. Since the minimal level of a hole in s_i is strictly increasing, for every

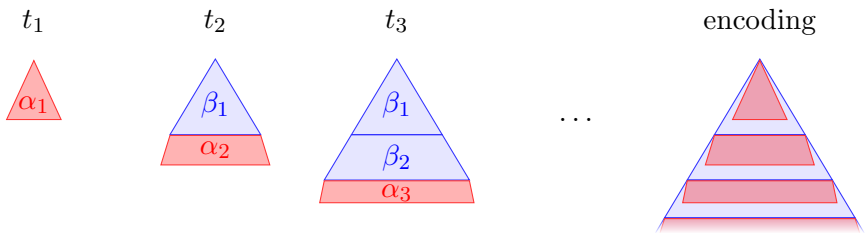


Figure 4.8: An example tree comb and its encoding as an infinite tree. In this example the generator satisfies $\mathrm{dom}(\alpha_i) \subseteq \mathrm{dom}(\beta_i)$ whereas general generators only satisfy $\alpha_i \lhd \beta_i$.

$i \geq 1$ there exists a $j \geq i$ with $t_i \lhd s_j$. Hence, one can inductively construct indices $1 = k_1 < k_2 < \ldots$ such that $t_{k_i+1} \lhd s_{k_{i+1}}$ for all $i \geq 1$. Then $(t_{k_i+1})_{i\geq 1}$ is a comb where the generator $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is defined such that $s_{k_{i+1}} = \beta_1 \ldots \beta_i$ and $t_{k_i+1} = \beta_1 \ldots \beta_{i-1}\alpha_i$ for $i \geq 1$. The comb property $\alpha_i \lhd \beta_i$ follows from $t_{k_i+1} \lhd s_{k_{i+1}}$. □

To define the encoding $\mathrm{enc}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of a comb generator, we need a few more definitions. For a tree t and context s with $t \lhd s$ we define the convolution $t \otimes s$ as before, but every $(\bot, x)$ is replaced with x. That is, $t \otimes s$ is again a context. We extend the convolution in a natural way to forests and context forests of the same width. If $\tau = (c_1, \ldots, c_n)$ is a context forest, let $\overline{\tau}$ be obtained from τ by attaching a new #-labeled root to each of the n contexts c_i. We can now define the *encoding* of a comb with generator $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ as the infinite tree

$$\mathrm{enc}(\boldsymbol{\alpha}, \boldsymbol{\beta}) := (\alpha_1 \otimes \beta_1)(\overline{\alpha_2 \otimes \beta_2})(\overline{\alpha_3 \otimes \beta_3}) \ldots$$

over the ranked alphabet $\Omega := (\Sigma_\bot)^2 \cup \{\#\}$ where $\# \notin \Sigma$ and $\mathrm{rk}(\#) = 1$. See Figure 4.8 for an illustration of the encoding. Here, the forests α_i are colored red and the context forests β_i are colored blue. It is not hard to see that there is an NBTA that accepts the set Enc_Σ of all comb encodings over the alphabet Σ.

4.3.3 General Relations

In this section we show how to evaluate Ramsey quantifiers for arbitrary tree-regular relations (Theorem 4.3.2). As in the word case, we first focus on binary relations $R \subseteq T_\Sigma \times T_\Sigma$, i.e. we assume that $d = 1$ and $k = 0$ in Theorem 4.3.2.

In Chapter 2 we only defined runs of NTAs on trees. For the following we need to extend the run definition to contexts. A *run* of an NTA $\mathcal{A} = (Q, \Sigma, \Delta, q_0)$ on a nontrivial ranked context $t \in \mathcal{C}_\Sigma$ is a context ρ over the alphabet $Q \times Q^{\leq m}$, where $m := \max\{\mathrm{rk}(a) \mid a \in \Sigma\}$, with $\mathrm{nodes}(\rho) = \mathrm{nodes}(t)$ and $\mathrm{holes}(\rho) = \mathrm{holes}(t)$ such that for all $u \in \mathrm{nodes}(\rho)$ with $\rho(u) = (q, q_1, \ldots, q_r)$ we have $(q, t(u), q_1, \ldots, q_r) \in \Delta$ and $\rho(ui) \in \{q_i\} \times Q^{\leq m}$ for all $i \in [1, r]$. Here, $Q^{\leq m} := \bigcup_{i=0}^{m} Q^i$. Note that each node in a run also carries the first component of the labels of its children with the purpose of predetermining the states in the holes.

We define a *decomposition* of a context t as $t = \tau_1 \dots \tau_n$ where the τ_i are context forests. For a generator $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of a comb $\boldsymbol{t}$ we define the $(\boldsymbol{\alpha}, \boldsymbol{\beta})$-decomposition of $t_i \otimes t_j$ for all $i < j$ as in the word case as

$$\begin{bmatrix} t_i \\ t_j \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_1 \end{bmatrix} \dots \begin{bmatrix} \beta_{i-1} \\ \beta_{i-1} \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} \begin{bmatrix} \varepsilon \\ \beta_{i+1} \end{bmatrix} \dots \begin{bmatrix} \varepsilon \\ \beta_{j-1} \end{bmatrix} \begin{bmatrix} \varepsilon \\ \alpha_j \end{bmatrix}.$$

We say that a decomposition $\rho = \rho_1 \dots \rho_n$ of a run of an NTA is *compatible* with a decomposition $t = \tau_1 \dots \tau_n$ of a context if $\rho_1 \dots \rho_i$ is a run on $\tau_1 \dots \tau_i$ for all $i \in [1, n]$. Note that the above definition of a run ensures that $\rho_1 \dots \rho_i$ already determines the first component of the root labels of ρ_{i+1} for all $i < n$.

We say that a generator $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of a comb $\boldsymbol{s}$ is *coarser* than a generator $(\boldsymbol{\gamma}, \boldsymbol{\delta})$ of a comb $\boldsymbol{t}$ if there exist indices $k_1 < k_2 < \dots$ such that $s_i = t_{k_i}$ and $\beta_1 \dots \beta_i = \delta_1 \dots \delta_{k_i}$ for all $i \geq 1$. In this case we also say that $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is the *coarsening* of $(\boldsymbol{\gamma}, \boldsymbol{\delta})$ defined by the subsequence $\boldsymbol{s}$ of $\boldsymbol{t}$.

Lemma 4.3.6. *Let $\boldsymbol{t}$ be a comb generated by $(\boldsymbol{\gamma}, \boldsymbol{\delta})$ that forms an infinite clique in a tree-regular relation $R \subseteq \mathcal{T}_\Sigma \times \mathcal{T}_\Sigma$ given as an NTA $\mathcal{A}$. There exist a coarsening $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of $(\boldsymbol{\gamma}, \boldsymbol{\delta})$ generating a comb $\boldsymbol{s}$, accepting runs $\rho(s_i, s_j)$ of $\mathcal{A}$ on $s_i \otimes s_j$, and context forests $\kappa_i, \lambda_i, \mu_{i,j}, \nu_{i,j}$ such that*

$$\rho(s_i, s_j) = \kappa_1 \dots \kappa_{i-1} \lambda_i \mu_{i,i+1} \dots \mu_{i,j-1} \nu_{i,j}$$

is a decomposition compatible with the $(\boldsymbol{\alpha}, \boldsymbol{\beta})$-decomposition of $s_i \otimes s_j$ for all $i < j$.

Proof. The proof is similar to the proof of Lemma 4.2.6 in the word case. We emphasize that the pigeonhole principle and Ramsey's theorem can be applied as in the word case since the unique prefixes of the runs of the NTA that are runs on a given context have bounded size. Let $\boldsymbol{t}$ be a comb generated by $(\boldsymbol{\gamma}, \boldsymbol{\delta})$ that forms an infinite clique in R and $\rho(t_i, t_j)$ be an accepting run of $\mathcal{A}$ on $t_i \otimes t_j$ for all $1 \leq i < j$. We establish the run structure as illustrated in Figure 4.3 column-wise.

Assume we already defined a coarsening $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of $(\boldsymbol{\gamma}, \boldsymbol{\delta})$ generating a comb $\boldsymbol{s}$ and context forests $(\kappa_i)_{i<n}$, $(\lambda_i)_{i<n}$, $(\mu_{i,j})_{i<j<n}$, and $(\nu_{i,j})_{i<j<n}$ for some $n \geq 1$ such that

$$\begin{aligned} \rho(s_i, s_j) &= \kappa_1 \dots \kappa_{i-1} \lambda_i \mu_{i,i+1} \dots \mu_{i,n-1} \tau_{i,j} \\ \rho(s_{i'}, s_{j'}) &= \kappa_1 \dots \kappa_{n-1} \sigma_{i',j'} \end{aligned}$$

for forests $\tau_{i,j}$, $\sigma_{i',j'}$ for all $1 \leq i < n \leq j$ and $n \leq i' < j'$. For all $1 \leq i < n$ we just set $\nu_{i,n} := \tau_{i,n}$.

We now define $\mu_{i,n}$ successively for each $1 \leq i < n$. In step i we apply the pigeonhole principle to get an infinite subsequence $\boldsymbol{r}$ of $\boldsymbol{s}$ starting with $s_1, \dots, s_n$ such that all runs $\rho(r_i, r_j)$ for $j > n$ have a common prefix $\kappa_1 \dots \kappa_{i-1} \lambda_i \mu_{i,i+1} \dots \mu_{i,n}$ which is a run on $(\beta_1 \otimes \beta_1) \dots (\beta_{i-1} \otimes \beta_{i-1})(\alpha_i \otimes \beta_i)(\varepsilon \otimes \beta_{i+1}) \dots (\varepsilon \otimes \beta_n)$. At the end of step i, we replace $\boldsymbol{s}$ with $\boldsymbol{r}$ and we replace $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ with the coarsening defined by $\boldsymbol{r}$.

Next we define λ_n. By the pigeonhole principle, there exists an infinite subsequence $\boldsymbol{r}$ of $\boldsymbol{s}$ starting with $s_1, \dots, s_n$ such that all runs $\rho(r_n, r_j)$ for $j > n$ have a common prefix

$\kappa_1 \dots \kappa_{n-1}\lambda_n$ which is a run on $(\beta_1 \otimes \beta_1) \dots (\beta_{n-1} \otimes \beta_{n-1})(\alpha_n \otimes \beta_n)$. Again, we replace $\boldsymbol{s}$ with $\boldsymbol{r}$ and $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ with the coarsening defined by $\boldsymbol{r}$.

Finally, by Ramsey's theorem there is an infinite subsequence $\boldsymbol{r}$ of $\boldsymbol{s}$ starting with $s_1, \dots, s_n$ such that all runs $\rho(r_i, r_j)$ for $n < i < j$ have a common prefix $\kappa_1 \dots \kappa_n$ which is a run on $(\beta_1 \otimes \beta_1) \dots (\beta_n \otimes \beta_n)$. We replace $\boldsymbol{s}$ with $\boldsymbol{r}$ and $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ with the coarsening defined by $\boldsymbol{r}$.

In the limit we obtain a coarsening $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of $(\boldsymbol{\gamma}, \boldsymbol{\delta})$ generating a comb $\boldsymbol{s}$ and the desired decomposition of the runs $\rho(s_i, s_j)$ that is compatible with the $(\boldsymbol{\alpha}, \boldsymbol{\beta})$-decomposition of $s_i \otimes s_j$ for all $i < j$. □

If R is given by a D↑TA, we can compute in polynomial time a *nondeterministic* Büchi tree automaton for the encoding of infinite cliques. The proof idea is that the runs $\mu_{i,j}$ and $\nu_{i,j}$ in Lemma 4.3.6 only depend on j.

Proposition 4.3.7. *Given a D↑TA $\mathcal{A}$ for a tree-regular relation $R \subseteq \mathcal{T}_\Sigma \times \mathcal{T}_\Sigma$, one can construct in logspace an NBTA $\mathcal{B}$ over the ranked alphabet $\Omega = (\Sigma_\perp)^2 \cup \{\#\}$ such that:*

- *If $\boldsymbol{t}$ is an infinite clique in R, then $\mathcal{B}$ accepts an encoding of a comb $\boldsymbol{s}$ which is a subsequence of $\boldsymbol{t}$.*
- *If $\mathcal{B}$ accepts $t \in \mathcal{T}_\Omega^\infty$, then t is an encoding of a comb $\boldsymbol{t}$ that is an infinite clique in R.*

Proof. Let $\mathcal{A}' = (Q, (\Sigma_\perp)^2, \Delta, q_0)$ be the NTA that is obtained by reverting the transitions of the D↑TA $\mathcal{A}$, where q_0 is a new initial state with $(q_0, a, q_1, \dots, q_r) \in \Delta$ for all transitions $(q_1, \dots, q_r, a, q)$ of $\mathcal{A}$ with final state q. We observe that $\mathcal{A}'$ has the same runs (without q_0 in the root) as $\mathcal{A}$ on trees. Let $\boldsymbol{t}$ be an infinite clique in R. By Lemma 4.2.5, we can assume that $\boldsymbol{t}$ is a comb. We apply Lemma 4.3.6 on $\mathcal{A}'$ and $\boldsymbol{t}$ to get a subcomb $\boldsymbol{s}$ of $\boldsymbol{t}$ generated by $(\boldsymbol{\alpha}, \boldsymbol{\beta})$, accepting runs $\rho(s_i, s_j)$ of $\mathcal{A}'$ on $s_i \otimes s_j$, and context forests $\kappa_i, \lambda_i, \mu_{i,j}, \nu_{i,j}$ such that

$$\rho(s_i, s_j) = \kappa_1 \dots \kappa_{i-1}\lambda_i\mu_{i,i+1} \dots \mu_{i,j-1}\nu_{i,j}$$

is a decomposition compatible with the $(\boldsymbol{\alpha}, \boldsymbol{\beta})$-decomposition of $s_i \otimes s_j$ for all $i < j$. Moreover, we have that $\mu_{i,j}$ and $\nu_{i,j}$ only depend on j since there are unique runs of $\mathcal{A}$ on the trees in the forests $(\varepsilon \otimes \beta_j)(\varepsilon \otimes \alpha_{j+1})$ and $\varepsilon \otimes \alpha_j$ (and $\mu_{i,j}, \nu_{i,j}$ cannot have q_0 in the roots). Thus, we can just write μ_j and ν_j for all $j > 1$.

We now construct an NBTA $\mathcal{B}$ over the alphabet Ω which accepts precisely all comb encodings $\mathrm{enc}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of a generator $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ with the above properties. Since the set of all comb encodings can be accepted by an NBTA and intersection of two NBTAs can be performed in logspace by Proposition 2.3.23, we can assume that the input tree is already a valid comb encoding.

A state of $\mathcal{B}$ consists of four components in which $\kappa_j, \lambda_j, \mu_j, \nu_j$ are simulated. To handle the special case where only κ_1 and λ_1 are simulated, we add a state $\perp$ to $\mathcal{A}'$ with transitions $\perp \xrightarrow{(a,b)} \boldsymbol{q}$ for all symbols $(a, b) \in (\Sigma_\perp)^2$ of rank r and $\boldsymbol{q} \in (Q_\perp)^r$. The NBTA $\mathcal{B}$ has the state set $(Q_\perp)^4$, initial state $(q_0, q_0, \perp, \perp)$, and the transitions

- $(p, q, r, s) \xrightarrow{(a,b)} \boldsymbol{p} \otimes \boldsymbol{q} \otimes \boldsymbol{r} \otimes \boldsymbol{s}$ if $\mathcal{A}'$ contains the transitions $p \xrightarrow{(b,b)} \boldsymbol{p}$, $q \xrightarrow{(a,b)} \boldsymbol{q}$, $r \xrightarrow{(\bot,b)} \boldsymbol{r}$, and $s \xrightarrow{(\bot,a)} \boldsymbol{s}$,

- $(p, q, q, \bot) \xrightarrow{\#} (p, p, q, q)$ for all $p, q \in Q$.

As final states we can take the set of all states $(Q_\bot)^4$. Note that for the convolution $\boldsymbol{p} \otimes \boldsymbol{q} \otimes \boldsymbol{r} \otimes \boldsymbol{s}$ we regard the sequences of states as words where the padding symbol is the state $\bot$. Correctness follows from the previous observations, where we remark that after reading $\#$, we can start the simulation of κ_j and λ_j (resp. μ_j and ν_j) in the same state since a run on a context already predetermines the states in the holes. □

Note that the proof above does not work for D↓TAs since $\mu_{i,j}$ and $\mu_{i',j}$ (resp. $\nu_{i,j}$ and $\nu_{i',j}$) are appended to different prefixes for $i \neq i'$, which means that the states at their roots can differ. Thus, we cannot just write μ_j (resp. ν_j) in that case.

We are now ready to prove Theorem 4.3.2. Note that if $\mathcal{A}$ is an ATA, we can first apply Proposition 2.3.20 to transform $\mathcal{A}$ into a D↑TA in exponential time and then apply the polynomial-time algorithm for D↑TAs. Hence, in the proof it suffices to handle the case where $\mathcal{A}$ is a D↑TA. The proof follows the same strategy as the proof of Theorem 4.2.1 in the word case. For convenience, let us repeat the statement of the theorem.

Theorem 4.3.2. *Given a D↑TA (resp. ATA) for a tree-regular relation $R \subseteq (\mathcal{T}_\Sigma)^{2d+k}$, one can construct in polynomial (resp. exponential) time an NTA for the relation $[\![\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!]$, where $|\boldsymbol{x}| = |\boldsymbol{y}| = d$ and $|\boldsymbol{z}| = k$.*

Proof. We first assume that $d = 1$ and $k = 0$. From the NBTA $\mathcal{B} = (Q, \Omega, \Delta, q_0, F)$ in Proposition 4.3.7 we can construct in polynomial time an NTA $\mathcal{C}$ over Σ which accepts $\alpha_1 \in \mathcal{T}_\Sigma$ if and only if some encoding $\mathrm{enc}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of a comb is accepted by $\mathcal{B}$. Indeed, we define $\mathcal{C} := (Q, \Sigma, \Delta', q_0)$ such that for all $q \in Q$, $a \in \Sigma$, and $p_i \in Q$ for $1 \leq i \leq \mathrm{rk}(a)$ we let

$$(q, a, (p_i)_{i \leq \mathrm{rk}(a)}) \in \Delta'$$

if and only if there exist $b \in \Sigma_\bot$ and $p_i \in Q$ for $\mathrm{rk}(a) < i \leq \mathrm{rk}(\binom{a}{b})$ such that

$$(q, \tbinom{a}{b}, (p_i)_{i \leq \mathrm{rk}(\binom{a}{b})}) \in \Delta$$

and $\mathcal{B}$ accepts some tree from state p_i for all $i > \mathrm{rk}(a)$. Note that $\mathcal{C}$ can be constructed in polynomial time given $\mathcal{B}$ since we need to perform a polynomial number of non-emptiness checks on $\mathcal{B}$, each of which takes polynomial time by Proposition 2.3.24. The NTA $\mathcal{C}$ satisfies that (i) for every infinite clique $\boldsymbol{t}$ of R some element t_i is accepted by $\mathcal{C}$ and (ii) if t is accepted by $\mathcal{C}$, then t belongs to an infinite clique of R.

Now assume that $d = 1$ and let $R \subseteq (\mathcal{T}_\Sigma)^{k+2}$ be given by a D↑TA $\mathcal{A}$. As in the word case, we construct a D↑TA $\mathcal{A}'$ for the binary relation

$$R' := \{(s \otimes c_1 \otimes \cdots \otimes c_k, t \otimes c_1 \otimes \cdots \otimes c_k) \mid (s, t, \boldsymbol{c}) \in R\} \subseteq \mathcal{T}_{(\Sigma_\bot)^{k+1}} \times \mathcal{T}_{(\Sigma_\bot)^{k+1}}.$$

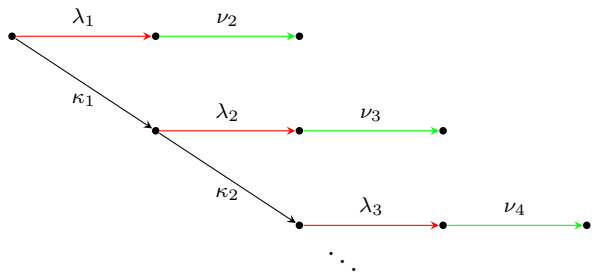


Figure 4.9: For transitive relations it suffices to consider a simplification of the run structure in Lemma 4.3.6 of the form $\rho(s_i, s_{i+1}) = \kappa_1 \dots \kappa_{i-1}\lambda_i\nu_{i+1}$.

Let $\mathcal{C}'$ be the NTA described above that accepts at least one tree from each infinite R'-clique and only accepts elements of infinite R'-cliques. Projecting away the first component using Proposition 2.3.33 yields the desired NTA for $[\![\exists^{\text{ram}} x, y \colon R(x, y, \boldsymbol{z})]\!]$.

By Proposition 3.1.4, the general case where $R \subseteq (\mathcal{T}_\Sigma)^{2d+k}$ can be reduced to the case where $d = 1$. □

4.3.4 Transitive Relations

Recall that a relation $R \subseteq A^{2d+k}$ is *transitive* if the binary relation $\{(\boldsymbol{a}, \boldsymbol{b}) \in A^d \times A^d \mid (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \in R\}$ is transitive for all $\boldsymbol{c} \in A^k$. In this section we show that if we assume R to be transitive, then the Ramsey quantifier can be evaluated in polynomial time even if R is given by an NTA. As before, we first assume that R is a binary relation, i.e. $d = 1$ and $k = 0$. The central observation is that in the transitive case an infinite sequence $\boldsymbol{t}$ of pairwise distinct elements forms a clique in R if and only if $(t_i, t_{i+1}) \in R$ for all $i \geq 1$. Thus, it suffices to consider a simplification of the run structure in Lemma 4.3.6 as illustrated in Figure 4.9. In particular, the $\mu_{i,j}$ are not needed and instead of all $\nu_{i,j}$ we only need $\nu_{i+1} := \nu_{i,i+1}$ for all $i \geq 1$. As in the general case, we start with the construction of an NBTA that accepts encodings of combs forming a clique in R.

Proposition 4.3.8. *Given an NTA $\mathcal{A}$ for a transitive tree-regular relation $R \subseteq \mathcal{T}_\Sigma \times \mathcal{T}_\Sigma$, one can construct in polynomial time an NBTA $\mathcal{B}$ over the ranked alphabet $\Omega = (\Sigma_\perp)^2 \cup \{\#\}$ such that:*

- *If $\boldsymbol{t}$ is an infinite clique in R, then $\mathcal{B}$ accepts an encoding of a comb $\boldsymbol{s}$ which is a subsequence of $\boldsymbol{t}$.*
- *If $\mathcal{B}$ accepts $t \in \mathcal{T}_\Omega^\infty$, then t is an encoding of a comb $\boldsymbol{t}$ that is an infinite clique in R.*

Proof. Let $\mathcal{A} = (Q, (\Sigma_\perp)^2, \Delta, q_0)$. We construct an NBTA $\mathcal{B}$ over Ω that accepts precisely all comb encodings $\text{enc}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ with the simplified run structure as in Figure 4.9. The construction is similar to the one in Proposition 4.3.7 and only removes the component that simulates μ_j. Again, we can assume that the input tree is already a valid comb encoding. A state in $\mathcal{B}$ consists of three components in which κ_i, λ_i, and ν_i are simulated.

To handle the special case at the beginning where only κ_1 and λ_1 are simulated, we add a state $\bot$ to $\mathcal{A}$ with transitions $\bot \xrightarrow{(a,b)} \boldsymbol{q}$ for all symbols $(a,b) \in (\Sigma_\bot)^2$ of rank r and $\boldsymbol{q} \in (Q_\bot)^r$. The NBTA $\mathcal{B}$ has the state set $(Q_\bot)^3$, initial state $(q_0, q_0, \bot)$, and the transitions

- $(p,q,r) \xrightarrow{(a,b)} \boldsymbol{p} \otimes \boldsymbol{q} \otimes \boldsymbol{r}$ if $\mathcal{A}$ contains the transitions $p \xrightarrow{(b,b)} \boldsymbol{p}$, $q \xrightarrow{(a,b)} \boldsymbol{q}$, and $r \xrightarrow{(\bot,a)} \boldsymbol{r}$,
- $(p,q,\bot) \xrightarrow{\#} (p,p,q)$ for all $p,q \in Q$.

As final states we can take the set of all states $(Q_\bot)^3$. □

Using the polynomially-sized NBTA $\mathcal{B}$ from Proposition 4.3.8, the following theorem can be proved analogously to Theorem 4.3.2. Here, we remark that the constructions in the proof preserve transitivity of the relations.

Theorem 4.3.9. *Given an NTA for a transitive tree-regular relation $R \subseteq (\mathcal{T}_\Sigma)^{2d+k}$, one can construct in polynomial time an NTA for the relation $[\![\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!]$, where $|\boldsymbol{x}| = |\boldsymbol{y}| = d$ and $|\boldsymbol{z}| = k$. In particular, the infinite clique problem over transitive tree-regular relations $R \subseteq (\mathcal{T}_\Sigma)^{2d}$ is* P*-complete.*

The P lower bound already holds for binary relations given as D↑TA or D↓TA by a logspace reduction from recurrent reachability (Proposition 3.2.1), which by Corollary 4.4.1 is P-complete for transitive relations.

4.3.5 Co-transitive Relations

Recall that a relation R is *co-transitive* if its complement $\overline{R}$ is transitive. In this section we show that if R is a co-transitive tree-regular relation, then the infinite clique problem can be solved in polynomial time. Again, we first assume R to be binary, i.e. $d = 1$.

A context is called *monadic* if it has exactly one hole. We will show that if a co-transitive tree-regular relation has an infinite clique, then there exists one which is a comb generated by a *monadic generator* $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ in which all β_i are monadic contexts. This also implies that all α_i are trees.

Lemma 4.3.10. *If a co-transitive tree-regular relation $R \subseteq \mathcal{T}_\Sigma \times \mathcal{T}_\Sigma$ has an infinite clique over a tree-regular language $L \subseteq \mathcal{T}_\Sigma$, then there exist an infinite clique $\boldsymbol{t}$ of R over L and a nontrivial monadic context β such that $t_1 \lhd \beta$ and $\beta \leq_{\mathsf{p}} t_i$ for all $i \geq 2$.*

Proof. Let $\mathcal{A} = (Q, (\Sigma_\bot)^2, \Delta, F)$ and $\mathcal{B} = (P, \Sigma, \Lambda, E)$ be D↑TAs for R and L, respectively. Suppose that $\boldsymbol{t}$ is an infinite clique in R over L. For $j \geq 2$ let c_j be the unique context with $\mathrm{nodes}(c_j) = \mathrm{dom}(t_1) \cap \mathrm{dom}(t_j)$ and $c_j \leq_{\mathsf{p}} t_j$. Notice that c_j is nontrivial since t_1 and t_j contain the root. Furthermore, we have $t_1 \lhd c_j$ since any hole $u \in \mathrm{holes}(c_j)$ is contained in $\mathrm{dom}(t_j) \setminus \mathrm{dom}(t_1)$. Since $\mathrm{nodes}(c_j) \subseteq \mathrm{dom}(t_1)$ for all $j \geq 2$, there are only finitely many distinct c_j. Hence, by the pigeonhole principle, we can reduce $\boldsymbol{t}$ to a subsequence starting with t_1 such that there is some context c with $c_j = c$ for all $j \geq 2$.

Suppose that $v^1, \dots, v^n$ are the holes of c in lexicographical order and $t_j = c[t_j^1, \dots, t_j^n]$ for some trees t_j^k. Again, by reducing $\boldsymbol{t}$ to a subsequence starting with t_1, we can further assume that $(t_j^k)_{j \geq 2}$ is an infinite sequence of pairwise distinct trees for each $k \in [1, n]$. Indeed, if $(t_j^k)_{j \geq 2}$ contains only finitely many distinct trees for some k, then some tree t must occur infinitely often in the sequence $(t_j^k)_{j \geq 2}$, say $t = t_{\ell_2}^k = t_{\ell_3}^k = \dots$ for some $1 = \ell_1 < \ell_2 < \dots$. We then extend c by plugging t into the hole v^k and we replace $\boldsymbol{t}$ with $(t_{\ell_j})_{j \geq 1}$. Clearly, duplicates in a sequence $(t_j^k)_{j \geq 2}$ that contains infinitely many distinct elements can also be removed by restricting to a subsequence.

Now, for all $i < j$ we have

$$t_i \otimes t_j = \begin{cases} (t_1 \otimes c)[\varepsilon \otimes t_j^1, \dots, \varepsilon \otimes t_j^n], & \text{if } 1 = i < j \\ (c \otimes c)[t_i^1 \otimes t_j^1, \dots, t_i^n \otimes t_j^n], & \text{if } 1 < i < j \end{cases}$$

where $t_1 \otimes c$ and $c \otimes c$ are naturally viewed as contexts with n holes. For $j \geq 2$ consider the accepting run ρ_j of $\mathcal{A}$ on $t_1 \otimes t_j$ and the accepting run π_j of $\mathcal{B}$ on t_j and color each index j with the tuple $(\rho_j(v^1), \dots, \rho_j(v^n), \pi_j(v^1), \dots, \pi_j(v^n))$. By the pigeonhole principle, we can pick numbers $1 = \ell_1 < \ell_2 < \dots$ such that $\{\ell_2, \ell_3, \dots\}$ is monochromatic. We then replace $\boldsymbol{t}$ with $(t_{\ell_i})_{i \geq 1}$. Hence, the accepting runs of $\mathcal{A}$ on $t_1 \otimes t_j$ for $j \geq 2$ visit the same states $r^1, \dots, r^n \in Q$ in the nodes $v^1, \dots, v^n$. Similarly, the accepting runs of $\mathcal{B}$ on the trees t_j visit the same states $p^1, \dots, p^n \in P$ in the nodes $v^1, \dots, v^n$. Therefore, for any $j_1, \dots, j_n \geq 2$ we have

$$(t_1, c[t_{j_1}^1, \dots, t_{j_n}^n]) \in R \qquad \text{and} \qquad c[t_{j_1}^1, \dots, t_{j_n}^n] \in L.$$

For all $1 < i < j$ consider the accepting run of $\mathcal{A}$ on $t_i \otimes t_j$ and let $q_{i,j}^k$ be the state reached in node v^k. By Ramsey's theorem, we can assume that there exist states $q^1, \dots, q^n \in Q$ such that $q_{i,j}^k = q^k$ for all $1 < i < j$ (again, after replacing $\boldsymbol{t}$ with a subsequence starting with t_1). Observe that $\mathcal{A}$ accepts the context $c \otimes c$ if it starts in nodes $v^1, \dots, v^k$ with the states $q^1, \dots, q^k$, respectively.

For every $0 \leq k \leq n$ define the tree

$$s_k := c[t_3^1, \dots, t_3^k, t_2^{k+1}, \dots, t_2^n].$$

We have $(s_0, s_n) = (t_2, t_3) \in R$. There must be an index $1 \leq k \leq n$ with $(s_{k-1}, s_k) \in R$ since otherwise by transitivity of $\overline{R}$, we would have $(s_0, s_n) \notin R$. Define the context

$$\beta := c[t_3^1, \dots, t_3^{k-1}, x, t_2^{k+1}, \dots, t_2^n].$$

Then we have that $(\beta[t_2^k], \beta[t_3^k]) \in R$. This is witnessed by the accepting run of $\mathcal{A}$ on the convolution $\beta[t_2^k] \otimes \beta[t_3^k]$, which reaches state q^k at node v^k. This implies that $(\beta[t_i^k], \beta[t_j^k]) \in R$ for all $i < j$ since the run of $\mathcal{A}$ on $t_i^k \otimes t_j^k$ also reaches q^k. Hence, the context β together with the trees t_1 and $\beta[t_i^k]$ for $i \geq 2$ satisfy the claim of the lemma. In particular, $t_1 \lhd \beta$ since $t_1 \lhd c$ and $c \leq_{\mathsf{p}} \beta$. □

Repeated application of Lemma 4.3.10 yields the desired infinite clique:

Lemma 4.3.11. *If a co-transitive tree-regular relation $R \subseteq \mathcal{T}_\Sigma \times \mathcal{T}_\Sigma$ has an infinite clique, then there exists an infinite clique of R that is a comb generated by a monadic generator.*

Proof. We prove the lemma by induction. Let $n \geq 0$ and suppose we have already constructed trees $\alpha_1, \ldots, \alpha_n$ and nontrivial monadic contexts $\beta_1, \ldots, \beta_n$ with $\alpha_i \triangleleft \beta_i$ for all $1 \leq i \leq n$ such that there exist trees $(t'_i)_{i>n}$ such that $(t_i)_{i\geq 1}$ is an infinite clique in R with

$$t_i := \begin{cases} \beta_1\beta_2 \ldots \beta_{i-1}\alpha_i, & \text{if } i \leq n \\ \beta_1\beta_2 \ldots \beta_n t'_i, & \text{if } i > n \end{cases}$$

where β is the trivial context if $n = 0$ and $\beta := \beta_1\beta_2 \ldots \beta_n$ if $n > 0$. Then $(t'_i)_{i>n}$ is an infinite clique in the relation $R' := \{(s,t) \mid (\beta s, \beta t) \in R\}$. It is easy to see that R' is again tree-regular and also co-transitive. Furthermore, all trees t'_i for $i > n$ belong to the tree-regular language $L := \bigcap_{i=1}^{n} \{t \mid (t_i, \beta t) \in R\}$. We can apply Lemma 4.3.10 and obtain a tree α_{n+1}, a nontrivial monadic context β_{n+1} with $\alpha_{n+1} \triangleleft \beta_{n+1}$, and trees $(t''_i)_{i>n+1}$ such that α_{n+1} together with $\beta_{n+1}t''_i$ for $i > n+1$ form an infinite clique in R' over L. Hence, $t_1, \ldots, t_n$ together with $\beta\alpha_{n+1}$ and $\beta\beta_{n+1}t''_i$ for $i > n+1$ form an infinite clique in R. By induction, we then obtain the desired monadic generator $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of an infinite clique in R. □

Using Lemmas 4.3.6 and 4.3.11, we can prove a statement similar to Lemma 4.2.7 for tree combs which are generated by a monadic generator.

Lemma 4.3.12. *If a co-transitive tree-regular relation $R \subseteq \mathcal{T}_\Sigma \times \mathcal{T}_\Sigma$ given as an NTA $\mathcal{A}$ has an infinite clique, then there exist a monadic generator $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of a comb $\boldsymbol{t}$, accepting runs $\rho(t_i, t_j)$ of $\mathcal{A}$ on $t_i \otimes t_j$, and contexts $\kappa_i, \lambda_i, \mu_j, \nu_j$ such that*

$$\rho(t_i, t_j) = \kappa_1 \ldots \kappa_{i-1}\lambda_i\mu_{i+1} \ldots \mu_{j-1}\nu_j$$

is a decomposition compatible with the $(\boldsymbol{\alpha}, \boldsymbol{\beta})$-decomposition of $t_i \otimes t_j$ for all $i < j$.

Proof. Suppose that R has an infinite clique. By Lemmas 4.3.6 and 4.3.11 and since a coarsening of a monadic generator is still monadic, there exist an infinite clique $\boldsymbol{t}$ in R with a monadic generator $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ and context forests $\kappa_i, \lambda_i, \mu_{i,j}, \nu_{i,j}$ for $i < j$ such that $\rho_{i,j} = \kappa_1 \ldots \kappa_{i-1}\lambda_i\mu_{i,i+1} \ldots \mu_{i,j-1}\nu_{i,j}$ is a decomposition compatible with the $(\boldsymbol{\alpha}, \boldsymbol{\beta})$-decomposition of $t_i \otimes t_j$. In particular, all context forests $\kappa_i, \lambda_i, \mu_{i,j}$ have exactly one hole and hence $\kappa_i, \lambda_i, \mu_{i,j}, \nu_{i,j}$ are in fact contexts.

Moreover, we can ensure that $\mu_{i,j} = \mu_{i',j}$ and $\nu_{i,j} = \nu_{i',j}$ for all $i < i' < j$ and can therefore just write μ_j and ν_j for all $j \geq 2$, respectively. For the proof we can reason similarly as in Lemma 4.2.7 by applying Ramsey's theorem to ensure that all contexts $\mu_{i,j}$ carry the same state in the root. □

Lemma 4.3.12 allows us to verify the runs of an infinite clique generated by a monadic generator using a polynomially-sized NBTA.

Theorem 4.3.13. *The infinite clique problem over co-transitive tree-regular relations $R \subseteq (\mathcal{T}_\Sigma)^{2d}$ is* P*-complete.*

Proof. We first show the upper bound. By Proposition 3.1.4, we can assume that $R \subseteq \mathcal{T}_\Sigma \times \mathcal{T}_\Sigma$, i.e. $d = 1$. Let R be given by an NTA $\mathcal{A} = (Q, (\Sigma_\perp)^2, \Delta, q_0)$. We can construct in polynomial time an NBTA $\mathcal{B}$ over the alphabet $\Omega = (\Sigma_\perp)^2 \cup \{\#\}$ which accepts all comb encodings $\mathrm{enc}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of monadic generators $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ for which contexts of the form $\kappa_j, \lambda_j, \mu_j, \nu_j$ as in Lemma 4.3.12 exist. Since the set of all monadic comb encodings can be accepted by an NBTA, we can assume that the input tree is already a valid encoding of a monadic generator. Then the construction of $\mathcal{B}$ is the same as in Proposition 4.3.7. Now, to decide whether R has an infinite clique, it remains to check non-emptiness of $L(\mathcal{B})$, which by Proposition 2.3.24 can be done in polynomial time.

The lower bound already holds for binary co-transitive relations $R \subseteq \mathcal{T}_\Sigma \times \mathcal{T}_\Sigma$ given as D↑TA or D↓TA and follows from a logspace reduction from the non-emptiness problem, which by Proposition 2.3.17 is P-complete for D↑TAs and D↓TAs. Let $L \subseteq \mathcal{T}_\Sigma$ be a tree-regular language given by a D↑TA (resp. D↓TA) $\mathcal{A}$. We define the relation $R \subseteq (\mathcal{T}_\Sigma \times \mathbb{N}) \times (\mathcal{T}_\Sigma \times \mathbb{N})$ such that for all $s, t \in \mathcal{T}_\Sigma$ and $m, n \in \mathbb{N}$ we have $((s, m), (t, n)) \in R$ if and only if $s \in L$. Clearly, R is co-transitive. Moreover, R has an infinite clique if and only if $L \neq \emptyset$. Indeed, if $(t_i, n_i)_{i \geq 1}$ is an infinite clique in R, then $t_1 \in L$ and, conversely, if there exists some $t \in L$, then $(t, i)_{i \geq 1}$ is an infinite clique in R. Note that R can be encoded as a tree-regular relation by representing the natural numbers in unary as paths and a D↑TA (resp. D↓TA) for it can be constructed from $\mathcal{A}$ in logspace. Furthermore, by Proposition 3.1.4, R can be transformed into a binary relation. □

We remark that the above approach using monadic generators can only be used to solve the infinite clique problem. It does not allow us to evaluate the Ramsey quantifier over co-transitive relations $R \subseteq (\mathcal{T}_\Sigma)^{2d+k}$ given as NTAs or D↓TAs with a better complexity than in Theorem 4.3.2. The reason is that in general not every clique contains a subclique that is generated by a monadic generator. In fact, in a comb generated by a monadic generator only one branch can grow unboundedly. Instead, Lemma 4.3.11 only proves existence of a potentially disjoint clique. Thus, the NBTA that accepts monadic comb encodings does not suffice to prove a statement similar to Proposition 4.3.7.

4.4 Applications

We now demonstrate some applications of our results on the evaluation of Ramsey quantifiers in (tree-)regular relations. To this end, we first consider liveness verification in form of recurrent reachability with generalized Büchi condition. As a concrete example with precise complexity results we consider ground tree rewriting systems. Furthermore, we apply our results to obtain precise complexity of the recognizability problem for (tree-)regular relations.

4.4.1 Recurrent Reachability with Generalized Büchi Condition

Let us recall the setting of recurrent reachability in the context of (tree-)regular relations. Since reachability via arbitrary paths in regular relations $R \subseteq A^{2d}$ is in general undecidable [BG04], we instead consider *transitive paths* in R, i.e. infinite sequences $(\boldsymbol{a}_i)_{i \geq 1}$ over

A^d with $(\boldsymbol{a}_i, \boldsymbol{a}_j) \in R$ for all $1 \leq i < j$. Given $R \subseteq A^{2d}$ and $L_1, \ldots, L_k \subseteq A^d$, we write $Rec(L_1, \ldots, L_k)[R]$ for the set of all initial vectors $\boldsymbol{a}_1$ of transitive paths $(\boldsymbol{a}_i)_{i \geq 1}$ in R that visit each L_j infinitely often. *Recurrent reachability with generalized Büchi condition* over (tree-)regular relations is the problem of testing whether $\boldsymbol{a}_1 \in Rec(L_1, \ldots, L_k)[R]$ for given (tree-)regular relations $R \subseteq A^{2d}$ and $L_1, \ldots, L_k \subseteq A^d$ and initial vector $\boldsymbol{a}_1 \in A^d$. If $k = 1$, this problem is simply called *recurrent reachability*.

Corollary 4.4.1. *Recurrent reachability is* NL*-complete over regular relations. It is* EXP*-complete over tree-regular relations given by NTAs or D↓TAs and* P*-complete if the tree-regular relations are transitive or given by D↑TAs.*

The upper bounds follow from Corollary 4.2.2 and Theorems 4.3.1 and 4.3.9 using the reduction to the infinite clique problem (Proposition 3.2.1). The EXP lower bound over tree-regular relations can be shown similarly to EXP-hardness of the infinite clique problem for NTAs and D↓TAs as noted at the end of Section 4.3.1. NL-hardness for regular relations given by DFAs and P-hardness for transitive tree-regular relations given by D↑TAs will be shown later in this section as a special case of the reduction that establishes the lower bound of recurrent reachability with generalized Büchi condition.

We can even compute an automaton for the set $Rec(L)[R]$ of initial vectors from the automata for R and L.

Corollary 4.4.2. *Given NFAs (resp. NTAs) for (tree-)regular relations $R \subseteq A^{2d}$ and $L \subseteq A^d$, one can construct an NFA (resp. NTA) for $Rec(L)[R]$ in logspace (resp. exponential time). For tree-regular relations the construction works in polynomial time if R and L are given by D↑TAs or if R is transitive.*

Proof. We can express $Rec(L)[R]$ by the formula

$$\varphi(\boldsymbol{x}) := \big(\exists^{\mathrm{ram}} \boldsymbol{y}, \boldsymbol{z} \colon R(\boldsymbol{x}, \boldsymbol{y}) \wedge L(\boldsymbol{y}) \wedge R(\boldsymbol{y}, \boldsymbol{z})\big) \vee \big(\exists \boldsymbol{y} \colon R(\boldsymbol{x}, \boldsymbol{y}) \wedge L(\boldsymbol{y}) \wedge R(\boldsymbol{y}, \boldsymbol{y})\big)$$

where $|\boldsymbol{x}| = |\boldsymbol{y}| = |\boldsymbol{z}| = d$. Here, the first disjunct (beginning with $\exists^{\mathrm{ram}}$) captures infinite paths visiting infinitely many configurations, whereas the second disjunct (beginning with $\exists$) captures infinite paths with only finitely many distinct configurations. Thus, we have $[\![\varphi(\boldsymbol{x})]\!] = Rec(L)[R]$.

If R and L are given by NFAs, we can construct in logspace an NFA for $[\![\varphi(\boldsymbol{x})]\!]$ using the closure properties of regular relations (Propositions 2.3.4 and 2.3.27) and Theorem 4.2.1. Over trees we use the closure of tree-regular relations (Propositions 2.3.15 and 2.3.33) and Theorems 4.3.2 and 4.3.9 to construct an NTA for $[\![\varphi(\boldsymbol{x})]\!]$ in exponential or polynomial time depending on whether R is transitive and how R and L are given. Here, note that before we can use the product construction for intersection, we first have to add components such that the vectors of free variables coincide. This does not incur an exponential blow-up in d since we can restrict the additional components to implicitly given alphabets. □

For recurrent reachability with generalized Büchi condition we show that over words the complexity increases from NL to PSPACE, while over trees it stays in EXP.

Theorem 4.4.3. *Recurrent reachability with generalized Büchi condition is* PSPACE*-complete over regular relations and* EXP*-complete over tree-regular relations.*

Again, we show that we can even compute an automaton for the set of initial vectors. We first observe that $\boldsymbol{a}_0 \in \mathit{Rec}(L_1, \dots, L_k)[R]$ if and only if there is a sequence $(\boldsymbol{a}_i)_{i\geq 1}$ such that $(\boldsymbol{a}_i, \boldsymbol{a}_j) \in R$ for all $0 \leq i < j$ and $\boldsymbol{a}_i \in L_{((i-1) \bmod k)+1}$ for all $i \geq 1$. The idea is to define a (tree-)regular relation R' that checks if a tuple $(\boldsymbol{a}_1, \dots, \boldsymbol{a}_{2k})$ forms a clique of size $2k+1$ in R starting with $\boldsymbol{a}_0$ such that $\boldsymbol{a}_i \in L_i$ for all $i \in [1, k]$ and then to use the Ramsey quantifier to check for an infinite clique in R'.

Theorem 4.4.4. *Given NFAs for regular relations $R \subseteq (\Sigma^*)^{2d}$ and $L_1, \dots, L_k \subseteq (\Sigma^*)^d$, one can compute in polynomial space an NFA for $\mathit{Rec}(L_1, \dots, L_k)[R]$.*

Proof. Let $\mathcal{A}$ and $\mathcal{A}_1, \dots, A_k$ be NFAs for R and $L_1, \dots, L_k$, respectively. First observe that for any vector of words $\boldsymbol{w}_0 \in (\Sigma^*)^d$ we have that $\boldsymbol{w}_0 \in \mathit{Rec}(L_1, \dots, L_k)[R]$ if and only if there is a sequence of vectors $(\boldsymbol{w}_i)_{i\geq 1}$ such that $(\boldsymbol{w}_i, \boldsymbol{w}_j) \in R$ for all $0 \leq i < j$ and $\boldsymbol{w}_i \in L_{((i-1) \bmod k)+1}$ for all $i \geq 1$. We define the relation

$$R_i := \big\{(\boldsymbol{u}_1, \dots, \boldsymbol{u}_{2k}, \boldsymbol{u}_0) \in ((\Sigma^*)^d)^{2k+1} \mid \boldsymbol{u}_i \in L_i\big\}$$

for all $i \in [1, k]$. Moreover, for all $1 \leq i < j \leq 2k$ let

$$R_{i,j} := \big\{(\boldsymbol{u}_1, \dots, \boldsymbol{u}_{2k}, \boldsymbol{u}_0) \in ((\Sigma^*)^d)^{2k+1} \mid (\boldsymbol{u}_i, \boldsymbol{u}_j) \in R\big\}.$$

Finally, we define the relation

$$R_{0,i} := \big\{(\boldsymbol{u}_1, \dots, \boldsymbol{u}_{2k}, \boldsymbol{u}_0) \in ((\Sigma^*)^d)^{2k+1} \mid (\boldsymbol{u}_0, \boldsymbol{u}_i) \in R\}$$

for all $i \in [1, k]$. Now, let

$$R' := \bigcap_{i=1}^{k} R_i \cap \bigcap_{1\leq i<j\leq 2k} R_{i,j} \cap \bigcap_{i=1}^{k} R_{0,i} \subseteq (\Sigma^*)^{2kd+d}$$

and

$$\varphi(\boldsymbol{z}) := \big(\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R'(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\big) \vee \big(\exists \boldsymbol{x} \colon R'(\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{z})\big)$$

where $|\boldsymbol{x}| = |\boldsymbol{y}| = kd$ and $|\boldsymbol{z}| = d$. Then $\mathit{Rec}(L_1, \dots, L_k)[R] = [\![\varphi(\boldsymbol{z})]\!]$.

Note that the product automaton $\mathcal{A}'$ for R' can be constructed in polynomial space using Proposition 2.3.4. By Theorem 4.2.1, an NFA for $[\![\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R'(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!]$ can be constructed in logspace given $\mathcal{A}'$. Moreover, by Proposition 2.3.27 (and the fact that an NFA for equality is easily constructible), an NFA for $[\![\exists \boldsymbol{x} \colon R'(\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{z})]\!]$ can also be constructed in logspace given $\mathcal{A}'$. Thus, in total we can construct in polynomial space an NFA for $[\![\varphi(\boldsymbol{z})]\!]$. □

We can use the same proof as in Theorem 4.4.4 also for tree-regular relations. If the relations are given by D↑TAs or if R is transitive, this results in an exponential-time procedure since by Theorems 4.3.2 and 4.3.9, the Ramsey quantifier can be evaluated in polynomial time in these cases and construction of the product automaton still requires polynomial space. For this we observe in the above proof that if R is transitive, then so is R' and if $\mathcal{A}$ and $\mathcal{A}_1, \dots, \mathcal{A}_k$ are D↑TAs, then also $\mathcal{A}'$ is a D↑TA.

Theorem 4.4.5. *Given D↑TAs (or NTAs if R is transitive) for tree-regular relations $R \subseteq (\mathcal{T}_\Sigma)^{2d}$ and $L_1, \dots, L_k \subseteq (\mathcal{T}_\Sigma)^d$, one can compute in exponential time an NTA for $Rec(L_1, \dots, L_k)[R]$.*

Note that if R is non-transitive and the relations are given by NTAs (or ATAs), then Theorem 4.4.5 only gives a double exponential-time algorithm by first converting to D↑TAs with an exponential blow-up using Proposition 2.3.20. However, the corresponding decision problem, i.e. checking whether some given vector of trees belongs to $Rec(L_1, \dots, L_k)[R]$, can be solved in exponential time even if R is non-transitive and the relations are given by ATAs (upper bound of Theorem 4.4.3 in the tree case). Here, we can avoid the exponential blow-up for the product automaton $\mathcal{A}'$ using the efficient closure properties of ATAs (Proposition 2.3.21). To make this work, we have to reduce the size of the alphabet for the ATA $\mathcal{A}'$. This can be achieved by encoding a tuple $(\boldsymbol{a}_1, \dots, \boldsymbol{a}_k)$ of vectors $\boldsymbol{a}_i \in (\Sigma_\perp)^d$ by a path $\boldsymbol{a}_1(\boldsymbol{a}_2(\dots \boldsymbol{a}_k(\#_m)\dots))$, where $\#_m$ is used as delimiter symbol of rank $m := \max\{\mathrm{rk}(\boldsymbol{a}_i) \mid 1 \leq i \leq k\}$. Then the ATA $\mathcal{A}'$ can be constructed in polynomial time.

Lemma 4.4.6. *Recurrent reachability with generalized Büchi condition is decidable in exponential time over tree-regular relations given by ATAs.*

Proof. Let R and $L_1, \dots, L_k$ be given by ATAs $\mathcal{A} = (Q, (\Sigma_\perp)^{2d}, \delta, q_0)$ and $\mathcal{A}_i = (Q_i, (\Sigma_\perp)^d, \delta_i, q_0^i)$ for all $i \in [1, k]$, respectively, and let $\boldsymbol{t}_0 \in (\mathcal{T}_\Sigma)^d$ be some given initial vector of trees. Let $r := \max\{\mathrm{rk}(\boldsymbol{a}) \mid \boldsymbol{a} \in (\Sigma_\perp)^d\}$ and $\Omega := (\Sigma_\perp)^d \cup \{\#_i \mid 0 \leq i \leq r\}$ be a new ranked alphabet with $\mathrm{rk}(\boldsymbol{a}) := 1$ for all $\boldsymbol{a} \in (\Sigma_\perp)^d$ and $\mathrm{rk}(\#_i) := i$ for all $i \in [1, r]$. Let $\otimes'$ be the convolution operation that uses the tuple $(\perp, \dots, \perp)$ of length d as padding symbol. For trees $t_1, \dots, t_n \in \mathcal{T}_{(\Sigma_\perp)^d}$ we define $p(t_1, \dots, t_n) \in \mathcal{T}_\Omega$ to be the tree $t_1 \otimes' \cdots \otimes' t_n$ where each node labeled with $(\boldsymbol{a}_1, \dots, \boldsymbol{a}_n)$ is replaced with a path $\boldsymbol{a}_1(\boldsymbol{a}_2(\dots \boldsymbol{a}_n(\#_m)\dots))$ where $m := \max\{\mathrm{rk}(\boldsymbol{a}_i) \mid 1 \leq i \leq n\}$. Let

$$R_p := \left\{(s,t) \in (\mathcal{T}_\Omega)^2 \mid \exists t_1, \dots, t_{2k} \in \mathcal{T}_{(\Sigma_\perp)^d} \colon s = p(t_1, \dots, t_k) \wedge t = p(t_{k+1}, \dots, t_{2k})\right\}$$

be a binary tree-regular relation that checks if the trees are in the image of p. Note that an ATA for R_p can easily be constructed in logspace (if we restrict to the alphabets implicitly given by $\mathcal{A}, \mathcal{A}_1, \dots, \mathcal{A}_k$) by keeping the maximal rank m of symbols on the current path since the last delimiter symbol in the states and verifying that the next delimiter symbol is $\#_m$. We define ATAs for relations R_i for all $i \in [1, k]$ and $R_{i,j}$ for all $1 \leq i < j \leq 2k$ with a similar meaning as in the word case. We start with the construction of the ATA

$$\mathcal{B}_i = (Q_i^{\mathcal{B}}, (\Omega_\perp)^2, \delta_i^{\mathcal{B}}, (q_0^i, 0))$$

for the binary relation R_i for all $i \in [1, k]$. Intuitively, $\mathcal{B}_i$ checks if in $p(t_1, \dots, t_k) \otimes p(t_{k+1}, \dots, t_{2k})$ we have that t_i is accepted by $\mathcal{A}_i$. The set of states of $\mathcal{B}_i$ is defined as

$$Q_i^{\mathcal{B}} := Q_i \times \{0, \dots, i-1\} \cup Q_i \times (\Sigma_\perp)^d \times \{i, \dots, k\}.$$

For all $q \in Q_i$, $j \in [0,k]$, $\boldsymbol{a}, \boldsymbol{b} \in (\Sigma_\perp)^d \cup \{\perp\}$, and $\boldsymbol{c} \in (\Sigma_\perp)^d$ we let

$$\begin{aligned}
\delta_i^{\mathcal{B}}\big((q,j),(\boldsymbol{a},\boldsymbol{b})\big) &:= \big((q,j+1),1\big), \text{ if } j < i-1\\
\delta_i^{\mathcal{B}}\big((q,i-1),(\boldsymbol{c},\boldsymbol{b})\big) &:= \big((q,\boldsymbol{c},i\big),1)\\
\delta_i^{\mathcal{B}}\big((q,\boldsymbol{c},j),(\boldsymbol{a},\boldsymbol{b})\big) &:= \big((q,\boldsymbol{c},j+1),1\big), \text{ if } i \le j < k
\end{aligned}$$

and for all $m_1, m_2 \in [0,r]$ with $\mathrm{rk}(\boldsymbol{c}) \le \max\{m_1,m_2\}$ let

$$\delta_i^{\mathcal{B}}\big((q,\boldsymbol{c},k),(\#_{m_1},\#_{m_2})\big) := \delta_i'(q,\boldsymbol{c})$$

where $\delta_i'(q,\boldsymbol{c})$ is the formula $\delta_i(q,\boldsymbol{c})$ in which each variable (p,ℓ) is replaced with $((p,0),\ell)$.

We now construct the ATA

$$\mathcal{A}_{i_1,i_2} = (Q_{i_1,i_2}, (\Omega_\perp)^2, \delta_{i_1,i_2}, (q_0,0))$$

for the binary relation R_{i_1,i_2} for all $1 \le i_1 < i_2 \le 2k$. Intuitively, $\mathcal{A}_{i_1,i_2}$ checks if in $p(t_1,\dots,t_k) \otimes p(t_{k+1},\dots,t_{2k})$ we have that $t_{i_1} \otimes' t_{i_2}$ is accepted by $\mathcal{A}$. We only show the construction for the case $1 \le i_1 \le k < i_2 \le 2k$ and $i_1 < i_2 - k$ and note that the other cases are analogous. The state set of $\mathcal{A}_{i_1,i_2}$ is defined as

$$\begin{aligned}
Q_{i_1,i_2} := {} & Q \times \{0,\dots,i_1-1\} \cup\\
& Q \times (\Sigma_\perp)^d \times \{i_1,\dots,i_2-k-1\} \cup\\
& Q \times (\Sigma_\perp)^d \times (\Sigma_\perp)^d \times \{i_2-k,\dots,k\}.
\end{aligned}$$

We now define the transition function. For all $q \in Q$, $j \in [0,k]$, $\boldsymbol{a}, \boldsymbol{b} \in (\Sigma_\perp)^d \cup \{\perp\}$, and $\boldsymbol{c}, \boldsymbol{d} \in (\Sigma_\perp)^d$ we let

$$\begin{aligned}
\delta_{i_1,i_2}\big((q,j),(\boldsymbol{a},\boldsymbol{b})\big) &:= \big((q,j+1),1\big), \text{ if } j < i_1-1\\
\delta_{i_1,i_2}\big((q,i_1-1),(\boldsymbol{c},\boldsymbol{b})\big) &:= \big((q,\boldsymbol{c},i_1),1\big)\\
\delta_{i_1,i_2}\big((q,\boldsymbol{c},j),(\boldsymbol{a},\boldsymbol{b})\big) &:= \big((q,\boldsymbol{c},j+1),1\big), \text{ if } i_1 \le j < i_2-k-1\\
\delta_{i_1,i_2}\big((q,\boldsymbol{c},i_2-k-1),(\boldsymbol{a},\boldsymbol{d})\big) &:= \big((q,\boldsymbol{c},\boldsymbol{d},i_2),1\big)\\
\delta_{i_1,i_2}\big((q,\boldsymbol{c},\boldsymbol{d},j),(\boldsymbol{a},\boldsymbol{b})\big) &:= \big((q,\boldsymbol{c},\boldsymbol{d},j+1),1\big), \text{ if } i_2-k \le j < k
\end{aligned}$$

and for all $m_1, m_2 \in [0,r]$ with $\mathrm{rk}(\boldsymbol{c}), \mathrm{rk}(\boldsymbol{d}) \le \max\{m_1,m_2\}$ let

$$\delta_{i_1,i_2}\big((q,\boldsymbol{c},\boldsymbol{d},k),(\#_{m_1},\#_{m_2})\big) := \delta'\big(q,(\boldsymbol{c},\boldsymbol{d})\big)$$

where $\delta'(q,(\boldsymbol{c},\boldsymbol{d}))$ is the formula $\delta(q,(\boldsymbol{c},\boldsymbol{d}))$ in which each variable (p,ℓ) is replaced with $((p,0),\ell)$.

The ATA for the binary relation $R_{0,i}$ with $i \in [1,k]$ that checks if in $p(t_1,\dots,t_k) \otimes p(t_{k+1},\dots,t_{2k})$ we have that $(t_{0,1} \otimes \cdots \otimes t_{0,d}) \otimes' t_i$ is accepted by $\mathcal{A}$, where $\boldsymbol{t}_0 = (t_{0,1},\dots,t_{0,d})$, can be constructed similarly to $\mathcal{A}_{i,j}$. Note that all the constructions above can be done in logspace (if we restrict to the alphabets implicitly given by $\mathcal{A}, \mathcal{A}_1, \dots, \mathcal{A}_k$).

Now, define

$$R' := R_p \cap \bigcap_{i=1}^{k} R_i \cap \bigcap_{1 \leq i < j \leq 2k} R_{i,j} \cap \bigcap_{i=1}^{k} R_{0,i}.$$

Then it holds that $\boldsymbol{t}_0 \in \mathit{Rec}(L_1, \ldots, L_k)[R]$ if and only if

$$\varphi := \big(\exists^{\mathrm{ram}} x, y \colon R'(x, y)\big) \vee \big(\exists x \colon R'(x, x)\big)$$

is valid. An ATA $\mathcal{A}'$ for R' can be constructed in polynomial time using the efficient closure properties of ATAs (Proposition 2.3.21). Since by Proposition 2.3.17, non-emptiness for NTAs is decidable in polynomial time, we can check validity of the first disjunct of φ using Theorem 4.3.2 and validity of the second disjunct of φ using Proposition 2.3.20 (and the fact that an ATA for equality is easily constructible) in time exponential in the size of $\mathcal{A}'$. This yields an exponential-time algorithm in total. □

For the lower bounds of Theorem 4.4.3 we reduce from the intersection non-emptiness problem of (tree-)regular languages $L_1, \ldots, L_k \subseteq A$, which is PSPACE-complete over words (Proposition 2.3.6) and EXP-complete over trees (Proposition 2.3.18). We define the binary (tree-)regular relation $R \subseteq A' \times A'$ such that $(a, b) \in R$ if and only if $a = c$ or $a = b$, where A' is the set of words (resp. trees) over the alphabet of A expanded with a fresh symbol c (of rank 0). Then $L_1 \cap \cdots \cap L_k \neq \emptyset$ if and only if $c \in \mathit{Rec}(L_1, \ldots, L_k)[R]$. Note that this means that the lower bounds of recurrent reachability with generalized Büchi condition already hold for transitive binary relations R and languages $L_1, \ldots, L_k$ (i.e. for $d = 1$) given by DFAs (resp. D↑TAs or D↓TAs).

For $k = 1$ the previous construction yields a reduction from the non-emptiness problem for DFAs (resp. D↑TAs), which is NL-complete by Proposition 2.3.5 (resp. P-complete by Proposition 2.3.17), to recurrent reachability over transitive (tree-)regular relations given by DFAs (resp. D↑TAs), proving the NL-hardness (resp. P-hardness) in Corollary 4.4.1.

Regular ground tree rewriting Rewriting systems provide finite representations of certain infinite-state transition systems by giving a set of rewriting rules. A well studied class of such systems is the one of prefix rewriting systems, where configurations of the transition system are words and the step relation corresponds to the application of a prefix rewriting rule. For example, any pushdown automaton can be written as a prefix rewriting system [Cau92]. Instead of words, one can also consider rewriting over trees. We define a *ground tree rewriting system* (GTRS) as in [Eng99] as a tuple $\mathcal{R} = (\Sigma, R)$ where Σ is a ranked alphabet and R is a finite set of rewriting rules of the form $u \hookrightarrow v$ with $u, v \in \mathcal{T}_\Sigma$. For trees $s, t \in \mathcal{T}_\Sigma$ we write $s \to_{\mathcal{R}} t$ if there are a rule $u \hookrightarrow v$ in R and a context $c \in \mathcal{C}_\Sigma$ with $|\mathrm{holes}(c)| = 1$ such that $s = c[u]$ and $t = c[v]$. We generalize the notion of a GTRS to a *regular ground tree rewriting system* (RGTRS) by allowing rules of the form $U \hookrightarrow V$ where $U, V \subseteq \mathcal{T}_\Sigma$ are tree-regular languages given by NTAs that abbreviate all the rules $u \hookrightarrow v$ with $u \in U$ and $v \in V$. An RGTRS $\mathcal{R}$ defines a transition system $(\mathcal{T}_\Sigma, \to_{\mathcal{R}})$. As usual, we denote the transitive closure of $\to_{\mathcal{R}}$ by $\to_{\mathcal{R}}^+$ and the reflexive closure of $\to_{\mathcal{R}}^+$ by $\to_{\mathcal{R}}^*$. For a set of trees $S \subseteq \mathcal{T}_\Sigma$ we let

$\mathrm{pre}^*_{\mathcal{R}}(S) := \{s \in \mathcal{T}_\Sigma \mid \exists t \in S\colon s \to^*_{\mathcal{R}} t\}$ and $\mathrm{post}^*_{\mathcal{R}}(S) := \{t \in \mathcal{T}_\Sigma \mid \exists s \in S\colon s \to^*_{\mathcal{R}} t\}$. It was shown by Löding [Löd06] that for any tree-regular S the sets $\mathrm{pre}^*_{\mathcal{R}}(S)$ and $\mathrm{post}^*_{\mathcal{R}}(S)$ are again tree-regular and NTAs for them can be computed in polynomial time given NTAs for S and $\mathcal{R}$. From this it follows that also the reachability relation $\to^+_{\mathcal{R}}$ can be computed in polynomial time (as shown in [LS07] for a more general tree rewrite system that we discuss in Section 4.5). Thus, since $\to^+_{\mathcal{R}}$ is transitive, we can apply Corollaries 4.4.1 and 4.4.2 to decide recurrent reachability for RGTRSs in polynomial time and construct an NTA for the set of initial trees.

Corollary 4.4.7. *Recurrent reachability is* P*-complete for RGTRSs. Moreover, given an RGTRS $\mathcal{R}$ and an NTA for a tree-regular language L, one can construct in polynomial time an NTA for $\mathit{Rec}(L)[\to^+_{\mathcal{R}}]$.*

This was already proven in [Löd06], but by using Theorem 4.4.3, we can also show that recurrent reachability with generalized Büchi condition is EXP-complete for RGTRSs. Here, EXP-hardness already holds for GTRSs, which can be shown by a similar reduction as above from intersection non-emptiness. We can even compute an NTA for the set of initial trees using Theorem 4.4.5 and the fact that $\to^+_{\mathcal{R}}$ is transitive.

Corollary 4.4.8. *Recurrent reachability with generalized Büchi condition is* EXP*-complete for RGTRSs. Moreover, given an RGTRS $\mathcal{R}$ and NTAs for tree-regular languages $L_1, \dots, L_k$, one can construct in exponential time an NTA for $\mathit{Rec}(L_1, \dots, L_k)[\to^+_{\mathcal{R}}]$.*

4.4.2 Recognizability

Recall that a k-ary relation R is (tree-)recognizable if it can be written as $R = \bigcup_{i=1}^n L_{i,1} \times \cdots \times L_{i,k}$ for (tree-)regular languages $L_{i,j}$. In the following we reduce the problem of (tree-)recognizability for (tree-)regular relations to the infinite clique problem over co-transitive (tree-)regular relations.

Corollary 4.4.9. *Given a regular relation $R \subseteq (\Sigma^*)^k$ by a DFA (resp. NFA), it is* NL*-complete (resp.* PSPACE*-complete) to decide whether R is recognizable.*

While over words we can use the generic algorithm to solve the infinite clique problem for general regular relations, in the tree case have to use the specialized algorithm for co-transitive tree-regular relations to obtain the precise complexity.

Corollary 4.4.10. *Given a tree-regular relation $R \subseteq (\mathcal{T}_\Sigma)^k$ by a D↑TA or D↓TA (resp. NTA), it is* P*-complete (resp.* EXP*-complete) to decide whether R is tree-recognizable.*

Let us first prove the upper bounds. As seen in Proposition 3.3.7, we can reduce (tree-)recognizability for (tree-)regular relations $R \subseteq A^k$ to the problem of checking whether the equivalence relation $\approx^R_j$ has finite index for all $j \in [1, n]$. Here, recall that two elements $a, b \in A$ are $\approx^R_j$-equivalent if and only if $\varphi_j(a, b)$ holds, where

$$\varphi_j(x, y) := \forall \boldsymbol{z}\colon R(x \odot_j \boldsymbol{z}) \leftrightarrow R(y \odot_j \boldsymbol{z}).$$

Given an automaton $\mathcal{A}$ for R, we now compute an automaton $\mathcal{A}_{\neg\varphi_j}$ for $[\![\neg\varphi_j]\!]$ using the fact that

$$\neg\varphi_j(x,y) \equiv \exists \boldsymbol{z}\colon \big(R(x \odot_j \boldsymbol{z}) \wedge \neg R(y \odot_j \boldsymbol{z})\big) \vee \big(\neg R(x \odot_j \boldsymbol{z}) \wedge R(y \odot_j \boldsymbol{z})\big).$$

If $\mathcal{A}$ is a DFA (resp. D↑TA or D↓TA), then one can compute $\mathcal{A}_{\neg\varphi_j}$ in logspace (resp. polynomial time) using Propositions 2.3.4 and 2.3.27 (resp. Propositions 2.3.15 and 2.3.33 and Proposition 2.3.16 for D↓TAs). If $\mathcal{A}$ is an NFA (resp. NTA), then this is possible in polynomial space (resp. exponential time) by first determinizing $\mathcal{A}$ using Proposition 2.3.3 (resp. Proposition 2.3.14). Then we apply Corollary 4.2.2 (resp. Theorem 4.3.13) to $\mathcal{A}_{\neg\varphi_j}$ to check in NL (resp. P) for an infinite clique in $[\![\neg\varphi_j]\!]$, which corresponds to $\approx_j^R$ having infinite index. Thus, in total this results in an NL-algorithm for DFAs (since NL = coNL by [Imm88; Sze88]), PSPACE-algorithm for NFAs, P-algorithm for D↑TAs and D↓TAs, and EXP-algorithm for NTAs. Note that in the construction of $\mathcal{A}_{\neg\varphi_j}$ it is sufficient to consider the complement with respect to the implicitly given alphabet as remarked after Propositions 2.3.4 and 2.3.15 since this does not have any effect on whether $[\![\neg\varphi_j]\!]$ has an infinite clique.

We now prove the lower bounds for DFAs, NFAs, D↑TAs, and NTAs by a reduction from the universality problem similar to [Bar+19].

Lemma 4.4.11. *Given a binary regular relation R by an NFA (resp. DFA), it is* PSPACE*-hard (resp.* NL*-hard) to decide whether R is recognizable. Given a binary tree-regular relation R by an NTA (resp. D↑TA), it is* EXP*-hard (resp.* P*-hard) to decide whether R is tree-recognizable.*

Proof. We give a logspace reduction from the universality problem, which by Proposition 2.3.7 is PSPACE-complete for NFAs and NL-complete for DFAs and by Proposition 2.3.19 is EXP-complete for NTAs and P-complete for D↑TAs. To ease notation, we only consider the word case and remark that the tree case is analogous. Recall that the universality problem asks whether for a given regular language $L \subseteq \Sigma^*$ it holds that $L = \Sigma^*$. Let $L \subseteq \Sigma^*$ be a regular language given by an NFA (resp. DFA) $\mathcal{A}$ (where we assume that Σ is the implicitly given alphabet). We define the binary regular relation

$$R_L := \{(u \otimes v, w) \mid u \in L \text{ or } v = w \in \Sigma^*\}.$$

It is easy to construct in logspace an NFA (resp. DFA) that recognizes R_L from $\mathcal{A}$ using Proposition 2.3.4. It remains to show that R_L is recognizable if and only if $L = \Sigma^*$.

If $L = \Sigma^*$, it holds that $R_L = \{(u \otimes v, w) \mid u, v, w \in \Sigma^*\}$, which is clearly recognizable. For the converse assume that there exists $u_0 \in \Sigma^* \setminus L$. Then the intersection of R_L with the recognizable relation $\{(u_0 \otimes v, w) \mid v, w \in \Sigma^*\}$ is the relation $\{(u_0 \otimes v, w) \mid v = w \in \Sigma^*\}$, which is clearly not recognizable. Since recognizable relations are closed under intersection (Proposition 2.3.28), it follows that R_L cannot be recognizable. □

Note that the above reduction does not preserve top-down determinism. Therefore, for D↓TAs we reduce from the emptiness problem instead.

Lemma 4.4.12. *Given a binary tree-regular relation R by a D↓TA, it is* P*-hard to decide whether R is recognizable.*

Proof. We give a logspace reduction from the emptiness problem for D↓TAs, which is P-complete by Proposition 2.3.17. Let $\mathcal{A} = (Q, \Sigma, \Delta, q_0)$ be a D↓TA. We construct a D↓TA $\mathcal{A}' = (Q, (\Gamma_\perp)^2, \Delta', q_0)$ for a binary tree-regular relation R' over $\Gamma := \Sigma \cup \{\#\}$ where $\# \notin \Sigma$ is a fresh symbol of rank 1. We define the transition relation Δ' such that

- $q_0 \xrightarrow{(\#,\#)} q_0$ is in Δ' and
- $q \xrightarrow{(a,a)} (q_1, \ldots, q_r)$ is in Δ' for all $q \xrightarrow{a} (q_1, \ldots, q_r)$ in Δ.

Clearly, $\mathcal{A}'$ can be constructed in logspace from $\mathcal{A}$.

It is easy to see that $R' \subseteq \{(t,t) \mid t \in \mathcal{T}_\Gamma\}$. Moreover, it holds that R' is finite if and only if $L(\mathcal{A}) = \emptyset$. Indeed, if $L(\mathcal{A}) = \emptyset$, then also $R' = \emptyset$ and if there exists $t \in L(\mathcal{A})$, then $(t_n, t_n) \in R'$ for all $n \geq 0$ where t_n is the resulting tree when padding a chain of #-symbols of length n to the root of t. Since every finite tree-regular relation is recognizable and every infinite subrelation of $\{(t,t) \mid t \in \mathcal{T}_\Gamma\}$ is clearly not recognizable, it holds that R' is recognizable if and only if $L(\mathcal{A}) = \emptyset$. □

The following example shows that we can use our decision procedure for recognizability to check whether formulas over certain structures are monadically decomposable.

Example 4.4.13. Recall that a formula is monadically decomposable if it can be written as a Boolean combination of formulas with only one fee variable. Since Presburger arithmetic satisfies the condition of Proposition 3.3.10, we have that a quantifier-free Presburger formula $\varphi(x_1, \ldots, x_k)$ is monadically decomposable in the quantifier-free fragment if and only if the equivalence relation $\approx_j^{[\![\varphi]\!]}$ has finite index for all $j \in [1, k-1]$. Since Presburger arithmetic is an automatic structure (see Example 2.4.9), it then follows from Proposition 3.3.7 that φ is monadically decomposable in the quantifier-free fragment if and only if $[\![\varphi]\!]_\mathfrak{p}$ is recognizable, where $\mathfrak{p}$ is the standard presentation of Presburger arithmetic. Note that a DFA for $[\![\varphi]\!]_\mathfrak{p}$ can be constructed in polynomial space if φ is quantifier-free since we can assume that $\mathfrak{p}$ is given by DFAs and the constructions for Boolean operations in Proposition 2.3.4 preserve determinism. Thus, by applying Corollary 4.4.9, we can check in polynomial space whether φ is monadically decomposable in the quantifier-free fragment of Presburger arithmetic. We will see in Section 5.6 that the precise complexity of this problem is coNP.

With a similar reasoning that uses Corollary 4.4.10, we can show that monadic decomposability in the quantifier-free fragment of Skolem arithmetic, which is a tree-automatic structure, is decidable in exponential time. Here, note that we can assume that the standard presentation of Skolem arithmetic from Example 2.4.9 is given by D↑TAs.

The reasoning in Example 4.4.13 can be generalized as follows.

Corollary 4.4.14. *Let $\mathfrak{A}$ be a fixed (tree-)automatic structure that satisfies the condition of Proposition 3.3.10. Then one can decide in polynomial space (resp. exponential time) whether a quantifier-free formula over $\mathfrak{A}$ is monadically decomposable in the quantifier-free fragment of $\mathfrak{A}$.*

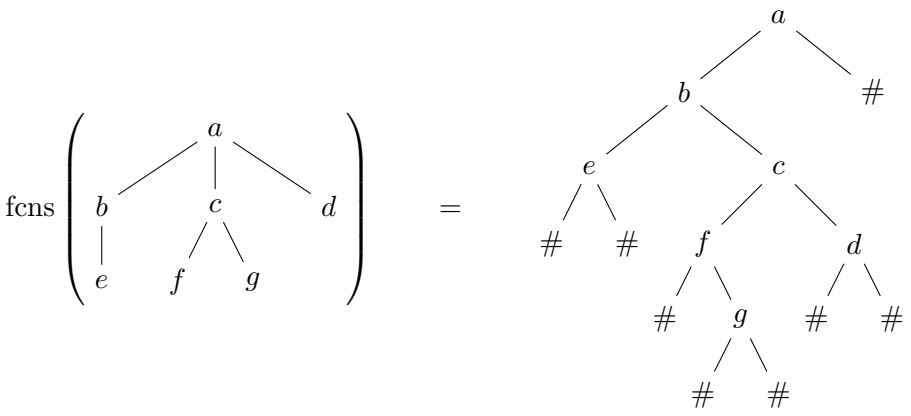


Figure 4.10: An example illustrating the first-child-next-sibling encoding.

4.5 Unranked Tree-Automatic Structures

In this section we consider the unranked tree analogues of Theorems 4.3.2 and 4.3.9. Furthermore, we apply our results to recurrent reachability in subtree and flat prefix rewriting systems.

A *nondeterministic unranked tree automaton* (NUTA) over the (unranked) alphabet Σ is a tuple $\mathcal{A} = (Q, \Sigma, \Delta, q_0)$ where Q is a finite set of states, $q_0 \in Q$ is the initial state, and $\Delta \subseteq Q \times \Sigma \times \mathrm{REG}(Q)$ is a finite set of transitions. Here, $\mathrm{REG}(Q)$ denotes the set of word-regular languages over Q and we assume that the regular language for each transition is given by an NFA without ε-transitions. A *run* of $\mathcal{A}$ on an unranked tree $t \in \mathcal{U}_\Sigma$ is an unranked tree $\rho \in \mathcal{U}_Q$ with $\mathrm{dom}(\rho) = \mathrm{dom}(t)$ such that for each inner node $u \in \mathrm{dom}(\rho)$ with children $u1, \ldots, ur \in \mathrm{dom}(\rho)$ there is a transition $(\rho(u), t(u), L) \in \Delta$ such that $\rho(u1) \ldots \rho(ur) \in L$. A run ρ is *accepting* if $\rho(\varepsilon) = q_0$ and for each leaf $u \in \mathrm{dom}(\rho)$ there is a transition $(\rho(u), t(u), L) \in \Delta$ such that $\varepsilon \in L$.

We define *unranked tree-regular* languages and relations and *unranked tree-automatic structures* analogously to the ranked case by using NUTAs instead of NTAs.

We now define a well-known regularity-preserving transformation from unranked to ranked trees (see e.g. [Com+08; Nev02; Got+05; Lib05]). We define the *first-child-next-sibling encoding* $\mathrm{fcns} : \mathbb{N}^* \to \{1,2\}^*$ such that $\mathrm{fcns}(\varepsilon) = \varepsilon$ and for all $u \in \mathbb{N}^*$ we have $\mathrm{fcns}(u1) = \mathrm{fcns}(u)1$ and $\mathrm{fcns}(u(i+1)) = \mathrm{fcns}(ui)2$ for all $i \geq 1$. For an unranked tree $t \in \mathcal{U}_\Sigma$ we define $t' = \mathrm{fcns}(t) \in \mathcal{T}_{\Sigma_\#}$ to be the binary tree with domain $\mathrm{fcns}(\mathrm{dom}(t)) := \bigcup_{u \in \mathrm{dom}(t)} \{\mathrm{fcns}(u), \mathrm{fcns}(u)1, \mathrm{fcns}(u)2\}$ such that

$$t'(\mathrm{fcns}(u)) := \begin{cases} t(u), & \text{if } u \in \mathrm{dom}(t) \\ \#, & \text{otherwise} \end{cases}$$

for all $u \in \mathbb{N}^*$ with $\mathrm{fcns}(u) \in \mathrm{fcns}(\mathrm{dom}(t))$. Here, we consider $\Sigma_\# := \Sigma \cup \{\#\}$ as a ranked alphabet with $\mathrm{rk}(a) = 2$ for all $a \in \Sigma$ and $\mathrm{rk}(\#) = 0$. The encoding is illustrated in Figure 4.10 on an example. For a relation $R \subseteq (\mathcal{U}_\Sigma)^k$ over unranked trees we write $\mathrm{fcns}(R) := \{(\mathrm{fcns}(t_1), \ldots, \mathrm{fcns}(t_k)) \mid (t_1, \ldots, t_k) \in R\}$.

Lemma 4.5.1. *The encoding* fcns *is a bijection between* $\mathcal{U}_\Sigma$ *and* $\mathrm{fcns}(\mathcal{U}_\Sigma)$.

The first-child-next-sibling encoding effectively preserves tree-regularity.

Proposition 4.5.2. *Given an NUTA for an unranked tree-regular language* $L \subseteq \mathcal{U}_\Sigma$*, one can construct in logspace an NTA for the tree-regular language* $\mathrm{fcns}(L)$*. Conversely, given an NTA for a tree-regular language* $L' \subseteq \mathrm{fcns}(\mathcal{U}_\Sigma)$*, one can construct in logspace an NUTA for the unranked tree-regular language* $\mathrm{fcns}^{-1}(L')$*.*

Proof. Let $\mathcal{A} = (Q, \Sigma, \Delta, q_0)$ be an NUTA with transition NFAs $\mathcal{A}_i = (P_i, Q, \Delta_i, p_0^i, F_i)$ for $i \in [1, m]$. We construct the NTA $\mathcal{A}' := (Q', \Sigma_\#, \Delta', q_0')$ where $Q' := Q \times [1, m] \times P_i \cup \{q_0', q_\#'\}$ and

- $q_0' \xrightarrow{a} \big((q, j, p_0^j), q_\#\big)$ is in Δ' for all $(q_0, a, L(\mathcal{A}_j)) \in \Delta$ and $q \in Q$,
- $(q_\#, \#)$ is in Δ',
- $(q, i, p) \xrightarrow{a} \big((q_1, j, p_0^j), (q_2, i, p_1)\big)$ is in Δ' for all $(q, a, L(\mathcal{A}_j)) \in \Delta$, $i \in [1, m]$, $(p, q, p_1) \in \Delta_i$, and $q_1, q_2 \in Q$, and
- $\big((q, i, p), \#\big)$ is in Δ' for all $i \in [1, m]$, $p \in F_i$, and $q \in Q$.

Intuitively, when going to the right, $\mathcal{A}'$ simulates the current transition NFA in its third component on the guessed state in the first component. When going to the left, $\mathcal{A}'$ starts a new simulation of a transition NFA. It is easy to verify that $L(\mathcal{A}') = \mathrm{fcns}(L(\mathcal{A}))$.

For the converse direction, let $\mathcal{A}' = (Q', \Sigma_\#, \Delta', q_0')$ be an NTA with $L(\mathcal{A}') \subseteq \mathrm{fcns}(\mathcal{U}_\Sigma)$. We construct the NUTA $\mathcal{A} := (Q, \Sigma, \Delta, q_0)$ where $Q := \Sigma \times Q' \cup \{q_0\}$ and Δ contains the transitions

- $\big(q_0, a, L(\mathcal{A}_{q_1})\big)$ for all $(q_0, a, q_1, q_2) \in \Delta'$ with $(q_2, \#) \in \Delta'$ and
- $\big((a, q), a, L(\mathcal{A}_q)\big)$ for all $(a, q) \in \Sigma \times Q'$.

Here, for all $q \in Q'$ we let $\mathcal{A}_q := (Q', \Sigma \times Q', \Delta_q, q, F_q)$ where Δ_q contains $p \xrightarrow{(a,p_1)} p_2$ for all $(p, a, p_1, p_2) \in \Delta'$ and $F_q := \{p \mid (p, \#) \in \Delta'\}$. This means that all transition NFAs of $\mathcal{A}$ have the same shape and only differ in their initial state. Intuitively, a transition NFA simulates $\mathcal{A}'$ on a whole all-right path. □

Unfortunately, for an unranked tree-regular relation R it does not hold in general that $\mathrm{fcns}(\otimes R) = \otimes \mathrm{fcns}(R)$. For example, consider the second row of Figure 4.11, which is not the same as first applying the convolution followed by fcns. This means that we cannot just apply Proposition 4.5.2 to an NUTA for an unranked tree-regular relation R to get an NTA for the relation $\mathrm{fcns}(R)$. However, with some small adaptions to the convolution and first-child-next-sibling encoding, we can show that an effective transformation from unranked tree-regular relations to tree-regular relations and vice versa is possible.

Let $\Sigma_{\#,\perp} := \Sigma \cup \{\#, \perp\}$ for an alphabet Σ with $\#, \perp \notin \Sigma$. For unranked trees $t_1, t_2 \in \mathcal{U}_\Sigma$ we define the *adapted convolution* $t' = t_1 \otimes' t_2 \in \mathcal{U}_{\Sigma_{\#,\perp} \times \Sigma_{\#,\perp}}$ such that $\mathrm{dom}(t') = \mathrm{dom}(t_1) \cup \mathrm{dom}(t_2)$ and for all $u \in \mathrm{dom}(t')$ we let

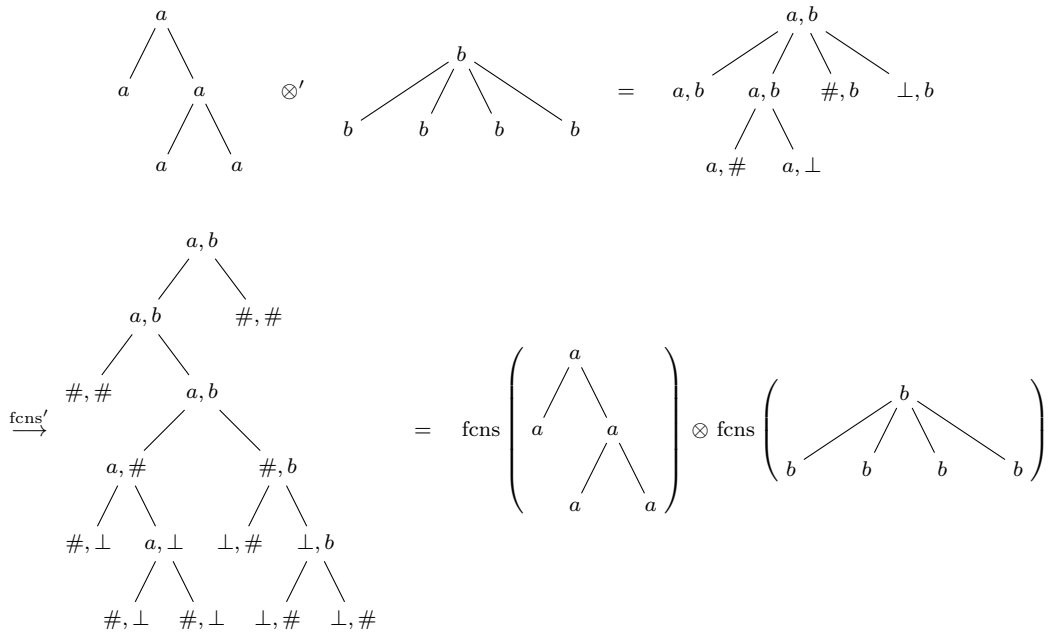


Figure 4.11: An example illustrating adapted convolution and adapted first-child-next-sibling encoding.

- $t'(u) := (t_1(u), t_2(u))$ if $u \in \mathrm{dom}(t_1) \cap \mathrm{dom}(t_2)$,
- $t'(u) := (t_1(u), \#)$ if $u \in \mathrm{dom}(t_1) \setminus \mathrm{dom}(t_2)$ and there exists $v \in \mathrm{dom}(t_2)$ such that u is the first child or right sibling of v,
- $t'(u) := (t_1(u), \bot)$ if $u \in \mathrm{dom}(t_1) \setminus \mathrm{dom}(t_2)$ and the above conditions do not hold,
- the other cases are symmetric.

This means that instead of always padding with $\bot$, the adapted convolution pads a component with $\#$ if it would still be ongoing after the first-child-next-sibling encoding. As for the standard convolution, we write $\otimes' R := \{s \otimes' t \mid (s,t) \in R\}$ for a relation $R \subseteq \mathcal{U}_\Sigma \times \mathcal{U}_\Sigma$.

For $t = t_1 \otimes' t_2 \in \mathcal{U}_{\Sigma_{\#,\bot} \times \Sigma_{\#,\bot}}$ we define the *adapted first-child-next-sibling encoding* $t' = \mathrm{fcns}'(t) \in \mathcal{T}_{\Sigma_{\#,\bot} \times \Sigma_{\#,\bot}}$ such that $\mathrm{dom}(t') = \mathrm{fcns}(\mathrm{dom}(t))$ and for all $u' = \mathrm{fcns}(u) \in \mathrm{dom}(t')$ we let

- $t'(u') := t(u)$ if $u \in \mathrm{dom}(t)$,
- $t'(u') := (\#, \#)$ if $u \notin \mathrm{dom}(t)$ and for parent $\mathrm{fcns}(v)$ of u' we have $t(v) \in \Sigma \times \Sigma$,
- $t'(u') := (\bot, \#)$ if $u \notin \mathrm{dom}(t)$ and for parent $\mathrm{fcns}(v)$ of u' we have $t(v) \in \{\#, \bot\} \times \Sigma$,
- $t'(u') := (\#, \bot)$ if $u \notin \mathrm{dom}(t)$ and for parent $\mathrm{fcns}(v)$ of u' we have $t(v) \in \Sigma \times \{\#, \bot\}$.

Here, we regard $\Sigma_{\#,\perp}$ as ranked alphabet with $\mathrm{rk}(a) = 2$ for all $a \in \Sigma$ and $\mathrm{rk}(\#) = \mathrm{rk}(\perp) = 0$. See Figure 4.11 for an example of the adapted convolution and encoding. Clearly, both $\otimes'$ and fcns' are injective. The next lemma shows that the connection between the adapted and classical notions of convolution and encoding suggested by Figure 4.11 holds true in general.

Lemma 4.5.3. *For all unranked trees $t_1, t_2 \in \mathcal{U}_\Sigma$ it holds that*

$$\mathrm{fcns}'(t_1 \otimes' t_2) = \mathrm{fcns}(t_1) \otimes \mathrm{fcns}(t_2).$$

Proof. The definitions of $\otimes'$ and fcns' ensure that the padding symbol $\#$ is used if the node would also be padded by fcns and otherwise the padding symbol $\perp$ is used. The result follows since nodes at the same position in t_1 and t_2 are mapped to the same position in the encodings $\mathrm{fcns}(t_1)$ and $\mathrm{fcns}(t_2)$. □

Next, we show that the adapted convolution and encoding can still be efficiently computed.

Lemma 4.5.4. *Given an NUTA for a relation $R \subseteq \mathcal{U}_\Sigma \times \mathcal{U}_\Sigma$, one can compute in logspace an NTA for the relation* $\mathrm{fcns}(R)$ *and vice versa.*

Proof. By Lemma 4.5.3, it suffices to show that $\otimes'$ and fcns' and their inverses can be computed in logspace. We start with the adapted convolution. Let $\mathcal{A}$ be an NUTA accepting $\otimes R$. We construct an NUTA $\mathcal{A}'$ accepting $\otimes' R$ from $\mathcal{A}$ such that the conditions in the definition of $\otimes'$ are satisfied. To this end, a state stores for each of the two components if it is the first child or right sibling (by guessing) of a node where the respective component is labeled with a symbol from Σ. Conversely, if $\mathcal{A}'$ is an NUTA accepting $\otimes' R$, we can construct an NUTA accepting $\otimes R$ by simply replacing $\#$ in the transitions of $\mathcal{A}'$ with $\perp$.

To construct an NTA accepting $\mathrm{fcns}'(\otimes' R)$ from an NUTA accepting $\otimes' R$, we use the same construction as in Proposition 4.5.2 for fcns, but we additionally store in states if the label of the parent of the current node is in $\Sigma \times \Sigma$, $\{\perp, \#\} \times \Sigma$, or $\Sigma \times \{\perp, \#\}$ and apply the padding with $\#$ and $\perp$ according to the definition of fcns'. The converse direction works analogously. □

Note that we can generalize the above constructions from the binary to the n-ary case such that the same statements hold. Here, we remark that we can again restrict to implicitly given alphabets to obtain the desired complexity results.

Lemma 4.5.4 implies that unranked tree-regular relations enjoy the same effective closure properties as tree-regular relations and the non-emptiness problem can be solved in polynomial time. Therefore, the first-order theory of every unranked tree-automatic structure is decidable for the same reason as over ranked trees. The following theorem proves that the Ramsey quantifier can be evaluated over unranked tree-regular relations with the same complexity as in the ranked case.

Theorem 4.5.5. *Given an unranked tree-regular relation $R \subseteq (\mathcal{U}_\Sigma)^{2d+k}$ by an NUTA $\mathcal{A}$, an NUTA for $[\![\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!]$ can be constructed in polynomial time if R is transitive and in exponential time otherwise. In particular, the infinite clique problem over (transitive) unranked tree-regular relations is* EXP*-complete (resp.* P*-complete).*

Proof. The lower bounds for the infinite clique problem directly follow from the ranked case since an NTA can be seen as a special NUTA where the transition NFAs only accept one sequence of states.

For the upper bounds we first compute an NTA $\mathcal{A}'$ for the relation $R' = \mathrm{fcns}(R)$ using Lemma 4.5.4. By Theorem 4.3.2, we can construct an NTA $\mathcal{B}'$ for the relation $[\![\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R'(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!]$ in exponential time given $\mathcal{A}'$. If R and therefore also R' are transitive, Theorem 4.3.9 implies that $\mathcal{B}'$ can be computed in polynomial time. Applying the reverse direction of Lemma 4.5.4, we can compute an NUTA for

$$\mathrm{fcns}^{-1}([\![\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R'(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!]) = [\![\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!]$$

in logspace given $\mathcal{B}'$, where the equality holds since fcns is a bijection. □

As in the ranked case, we can reformulate Theorem 4.5.5 for unranked tree-automatic structures.

Corollary 4.5.6. *Given an unranked tree-automatic structure $\mathfrak{A}$ by a presentation $\mathfrak{p}$ and a formula $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ over $\mathfrak{A}$ with $|\boldsymbol{x}| = |\boldsymbol{y}|$, one can effectively compute an NUTA for the relation $[\![\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]\!]_{\mathfrak{p}}$. Moreover, if an NUTA for $[\![\varphi]\!]_{\mathfrak{p}}$ is given, the construction can be done in exponential time.*

Recurrent reachability We can also use the first-child-next-sibling encoding and Corollaries 4.4.1 and 4.4.2 to obtain precise complexity for recurrent reachability.

Corollary 4.5.7. *Recurrent reachability over (transitive) unranked tree-regular relations is* EXP*-complete (resp.* P*-complete). Moreover, given NUTAs for unranked tree-regular relations $R \subseteq (\mathcal{U}_\Sigma)^{2d}$ and $L \subseteq (\mathcal{U}_\Sigma)^d$, one can compute an NUTA for $\mathit{Rec}(L)[R]$ in polynomial time if R is transitive and in exponential time otherwise.*

Using Theorems 4.4.3 and 4.4.5 we further obtain the corresponding results on recurrent reachability with generalized Büchi condition.

Corollary 4.5.8. *Recurrent reachability with generalized Büchi condition over unranked tree-regular relations is* EXP*-complete. Moreover, given NUTAs for unranked tree-regular relations $R \subseteq (\mathcal{U}_\Sigma)^{2d}$ and $L_1, \dots, L_k \subseteq (\mathcal{U}_\Sigma)^d$, one can compute an NUTA for $\mathit{Rec}(L_1, \dots, L_k)[R]$ in exponential time if R is transitive.*

Subtree and flat prefix rewriting systems Löding and Spelten [LS07] introduced tree rewriting systems over unranked trees that generalize the ground tree rewriting systems that we have seen in Section 4.4. Again, we can apply our results on recurrent reachability to those systems.

For a tree $t \in \mathcal{U}_\Sigma$ with $\mathrm{ht}(t) = 1$ we denote the sequence of leaves of t read from left to right by $\mathrm{flatfront}(t)$. Here, $\mathrm{ht}(t) := \max\{|x| \mid x \in \mathrm{dom}(t)\}$ is the height of t.

A *subtree and flat prefix rewriting system* (SFPRS) over unranked trees in $\mathcal{U}_\Sigma$ is a tuple $\mathcal{R} = (\Sigma, R)$ where Σ is an unranked alphabet and R is a finite set of rules of two types:

1) subtree substitution of the form $r_i \colon u_i \hookrightarrow v_i$ for $i \in I$ and $u_i, v_i \in \mathcal{U}_\Sigma$ and

2) flat prefix substitution of the form $r_j \colon u_j \hookrightarrow v_j$ for $j \in J$ and $u_j, v_j \in \Sigma^+$

with $I \cup J = \{1, \ldots, |R|\}$ and $I \cap J = \emptyset$. For trees $s, t \in \mathcal{U}_\Sigma$ we write $s \to_\mathcal{R} t$ if

1) t is derived from s by applying a subtree substitution rule r_i for some $i \in I$, i.e. there is a context $c \in \mathcal{C}_\Sigma$ with $|\mathrm{holes}(c)| = 1$ such that $s = c[u_i]$ and $t = c[v_i]$ or

2) t is derived from s by applying a flat prefix substitution rule r_j for some $j \in J$, i.e. there is a context $c \in \mathcal{C}_\Sigma$ with $|\mathrm{holes}(c)| = 1$ and trees $s', t' \in \mathcal{U}_\Sigma$ with $\mathrm{ht}(s') = \mathrm{ht}(t') = 1$ and $s'(\varepsilon) = t'(\varepsilon)$ such that $s = c[s']$, $t = c[t']$, $\mathrm{flatfront}(s') = u_j w$, and $\mathrm{flatfront}(t') = v_j w$ for some $w \in \Sigma^*$.

Intuitively, a flat prefix substitution replaces a matching prefix on the flat front of a tree with another word.

The definition of an SFPRS can be extended to a *regular SFPRS* by allowing subtree substitution rules of the form $U_i \hookrightarrow V_i$ with unranked tree-regular languages $U_i, V_i \subseteq \mathcal{U}_\Sigma$ and flat prefix substitution rules of the form $U_j \hookrightarrow V_j$ with regular languages $U_j, V_j \subseteq \Sigma^*$. Here, we assume that the sets are given by NUTAs and NFAs. Clearly, SFPRSs are special regular SFPRSs where the rules only consist of singleton sets. An SFPRS $\mathcal{R}$ defines a transition system $(\mathcal{U}_\Sigma, \to_\mathcal{R})$. As usual, we denote the transitive closure of $\to_\mathcal{R}$ by $\to_\mathcal{R}^+$ and the reflexive closure of $\to_\mathcal{R}^+$ by $\to_\mathcal{R}^*$. For a set of trees $S \subseteq \mathcal{U}_\Sigma$ we let $\mathrm{pre}^*_\mathcal{R}(S) := \{s \in \mathcal{U}_\Sigma \mid \exists t \in S \colon s \to_\mathcal{R}^* t\}$ and $\mathrm{post}^*_\mathcal{R}(S) := \{t \in \mathcal{U}_\Sigma \mid \exists s \in S \colon s \to_\mathcal{R}^* t\}$. It was shown by Löding and Spelten [LS07] that for any unranked tree-regular S the sets $\mathrm{pre}^*_\mathcal{R}(S)$ and $\mathrm{post}^*_\mathcal{R}(S)$ are again unranked tree-regular. Moreover, it can be observed that NUTAs for them can be computed in polynomial time given NUTAs for S and $\mathcal{R}$. From this it follows that also an NUTA for the reachability relation $\to_\mathcal{R}^+$ can be computed in polynomial time. Thus, since $\to_\mathcal{R}^+$ is transitive, we can apply Corollary 4.5.7 to decide recurrent reachability for regular SFPRSs in polynomial time and construct an NUTA for the set of initial trees.

Corollary 4.5.9. *Recurrent reachability is* P*-complete for (regular) SFPRSs. Moreover, given a regular SFPRS and an NUTA for an unranked tree-regular language L, one can construct in polynomial time an NUTA for $\mathrm{Rec}(L)[\to_\mathcal{R}^+]$.*

Here, the lower bound follows since recurrent reachability is already P-hard for GTRSs. Using Corollary 4.5.8, we further obtain the precise complexity of recurrent reachability with generalized Büchi condition for regular SFPRSs.

Corollary 4.5.10. *Recurrent reachability with generalized Büchi condition is* EXP*-complete for (regular) SFPRSs. Moreover, given a regular SFPRS and an NUTAs for unranked tree-regular languages $L_1, \dots, L_k$, one can construct in exponential time an NUTA for $\mathrm{Rec}(L_1, \dots, L_k)[\to_{\mathcal{R}}^+]$.*

4.6 Beyond 2-Ramsey Quantifiers

In Section 3.1 we defined a more general version of the Ramsey quantifier. Recall that for a structure $\mathfrak{A}$ with domain A the *n-Ramsey quantifier* $\exists^{n\text{-ram}}$ with $n \geq 2$ of dimension d is defined such that $\mathfrak{A} \models \exists^{n\text{-ram}} \boldsymbol{x}_1, \dots, \boldsymbol{x}_n \colon \varphi(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n, \boldsymbol{c})$ for some $\boldsymbol{c} \in A^{|\boldsymbol{z}|}$ if and only if there exists an infinite sequence $(\boldsymbol{a}_i)_{i \geq 1}$ of pairwise distinct tuples $\boldsymbol{a}_i \in A^d$ such that $\mathfrak{A} \models \varphi(\boldsymbol{a}_{i_1}, \dots, \boldsymbol{a}_{i_n}, \boldsymbol{c})$ for all $1 \leq i_1 < \cdots < i_n$. This means that instead of an infinite clique in a graph, $\exists^{n\text{-ram}}$ states the existence of a so-called infinite (directed) *n-clique* in an n-hypergraph, i.e. a graph where every edge connects n vertices. The *infinite n-clique problem* asks whether $\mathfrak{A} \models \exists^{n\text{-ram}} \boldsymbol{x}_1, \dots, \boldsymbol{x}_n \colon \varphi(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$ for a given formula $\varphi(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$ over some fixed structure $\mathfrak{A}$, where n is part of the input. In the *parameterized* version of this problem, n is considered a parameter. In the following we explore how the above results on Ramsey quantifiers over (tree-)automatic structures change if we consider n-Ramsey quantifiers instead. We will see that the proof strategies can be adapted to the more general notion. To this end, we need a more general version of Ramsey's theorem that can easily be proven by induction.

Theorem 4.6.1 (Ramsey's theorem for hypergraphs)**.** *Any complete infinite undirected n-hypergraph whose edges are colored with finitely many colors contains an infinite monochromatic n-clique.*

4.6.1 Word-Automatic Structures

For a word w let $w^{(k)}$ be the k-fold convolution of w with itself, where we assume that for words $w_1, \dots, w_m$ and integers $k_1, \dots, k_m \geq 0$ we have that $w_1^{(k_1)} \otimes \cdots \otimes w_m^{(k_m)}$ is a convolution of $k_1 + \cdots + k_m$ words rather than m words using parentheses. Let $\boldsymbol{v}$ be a comb generated by $(\boldsymbol{\alpha}, \boldsymbol{\beta})$. For $n \in \mathbb{N}$ and $1 \leq i_1 < \cdots < i_n$ we define the *$(\boldsymbol{\alpha}, \boldsymbol{\beta})$-decomposition* of the convolution $v_{i_1} \otimes \cdots \otimes v_{i_n}$ as

$$v_{i_1} \otimes \cdots \otimes v_{i_n} = u_1 \dots u_n$$

with

$$u_j := (\varepsilon^{(j-1)} \otimes \beta_{i_{j-1}+1}^{(n-j+1)}) \dots (\varepsilon^{(j-1)} \otimes \beta_{i_j-1}^{(n-j+1)})(\varepsilon^{(j-1)} \otimes \alpha_{i_j} \otimes \beta_{i_j}^{(n-j)})$$

for all $j \in [1, n]$ where $i_0 := 0$. That is, u_j contains in its j-th component the suffix of v_{i_j} after $v_{i_1}, \dots, v_{i_{j-1}}$ have ended. So for $n = 2$ the definition coincides with the one in Section 4.2.

To ease notation, for $n, m \in \mathbb{N}$ we write

$$K^n_{<m} = \{\kappa_{i_1,\dots,i_j} \mid j \in [1, n] \wedge 1 \leq i_1 < \cdots < i_j < m\}$$

for a set of words and let $K^d := \bigcup_{m \in \mathbb{N}} K^n_{<m}$. Similarly, we define $\Lambda^n_{<m}$ and Λ^n where elements are named by $\lambda_{i_1,\dots,i_j}$. For $j \in [0,n]$ and $1 \leq i_1 < \cdots < i_j$ we let $\pi_{i_1,\dots,i_j} := \psi_1 \dots \psi_j$ be a decomposition with

$$\psi_\ell := \kappa_{i_1,\dots,i_{\ell-1},i_{\ell-1}+1} \dots \kappa_{i_1,\dots,i_{\ell-1},i_\ell-1} \lambda_{i_1,\dots,i_\ell}$$

for all $\ell \in [1,j]$ where $i_0 := 0$ and $\pi := \varepsilon$. Intuitively, ψ_ℓ will be a run on u_ℓ as defined above. Recall that a decomposition of a run $\rho = \rho_1 \dots \rho_k$ of an NFA is *compatible* with a decomposition $w = u_1 \dots u_k$ of a word if ρ_i is a run on u_i for all $i \in [1,k]$.

Lemma 4.6.2. *If $\boldsymbol{w}$ is an infinite n-clique in a regular relation $R \subseteq (\Sigma^*)^n$ given as an NFA $\mathcal{A}$, then there exist a generator $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of a subsequence $\boldsymbol{v}$ of $\boldsymbol{w}$, accepting runs $\rho(v_{i_1}, \dots, v_{i_n})$ of $\mathcal{A}$ on $v_{i_1} \otimes \cdots \otimes v_{i_n}$, and sets of runs K^d, Λ^d such that*

$$\rho(v_{i_1}, \dots, v_{i_n}) = \pi_{i_1,\dots,i_n}$$

is a decomposition compatible with the $(\boldsymbol{\alpha}, \boldsymbol{\beta})$-decomposition of $v_{i_1} \otimes \cdots \otimes v_{i_n}$ for all $1 \leq i_1 < \cdots < i_n$.

Proof. The proof is analogous to the proof of Lemma 4.2.6. Let $\boldsymbol{w}$ be an infinite n-clique in R and $\rho(w_{i_1}, \dots, w_{i_n})$ be an accepting run of $\mathcal{A}$ on $w_{i_1} \otimes \cdots \otimes w_{i_n}$ for all $1 \leq i_1 < \cdots < i_n$. By Lemma 4.2.5, we can assume that $\boldsymbol{w}$ is a comb. We establish a run structure similar to the one depicted in Figure 4.3, but instead of a comb of combs (which corresponds to $n = 2$), we get an n-fold nested comb structure.

Assume that we already defined a subcomb $\boldsymbol{v}$ of $\boldsymbol{w}$ with generator $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ and sets of runs $K^n_{<m}$ and $\Lambda^n_{<m}$ for some $m \geq 1$ such that

$$\rho(v_{i_1}, \dots, v_{i_n}) = \pi_{\boldsymbol{i}} \kappa_{\boldsymbol{i}, i_j+1} \dots \kappa_{\boldsymbol{i}, m-1} \sigma(v_{i_1}, \dots, v_{i_n})$$

for runs $\sigma(v_{i_1}, \dots, v_{i_n})$ for all

$$1 \leq i_1 < \cdots < i_j < m \leq i_{j+1} < \cdots < i_n$$

and $\boldsymbol{i} := (i_1, \dots, i_j)$ with $j \in [0,n]$ and $i_0 := 0$.

We now define $\kappa_{i_1,\dots,i_j,m}$ successively for all $1 \leq i_1 < \cdots < i_j < m$ and $j \in [0, n-1]$. For each choice of $i_1, \dots, i_j$ we apply Ramsey's theorem for $(n-j)$-hypergraphs or pigeonhole principle if $j = n-1$ to get an infinite subsequence $\boldsymbol{u}$ of $\boldsymbol{v}$ starting with $v_1, \dots, v_m$ such that all runs $\sigma(u_{i_1}, \dots, u_{i_n})$ for $m < i_{j+1} < \cdots < i_n$ have a common prefix $\kappa_{i_1,\dots,i_j,m}$ which is a run on $\varepsilon^{(j)} \otimes \beta_m^{(n-j)}$. Then we replace $\boldsymbol{v}$ with $\boldsymbol{u}$ and $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ with the coarsening defined by $\boldsymbol{u}$.

Next we define $\lambda_{i_1,\dots,i_j,m}$ successively for all $1 \leq i_1 < \cdots < i_j < m$ and $j \in [0, n-2]$. For each choice of $i_1, \dots, i_j$ we apply Ramsey's theorem for $(n-j-1)$-hypergraphs or pigeonhole principle if $j = n-2$ to get an infinite subsequence $\boldsymbol{u}$ of $\boldsymbol{v}$ starting with $v_1, \dots, v_m$ such that all runs $\sigma(u_{i_1}, \dots, u_{i_n})$ for $m = i_{j+1} < \cdots < i_n$ have a common prefix $\lambda_{i_1,\dots,i_j,m}$ which is a run on $\varepsilon^{(j)} \otimes \alpha_m \otimes \beta_m^{(n-j-1)}$. Then we replace $\boldsymbol{v}$ with $\boldsymbol{u}$ and $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ with the coarsening defined by $\boldsymbol{u}$. Finally, we define $\lambda_{i_1,\dots,i_{n-1},m}$ for all $1 \leq i_1 < \cdots < i_{n-1} < m$ as $\sigma(v_{i_1}, \dots, v_{i_{n-1}}, v_n)$.

In the limit we obtain a comb $\boldsymbol{v}$ with generator $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ that is a subsequence of $\boldsymbol{w}$ and the desired decomposition of the runs $\rho(v_{i_1}, \dots, v_{i_n})$ that is compatible with the $(\boldsymbol{\alpha}, \boldsymbol{\beta})$-decomposition of $v_{i_1} \otimes \cdots \otimes v_{i_n}$ for all $1 \leq i_1 < \cdots < i_n$. □

We show that the runs $\kappa_{i_1,\dots,i_j}$ and $\lambda_{i_1,\dots,i_j}$ can be chosen such that they only depend on j and i_j. To this end, for all $n \in \mathbb{N}$ we write $\tilde{K}^n = \{\tilde{\kappa}_i^j \mid j \in [1,n] \wedge i \geq j\}$ and $\tilde{\Lambda}^n = \{\tilde{\lambda}_i^j \mid j \in [1,n] \wedge i \geq j\}$ for sets of words. For $j \in [1,n]$ and $1 \leq i_1 < \cdots < i_j$ let $\tilde{\pi}_{i_1,\dots,i_j} := \tilde{\psi}_1 \dots \tilde{\psi}_j$ with

$$\tilde{\psi}_\ell := \tilde{\kappa}^\ell_{i_{\ell-1}+1} \dots \tilde{\kappa}^\ell_{i_\ell - 1} \tilde{\lambda}^\ell_{i_\ell}$$

for all $\ell \in [1,j]$ where $i_0 := 0$.

Lemma 4.6.3. *If $\boldsymbol{w}$ is an infinite d-clique in a regular relation $R \subseteq (\Sigma^*)^n$ given as an NFA $\mathcal{A}$, then there exist a generator $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ for a subsequence $\boldsymbol{v}$ of $\boldsymbol{w}$, accepting runs $\rho(v_{i_1}, \dots, v_{i_n})$ of $\mathcal{A}$ on $v_{i_1} \otimes \cdots \otimes v_{i_n}$, and sets of runs $\tilde{K}^n, \tilde{\Lambda}^n$ such that*

$$\rho(v_{i_1}, \dots, v_{i_n}) = \tilde{\pi}_{i_1,\dots,i_n}$$

is a decomposition compatible with the $(\boldsymbol{\alpha}, \boldsymbol{\beta})$-decomposition of $v_{i_1} \otimes \cdots \otimes v_{i_n}$ for all $1 \leq i_1 < \cdots < i_n$.

Proof. We generalize the proof of Lemma 4.2.7. Suppose that R has an infinite n-clique. Then there exist an infinite n-clique $\boldsymbol{w}$ in R generated by $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ and sets of runs K^n, Λ^n as in Lemma 4.6.2.

It remains to ensure that for any $j \in [1,n]$ we have $\kappa_{i_1,\dots,i_j} = \kappa_{i'_1,\dots,i'_{j-1},i_j}$ and $\lambda_{i_1,\dots,i_j} = \lambda_{i'_1,\dots,i'_{j-1},i_j}$ for all $1 \leq i_1 < \cdots < i_j$ and $1 \leq i'_1 < \cdots < i'_{j-1} < i_j$. To this end, consider the initial state of the run $\kappa_{i_1,\dots,i_{j-1},i_j+1}$ for $j \in [2,n]$. By Ramsey's theorem for j-hypergraphs, there exist indices $k_1^j < k_2^j < \dots$ such that all runs $\kappa_{k^j_{i_1},\dots,k^j_{i_{j-1}},k^j_{i_j}+1}$ have the same initial state. Thus, there exist indices $k_1 < k_2 < \dots$ such that all runs $\kappa_{k_{i_1},\dots,k_{i_{j-1}},k_{i_j}+1}$ have the same initial state for each $j \in [2,n]$. We define $v_i = w_{k_i}$ for all $i \geq 1$ and

$$\begin{aligned}
\tilde{\kappa}_i^j &:= \kappa_{k_1,\dots,k_{j-1},k_{i-1}+1} \dots \kappa_{k_1,\dots,k_{j-1},k_i} \\
\tilde{\lambda}_i^j &:= \kappa_{k_1,\dots,k_{j-1},k_{i-1}+1} \dots \kappa_{k_1,\dots,k_{j-1},k_i-1} \lambda_{k_1,\dots,k_{j-1},k_i}
\end{aligned}$$

for all $j \in [1,n]$ and $i \geq j$ where $k_0 := 0$. Observe that the composition $\tilde{\lambda}_i^j \tilde{\kappa}_{i+1}^{j+1}$ forms a valid run since $\kappa_{k_1,\dots,k_{j-1},k_i,k_i+1}$ and $\kappa_{k_1,\dots,k_j,k_i+1}$ have the same initial state. Then $\tilde{\pi}_{i_1,\dots,i_n}$ is an accepting run on $v_{i_1} \otimes \cdots \otimes v_{i_n}$. Furthermore, this run decomposition is compatible with the $(\boldsymbol{\gamma}, \boldsymbol{\delta})$-decomposition of $v_{i_1} \otimes \cdots \otimes v_{i_n}$ where the generator $(\boldsymbol{\gamma}, \boldsymbol{\delta})$ is defined as

$$\begin{aligned}
\delta_i &:= \beta_{k_{i-1}+1} \dots \beta_{k_i} \\
\gamma_i &:= \beta_{k_{i-1}+1} \dots \beta_{k_i-1} \alpha_{k_i}
\end{aligned}$$

for all $i \geq 1$ where $k_0 := 0$. □

Similar to Proposition 4.2.8 for the case $n = 2$, we can now construct an NBA for the encodings of infinite n-cliques that simulates the runs in $\tilde{K}^n$ and $\tilde{\Lambda}^n$ from Lemma 4.6.3 in $2n$ components. For an automaton $\mathcal{A}$ let $|\mathcal{A}|$ denote the size of $\mathcal{A}$ in the usual encoding.

Proposition 4.6.4. *Given an NFA $\mathcal{A}$ for a regular relation $R \subseteq (\Sigma^*)^n$ with $n \geq 2$, one can construct in $\mathcal{O}(n \cdot \log |\mathcal{A}|)$ space an NBA $\mathcal{B}$ over the alphabet $(\Sigma_\perp \times \Sigma) \cup \{\#\}$ such that:*

- *If $\boldsymbol{w}$ is an infinite n-clique in R, then $\mathcal{B}$ accepts an encoding of a comb $\boldsymbol{v}$ which is a subsequence of $\boldsymbol{w}$.*
- *If $\mathcal{B}$ accepts an encoding of a comb $\boldsymbol{w}$, then $\boldsymbol{w}$ is an infinite n-clique in R.*

We are now ready to prove a theorem analogous to Theorem 4.2.1 for the evaluation of n-Ramsey quantifiers. We observe that the space complexity of the construction depends linearly on n and logarithmically on the size of the input NFA, which means that for any fixed n the asymptotic space complexity is the same as in Theorem 4.2.1.

Theorem 4.6.5. *Given a regular relation $R \subseteq (\Sigma^*)^{nd+k}$ with $n \geq 2$ by an NFA $\mathcal{A}$, one can construct in $\mathcal{O}(n \cdot \log |\mathcal{A}|)$ space an NFA for the relation $[\![\exists^{n\text{-ram}} \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \colon R(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, \boldsymbol{z})]\!]$, where $|\boldsymbol{x}_1| = \cdots = |\boldsymbol{x}_n| = d$ and $|\boldsymbol{z}| = k$.*

Proof. The proof is the same as in the case $n = 2$ by first assuming that $d = 1$ and constructing an NFA for the n-ary relation

$$R' := \{(u_1 \otimes c_1 \otimes \cdots \otimes c_k, \ldots, u_n \otimes c_1 \otimes \cdots \otimes c_k) \mid (u_1, \ldots, u_n, \boldsymbol{c}) \in R\}.$$

Then using Proposition 4.6.4, we construct an NFA which accepts at least one word from each infinite n-clique in R' and only accepts elements of infinite n-cliques in R'. Projecting away the first component yields the desired NFA. Moreover, the general case where $d \geq 1$ follows from an easy generalization of Proposition 3.1.4 to n-Ramsey quantifiers. □

Theorem 4.6.5 shows that n-Ramsey quantifiers can be effectively evaluated in automatic structures and can be reformulated as follows.

Corollary 4.6.6. *Given an automatic structure $\mathfrak{A}$ by a presentation $\mathfrak{p}$ and a formula $\varphi(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, \boldsymbol{z})$ over $\mathfrak{A}$ with $|\boldsymbol{x}_1| = \cdots = |\boldsymbol{x}_n|$, one can effectively compute an NFA for the relation $[\![\exists^{n\text{-ram}} \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \colon \varphi(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, \boldsymbol{z})]\!]_\mathfrak{p}$. Moreover, if an NFA $\mathcal{A}$ for $[\![\varphi]\!]_\mathfrak{p}$ is given, the construction can be done in $\mathcal{O}(n \cdot \log |\mathcal{A}|)$ space.*

To analyze the complexity of the infinite n-clique problem, we may regard n as part of the input or as a parameter. To deal with the latter, we first briefly introduce some notions from parameterized complexity theory that are needed in the following. For a deeper discussion on the topic we refer to [DF99; FG06].

Recall that formally a decision problem is a language, say over alphabet Σ, whereas a parameterized problem is a subset of $\Sigma^* \times \mathbb{N}$, where the natural number is regarded as the parameter. A parameterized problem $Q \subseteq \Sigma^* \times \mathbb{N}$ is in the parameterized complexity

class (uniform-)XC if for every $n \in \mathbb{N}$ the slice $Q_n := \{w \mid (w,n) \in Q\}$ lies in the classical complexity class C and there is a computable function assigning to every $n \in \mathbb{N}$ a Turing machine witnessing that Q_n is in C. By [CFG03], a parameterized problem $Q \subseteq \Sigma^* \times \mathbb{N}$ is in (uniform-)XNL if and only if there is a computable function $f\colon \mathbb{N} \to \mathbb{N}$ and a nondeterministic Turing machine that given a pair $(w,n) \in \Sigma^* \times \mathbb{N}$, decides whether $(w,n) \in Q$ in space at most $f(n) \cdot \log|w|$.

A PL-reduction from the parameterized problem $Q \subseteq \Sigma^* \times \mathbb{N}$ to the parameterized problem $Q' \subseteq (\Sigma')^* \times \mathbb{N}$ is a mapping $r\colon \Sigma^* \times \mathbb{N} \to (\Sigma')^* \times \mathbb{N}$ such that

1) for all $(w,n) \in \Sigma^* \times \mathbb{N}$ we have $(w,n) \in Q$ if and only if $r(w,n) \in Q'$,

2) there exists a computable function $g\colon \mathbb{N} \to \mathbb{N}$ such that for all $(w,n) \in \Sigma^* \times \mathbb{N}$ with $r(w,n) = (w',n')$ we have $n' \leq g(n)$, and

3) there exist a computable function $f\colon \mathbb{N} \to \mathbb{N}$ and a constant $c \in \mathbb{N}$ such that r is computable in space at most $f(n) + c \cdot \log|w|$.

In the following we only consider PL-reductions with $n = n'$ for all $n \in \mathbb{N}$ (a.k.a. *level-by-level* [SW15]).

Corollary 4.6.7. *The (parameterized) infinite n-clique problem over regular relations $R \subseteq (\Sigma^*)^{nd}$ given as NFA or DFA is* PSPACE*-complete (resp.* XNL*-complete under* PL*-reductions).*

Proof. The upper bounds follow from Theorem 4.6.5. For the lower bound of the parameterized version, we reduce from the *n-intersection non-emptiness problem*, which is the parameterized version of the intersection non-emptiness problem that asks whether the intersection of n regular languages given by DFAs for parameter $n \geq 1$ is non-empty. In [Weh14; Weh16] it is shown that this problem is XNL-complete under PL-reductions.

Let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be the given DFAs over the alphabet Σ and $\Sigma_\# := \Sigma \cup \{\#\}$ for a fresh symbol $\# \notin \Sigma$. We define the relation $R \subseteq ((\Sigma_\#)^*)^n$ such that $(w_1, \ldots, w_n) \in R$ if and only if for all $i \in [1,n]$ we have that $w_i = \#^{h_i} v_i$ for some $h_i \in \mathbb{N}$ and $v_i \in L(\mathcal{A}_i)$ such that $|w_j| \leq h_{j+1}$ for all $j \in [1, n-1]$. It is easy to see that R is a regular relation and a DFA for R can be constructed in $\mathcal{O}(\log(|\mathcal{A}_1| + \cdots + |\mathcal{A}_n|))$ space since due to the padding with #-symbols, we can simulate the $\mathcal{A}_i$ sequentially rather than in parallel. We claim that $\bigcap_{i=1}^n L(A_i) \neq \emptyset$ if and only if R has an infinite n-clique. If there exists $w \in \bigcap_{i=1}^n L(A_i)$, then

$$w_1 := \#w$$
$$w_{i+1} := \#^{|w_i|} w$$

for $i \geq 1$ defines an infinite n-clique in R. Conversely, if $(w_i)_{i \geq 1}$ is an infinite n-clique in R with $w_i = \#^{h_i} v_i$ for all $i \geq 1$, then for all $i \in [1,n]$ we have that $v_n \in L(\mathcal{A}_i)$ since

$$(w_{n-i+1}, \ldots, w_{n-1}, w_n, w_{n+1}, \ldots, w_{2n-i}) \in R.$$

Note that the above reduction shows that the parameterized infinite n-clique problem is already XNL-hard under PL-reductions for n-ary regular relations (i.e. with $d = 1$)

given by DFAs. Moreover, observe that the above yields a logspace reduction from the (non-parameterized) intersection non-emptiness problem, which by Proposition 2.3.6 is PSPACE-complete, to the (non-parameterized) infinite n-clique problem. □

4.6.2 Tree-Automatic Structures

We now consider n-Ramsey quantifiers in the context of tree-regular relations. Similar to the word case, for a comb $\boldsymbol{s}$ of trees with generator $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ we lift the definition of $(\boldsymbol{\alpha}, \boldsymbol{\beta})$-decompositions to convolutions $s_{i_1} \otimes \cdots \otimes s_{i_n}$. Likewise, we let K^d and Λ^d denote sets of context forests and $\pi_{i_1,\dots,i_n}$ denote the corresponding tree decomposition.

Lemma 4.6.8. *If $\boldsymbol{t}$ is an infinite n-clique in a tree-regular relation $R \subseteq (\mathcal{T}_\Sigma)^n$ given as an NTA $\mathcal{A}$, then there exist a generator $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of a subsequence $\boldsymbol{s}$ of $\boldsymbol{t}$, accepting runs $\rho(s_{i_1}, \dots, s_{i_n})$ of $\mathcal{A}$ on $s_{i_1} \otimes \cdots \otimes s_{i_n}$, and sets of context forests K^d, Λ^d such that*

$$\rho(s_{i_1}, \dots, s_{i_n}) = \pi_{i_1,\dots,i_n}$$

is a decomposition compatible with the $(\boldsymbol{\alpha}, \boldsymbol{\beta})$-decomposition of $s_{i_1} \otimes \cdots \otimes s_{i_n}$ for all $1 \leq i_1 < \cdots < i_n$.

Proof. The proof is, similar to the proof of Lemma 4.6.2 in the word case, a natural generalization of Lemma 4.3.6. □

With a generalization of the arguments in Proposition 4.6.9 using Lemma 4.6.8, we one can now construct an NBTA that accepts encodings of infinite n-cliques. If the relation is given by a D↑TA, then this construction works in time polynomial in $|\mathcal{A}|$ and exponential in n.

Proposition 4.6.9. *Given a D↑TA $\mathcal{A}$ for a tree-regular relation $R \subseteq (\mathcal{T}_\Sigma)^n$ with $n \geq 1$, one can construct in $\mathcal{O}(|\mathcal{A}|^{p(n)})$ time for a polynomial p an NBTA $\mathcal{B}$ over the ranked alphabet $(\Sigma_\perp)^2 \cup \{\#\}$ such that:*

- *If $\boldsymbol{t}$ is an infinite n-clique in R, then $\mathcal{B}$ accepts an encoding of a comb $\boldsymbol{s}$ which is a subsequence of $\boldsymbol{t}$.*
- *If $\mathcal{B}$ accepts an encoding of a comb $\boldsymbol{t}$, then $\boldsymbol{t}$ is an infinite n-clique in R.*

Using Proposition 4.6.9, we can generalize the proof of Theorem 4.3.2 to n-Ramsey quantifiers. Note that if the relation is given by an NTA $\mathcal{A}$ instead of a D↑TA, we can first determinize $\mathcal{A}$ using Proposition 2.3.14 with an exponential blow-up in $|\mathcal{A}|$ and subsequently apply the algorithm for D↑TAs, which is only exponential in n, resulting in an algorithm that is exponential in both $|\mathcal{A}|$ and n.

Theorem 4.6.10. *Given a D↑TA (resp. NTA) for a tree-regular relation $R \subseteq (\mathcal{T}_\Sigma)^{nd+k}$ with $n \geq 2$, one can construct in $\mathcal{O}(|\mathcal{A}|^{p(n)})$ (resp. $\mathcal{O}(2^{p(n \cdot |\mathcal{A}|)})$) time for a polynomial p an NTA for the relation $[\![\exists^{n\text{-ram}} \boldsymbol{x}_1, \dots, \boldsymbol{x}_n \colon R(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n, \boldsymbol{z})]\!]$, where $|\boldsymbol{x}_1| = \cdots = |\boldsymbol{x}_n| = d$ and $|\boldsymbol{z}| = k$.*

We can again reformulate Theorem 4.6.10 for tree-automatic structures.

Corollary 4.6.11. *Given a tree-automatic structure $\mathfrak{A}$ by a presentation $\mathfrak{p}$ and a formula $\varphi(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n, \boldsymbol{z})$ over $\mathfrak{A}$ with $|\boldsymbol{x}_1| = \cdots = |\boldsymbol{x}_n|$, one can effectively compute an NTA for the relation $[\![\exists^{n\text{-ram}} \boldsymbol{x}_1, \dots, \boldsymbol{x}_n \colon \varphi(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n, \boldsymbol{z})]\!]_{\mathfrak{p}}$. Moreover, if a D↑TA $\mathcal{A}$ for $[\![\varphi]\!]_{\mathfrak{p}}$ is given, the construction can be done in $\mathcal{O}(|\mathcal{A}|^{p(n)})$ time.*

If we consider n as part of the input, Theorem 4.6.10 implies that the infinite n-clique problem is decidable in exponential time regardless of the automaton model.

Corollary 4.6.12. *The infinite n-clique problem over tree-regular relations $R \subseteq (\mathcal{T}_\Sigma)^{nd}$ given as NTA, D↑TA, or D↓TA is* EXP*-complete.*

Proof. The upper bounds follow from Theorem 4.6.10. The lower bounds for NTAs and D↓TAs already hold for $n = 2$ by Theorem 4.3.1. To show the lower bound for D↑TAs, we can reduce from the intersection non-emptiness problem for D↑TAs, which by Proposition 2.3.18 is EXP-complete. The reduction works similarly as in Corollary 4.6.7. Let $\mathcal{A}_1, \dots, \mathcal{A}_n$ be the given D↑TAs over the ranked alphabet Σ and let $\Sigma_\# := \Sigma \cup \{\#\}$ for a fresh symbol $\# \notin \Sigma$ of rank 1. For a tree $t \in \mathcal{T}_\Sigma$ let $\#_h(t) \in \mathcal{T}_{\Sigma_\#}$ be the tree t appended with a chain of #-symbols of length $h \in \mathbb{N}$ to the top. We define the relation $R \subseteq (\mathcal{T}_{\Sigma_\#})^n$ such that $(t_1, \dots, t_n) \in R$ if and only if for all $i \in [1, n]$ we have that $t_i = \#_{h_i}(s_i)$ for some $h_i \in \mathbb{N}$ and $s_i \in L(\mathcal{A}_i)$ such that the length of the left-most path in t_j is at most h_{j+1} for all $j \in [1, n-1]$. Then it holds that $\bigcap_{i=1}^n L(\mathcal{A}_i) \neq \emptyset$ if and only if R has an infinite n-clique. Note that a D↑TA that recognizes R can be constructed in time polynomial in $|\mathcal{A}_1| + \cdots + |\mathcal{A}_n|$. □

Corollary 4.6.12 implies that the parameterized infinite n-clique problem for NTAs and D↓TAs is in XEXP. This is in some sense optimal since the problem is EXP-hard for all slices with $n \geq 2$: We can reduce the problem for $n = 2$, which is EXP-complete by Theorem 4.3.1 (already for $d = 1$), to the problem for any fixed $n \geq 2$ by constructing from the binary relation $R \subseteq (\mathcal{T}_\Sigma)^2$ the n-ary relation $R' \subseteq (\mathcal{T}_\Sigma)^n$ such that $(t_1, \dots, t_n) \in R'$ if and only if $(t_i, t_j) \in R$ for all $1 \leq i < j \leq n$.

We define an FPT-reduction similarly as a PL-reduction, but the reduction has to be computable in $f(n) \cdot |w|^c$ time. Again, we only consider level-by-level reductions.

Corollary 4.6.13. *The parameterized infinite n-clique problem over tree-regular relations $R \subseteq (\mathcal{T}_\Sigma)^{nd}$ given as D↑TA is* XP*-complete under* FPT*-reductions.*

Proof. The upper bound follows from Theorem 4.6.10. For the lower bound we can use the same reduction as in Corollary 4.6.12 from the n-intersection non-emptiness problem for D↑TAs, which is XP-complete under FPT-reductions (shown in [SW15] for D↓TAs but similar for D↑TAs). □

5 Ramsey Quantifiers in Linear Arithmetics

Parts of this chapter were published by the author in [Ber+24].

In the previous chapter we saw that the evaluation of the Ramsey quantifier over (tree-)regular relations can be applied to obtain precise complexity results for both liveness verification and recognizability. We now consider similar questions in the setting of linear arithmetics, which will allow us to make use of highly optimized SMT solvers. For liveness, instead of (tree) automata, we assume that the reachability relation is provided as a formula in linear integer/real arithmetic. For systems like succinct one-counter automata [Li+20], continuous vector addition systems with states [BH17], and Parikh automata [Sei+04] one can even compute an existential formula for the reachability relation in polynomial time. Using Ramsey quantifiers, we will then deduce the precise complexity of linear liveness for these classes of systems. Liner liveness is similar to recurrent reachability, where the set of target configurations is defined by a formula in linear arithmetics. But it additionally allows to express conditions that should hold between each pair of the infinite run. For example, one can demand that there exists some positive number such that the value of a real clock increases by at least this number between every pair of configurations on the run. This ensures that the clock value grows unboundedly.

As the analogue to recognizability from a logic perspective, we will see that Ramsey quantifiers can be used to decide monadic decomposability of quantifier-free formulas in linear integer/real arithmetic with optimal complexity. Similar to recognizability, the connection can be drawn via finite-index equivalence relations. As a further application, we can use the Ramsey quantifier to check with precise complexity whether a relation given as formula in quantifier-free Presburger arithmetic is a well-quasi-ordering. This is relevant in the context of well-structured transition systems [FG19a].

The main result of this chapter concerns the elimination of the Ramsey quantifier applied to an existential formula in linear integer/real arithmetic. More precisely, our goal is to find an existential formula in LIA, LRA, or LIRA that is equivalent to

$$\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \exists \boldsymbol{w} \colon \gamma(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{z})$$

where γ is a quantifier-free formula in the respective theory. The main idea is to indentify conditions expressible in the logic under consideration that hold if and only if the input formula has an infinite clique. However, those conditions will not directly define the clique. They will only be equivalent to existence of a clique. The reason is that there are Presburger formulas that have infinite cliques, but none of them is definable in Presburger arithmetic. Instead, we will show that we can focus on arithmetic progressions, i.e. sequences of the form

$$\boldsymbol{a}_0, \boldsymbol{a}_0 + \boldsymbol{a}, \boldsymbol{a}_0 + 2\boldsymbol{a}, \dots$$

that are guaranteed to contain an infinite clique as a subsequence. Similarly, for formulas in linear real arithmetic we consider sequences of the form

$$\boldsymbol{a} - \boldsymbol{d}_{\text{con}} + \boldsymbol{d}_{\infty}, \boldsymbol{a} - \frac{1}{2}\boldsymbol{d}_{\text{con}} + 2\boldsymbol{d}_{\infty}, \boldsymbol{a} - \frac{1}{3}\boldsymbol{d}_{\text{con}} + 3\boldsymbol{d}_{\infty}, \dots$$

where the vector $\boldsymbol{d}_{\text{con}}$ is used to express convergent behavior, which is not needed in the Presburger case. For example, consider the formula $\varphi(x, y) := x < y \wedge x < 1$. Clearly, when interpreted over the integers, φ does not contain any infinite cliques since it ensures that any infinite sequence is strictly increasing but bounded from above by 1. Over the reals, however, φ contains, for example, the infinite clique $0, \frac{1}{2}, \frac{3}{4}, \frac{7}{8}, \dots$, which is strictly increasing and converges to 1.

5.1 Main Results

Before we dive into the proofs, let us summarize the main results of this chapter. As mentioned above, our goal is to compute a formula ψ' that is equivalent to

$$\psi := \exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \exists \boldsymbol{w} \colon \gamma(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{z})$$

where γ is a quantifier-free LIA, LRA, or LIRA formula such that ψ' only uses existential quantifiers, i.e. ψ' is contained in the existential fragment of LIA, LRA, or LIRA.

Eliminating existential quantifiers The first step towards the elimination of the Ramsey quantifier in ψ is to simplify the problem by showing that we can always assume the vector $\boldsymbol{w}$ of existentially quantified variables to be empty. More specifically, in Section 5.2 we prove that existential quantifiers in the scope of a Ramsey quantifier can be eliminated by extending the Ramsey quantified vectors of variables:

Theorem 5.2.1. *Let $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{z})$ be an existential formula in LIRA, where $\boldsymbol{x}$ and $\boldsymbol{y}$ have the same type. Then the formulas*

(i) $\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \exists \boldsymbol{w} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{z})$ *and*

(ii) $\exists^{\text{ram}} (\boldsymbol{x}, \boldsymbol{v}_1, \boldsymbol{v}_2), (\boldsymbol{y}, \boldsymbol{w}_1, \boldsymbol{w}_2) \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{v}_1 + \boldsymbol{w}_2, \boldsymbol{z}) \wedge \boldsymbol{x} \neq \boldsymbol{y}$

are equivalent, where $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{w}_1, \boldsymbol{w}_2$ have the same type as $\boldsymbol{w}$.

In particular, note that this only introduces linearly many new Ramsey quantified variables. If we would use existing quantifier elimination techniques (without introducing new Ramsey quantified variables) instead, this would incur an exponential blow-up for Presburger arithmetic [Haa+24; CMS24] and a double exponential blow-up for LIRA [Wei99]. Loosely speaking, Theorem 5.2.1 tells us that instead of choosing a new valuation of $\boldsymbol{w}$ for each edge of the clique, we can add two vectors to every node such that for $\boldsymbol{w}$ we can always choose the sum of the first additional vector of the left node and the second additional vector of the right node.

Eliminating Ramsey quantifiers in LIA In Section 5.3 we show how to eliminate the Ramsey quantifier in case of existential Presburger formulas.

Theorem 5.3.1. *Given an existential Presburger formula $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ with $|\boldsymbol{x}| = |\boldsymbol{y}|$, one can construct in polynomial time an existential Presburger formula of linear size that is equivalent to $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$.*

As a consequence of Theorem 5.2.1 we can assume that φ is quantifier-free. The key idea is to use existential quantifiers to guess how the left-hand sides and right-hand sides in a conjunction of inequalities of the form

$$\bigwedge_{i=1}^{n} \boldsymbol{r}_i \cdot \boldsymbol{x} < \boldsymbol{s}_i \cdot \boldsymbol{y} + \boldsymbol{t}_i \cdot \boldsymbol{z} + h_i$$

evolve. This information will be used to formulate necessary and sufficient conditions for the existence of an infinite clique. For example, if over an infinite sequence $(\boldsymbol{a}_k)_{k \geq 1}$ the left-hand side $(\boldsymbol{r}_i \cdot \boldsymbol{a}_k)_{k \geq 1}$ grows unboundedly, but the right-hand side $(\boldsymbol{s}_i \cdot \boldsymbol{a}_k + \boldsymbol{t}_i \cdot \boldsymbol{c} + h_i)_{k \geq 1}$ is bounded from above, then no infinite subsequence of $(\boldsymbol{a}_k)_{k \geq 1}$ can form a clique with respect to $\boldsymbol{c}$. Whereas, if $(\boldsymbol{s}_i \cdot \boldsymbol{a}_k + \boldsymbol{t}_i \cdot \boldsymbol{c} + h_i)_{k \geq 1}$ also grows unboundedly, then we can always find an infinite subsequence $(\boldsymbol{a}_{i_k})_{k \geq 1}$ such that

$$\boldsymbol{r}_i \cdot \boldsymbol{a}_{i_k} < \boldsymbol{s}_i \cdot \boldsymbol{a}_{i_\ell} + \boldsymbol{t}_i \cdot \boldsymbol{c} + h_i$$

holds for all $1 \leq k < \ell$. We will show that a Presburger formula can verify the existence of an infinite sequence that results in the guessed evolution of both sides of the inequalities.

Eliminating Ramsey quantifiers in LRA With a similar strategy as in Theorem 5.3.1 we will prove in Section 5.4 that also in the existential fragment of LRA the Ramsey quantifier can be efficiently eliminated.

Theorem 5.4.1. *Given an existential formula $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ in LRA with $|\boldsymbol{x}| = |\boldsymbol{y}|$, one can construct in polynomial time an existential formula in LRA of linear size that is equivalent to $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$.*

For the proof we also have to take the possibly convergent behavior of the sides of the inequalities into account.

Eliminating Ramsey quantifiers in LIRA To prove the analogous result for LIRA, we reduce in Section 5.5 to the elimination in LIA and LIRA by computing the decomposition using Lemma 2.4.5.

Theorem 5.5.1. *Given an existential formula $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ in LIRA, where $\boldsymbol{x}$ and $\boldsymbol{y}$ have the same type, one can construct in polynomial time an existential formula in LIRA of linear size that is equivalent to $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$.*

As a consequence of Theorem 5.5.1, we obtain the precise complexity of the infinite clique problem.

Corollary 5.5.2. *The infinite clique problem for existential formulas in LIRA is* NP*-complete.*

Note that decidability of the infinite clique problem for LIRA formulas already follows from [DI02]. However, the procedure by Dang and Ibarra [DI02], on the one hand, requires the input formula to be quantifier-free and, on the other hand, yields a formula with multiple quantifier alternations, whose satisfiability has to be checked. For checking satisfiability of a LIRA formula with quantifier alternations they provide, based on [Wei99], a double exponential-time upper bound.

Monadic decomposability In Section 5.6 we present several applications of the above results. Recall that a formula is monadically decomposable if it can be written as a Boolean combination of formulas with only a single free variable. In Proposition 3.3.10 we showed that a quantifier-free formula $\varphi(x_1, \ldots, x_k)$ is monadically decomposable in the quantifier-free fragment of LIA, LRA, or LIRA if and only if certain equivalence relations $\approx_j$ for $j \in [1, k-1]$ have finite index. Since $\not\approx_j$ is an existential formula in LIA, LRA, or LIRA and an equivalence relation has infinite index if and only if its complement has an infinite clique, we can use Corollary 5.5.2 to check for infinite index of $\approx_j$ in NP.

Corollary 5.6.1. *Given a quantifier-free formula in LIA, LRA, or LIRA, deciding monadic decomposability in the respective quantifier-free fragment is* coNP*-complete.*

In the case of Presburger arithmetic coNP-completeness was already known [Hag+20], but for LRA and LIRA the precise complexity remained open.

Linear liveness As we have already seen in Section 4.4.1, the Ramsey quantifier can be used to express liveness conditions. We will consider several concrete classes of systems (involving counters and/or clocks) whose reachability relations are expressible in linear arithmetics and for which the above results can be applied to answer liveness questions. Here, we consider the linear liveness problem that asks for the existence of an infinite run such that between every pair of infinitely many configurations on the run a condition specified in existential linear arithmetics is satisfied. A special case of this problem is to check for an infinite run of configurations satisfying some property infinitely often. For example, we show that linear liveness is NP-complete for continuous vector addition systems with states, succinct one-counter automata, reversal-bounded counter machines, and Parikh automata.

Checking for WQO Recently, in the context of well-structured transition systems, Finkel and Gupta [FG19a] raised the question of how to check whether a relation defined by a Presburger formula is a well-quasi-ordering (WQO). Using Theorem 5.3.1, we can settle the precise complexity of this problem in case the relation is given as a quantifier-free Presburger formula.

Corollary 5.6.17. *Given a quantifier-free Presburger formula $\varphi(\boldsymbol{x}, \boldsymbol{y})$ with $|\boldsymbol{x}| = |\boldsymbol{y}| = d$, it is* coNP*-complete to decide whether $[\![\varphi]\!] \subseteq \mathbb{Z}^d \times \mathbb{Z}^d$ is a WQO.*

5.2 Eliminating Existential Quantifiers

Our main result of this chapter concerns the elimination of the Ramsey quantifier from formulas of the form $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ where ψ is an existential formula in LIRA. In this section we first show that we can reduce to the case where ψ is quantifier-free. The problem with eliminating quantifiers first is that general quantifier elimination techniques incur a blow-up (exponential for Presburger arithmetic [Haa+24; CMS24] and double exponential for LRA [Wei99]), even when restricted to a single block of existential quantifiers. Instead of relying on these techniques, we show that existentially quantified variables occurring in the scope of a Ramsey quantifier can be exchanged for more, but only linearly many, Ramsey quantified variables. More precisely, we prove the following equivalence:

Theorem 5.2.1. *Let $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{z})$ be an existential formula in LIRA, where $\boldsymbol{x}$ and $\boldsymbol{y}$ have the same type. Then the formulas*

(i) $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \exists \boldsymbol{w} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{z})$ *and*

(ii) $\exists^{\mathrm{ram}} (\boldsymbol{x}, \boldsymbol{v}_1, \boldsymbol{v}_2), (\boldsymbol{y}, \boldsymbol{w}_1, \boldsymbol{w}_2) \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{v}_1 + \boldsymbol{w}_2, \boldsymbol{z}) \wedge \boldsymbol{x} \neq \boldsymbol{y}$

are equivalent, where $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{w}_1, \boldsymbol{w}_2$ have the same type as $\boldsymbol{w}$.

Thus, if we have a formula $\exists \boldsymbol{w} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{z})$ where φ is quantifier-free, then Theorem 5.2.1 allows us to eliminate the block $\exists \boldsymbol{w}$ of existential quantifiers by moving $\boldsymbol{w}$ under the Ramsey quantifier. Note that both formulas express the existence of an infinite clique. The first one says that for every edge $\boldsymbol{x} \to \boldsymbol{y}$ in the clique we can choose a vector $\boldsymbol{w}$ such that $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{z})$ is satisfied. The second formula says that $\boldsymbol{w}$ can be chosen in a specific way: It says that for each vertex one can choose two vectors $(\boldsymbol{w}_1, \boldsymbol{w}_2)$ such that for each edge $\boldsymbol{x} \to \boldsymbol{y}$ the vector $\boldsymbol{w}$ can be the sum of $\boldsymbol{w}_1$ for $\boldsymbol{x}$ and $\boldsymbol{w}_2$ for $\boldsymbol{y}$. Thus, (ii) clearly implies (i). The challenge is to show that also the reverse implication holds. The rest of this section is devoted to proving Theorem 5.2.1 by first considering the integer and the real case separately and then using Lemma 2.4.5 for the mixed case.

5.2.1 Linear Integer Arithmetic

We start with the elimination of existential quantifiers in case of Presburger arithmetic.

Theorem 5.2.2. *Let $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{z})$ be an existential formula in Presburger arithmetic with $|\boldsymbol{x}| = |\boldsymbol{y}|$. Then the formulas*

(i) $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \exists \boldsymbol{w} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{z})$ *and*

(ii) $\exists^{\mathrm{ram}} (\boldsymbol{x}, \boldsymbol{v}_1, \boldsymbol{v}_2), (\boldsymbol{y}, \boldsymbol{w}_1, \boldsymbol{w}_2) \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{v}_1 + \boldsymbol{w}_2, \boldsymbol{z}) \wedge \boldsymbol{x} \neq \boldsymbol{y}$

are equivalent.

To prove Theorem 5.2.2, it suffices to prove it in case $\boldsymbol{w}$ consists of just one variable w. Then, Theorem 5.2.2 follows by induction: Since we allow φ to have existential quantifiers, we can eliminate the block $\exists \boldsymbol{w}$ from the outermost to the innermost variable. Therefore, it remains to show the following.

Lemma 5.2.3. *Let $\varphi(\boldsymbol{x}, \boldsymbol{y}, w, \boldsymbol{z})$ be an existential formula in Presburger arithmetic with $|\boldsymbol{x}| = |\boldsymbol{y}|$. Then the formulas*

(i) $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \exists w \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, w, \boldsymbol{z})$ *and*

(ii) $\exists^{\mathrm{ram}} (\boldsymbol{x}, v_1, v_2), (\boldsymbol{y}, w_1, w_2) \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, v_1 + w_2, \boldsymbol{z}) \wedge \boldsymbol{x} \neq \boldsymbol{y}$

are equivalent.

Simple Presburger formulas Before we consider the general case, let us assume that we are dealing with Presburger formulas of a simplified form. Let $\boldsymbol{u}$ be a vector of n variables and w be a variable. We say that a Presburger formula $\varphi(\boldsymbol{u}, w)$ is *w-simple* if it is a Boolean combination of formulas of the form $\boldsymbol{r} \cdot \boldsymbol{u} + c < w$, $w < \boldsymbol{r} \cdot \boldsymbol{u} + c$, and modulo constraints over $\boldsymbol{u}$ and w, where $\boldsymbol{r} \in \mathbb{Z}^n$ and $c \in \mathbb{Z}$. In particular, this means that w-simple formulas are quantifier-free.

Lemma 5.2.4. *Let $\varphi(\boldsymbol{x}, \boldsymbol{y}, w, \boldsymbol{z})$ be a w-simple Presburger formula with $|\boldsymbol{x}| = |\boldsymbol{y}|$. Then the formulas*

(i) $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \exists w \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, w, \boldsymbol{z})$

(ii) $\exists^{\mathrm{ram}} (\boldsymbol{x}, v_1, v_2), (\boldsymbol{y}, w_1, w_2) \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, v_1 + w_2, \boldsymbol{z}) \wedge \boldsymbol{x} \neq \boldsymbol{y}$

are equivalent.

Proof. We first move all negations in φ inwards to the atoms and possibly negate them (which for modulo constraints introduces disjunctions). We then bring φ into disjunctive normal form and distribute the Ramsey quantifier and existential quantifier over the disjunctions. Since φ is simple, we can assume that it is a conjunction of inequalities

$$\alpha_i(\boldsymbol{x}) + \beta_i(\boldsymbol{y}) + \gamma_i(\boldsymbol{z}) < w$$

for $i \in [1, n]$ and

$$w < \alpha'_j(\boldsymbol{x}) + \beta'_j(\boldsymbol{y}) + \gamma'_j(\boldsymbol{z})$$

for $j \in [1, m]$ and modulo constraints

$$\delta_\ell(\boldsymbol{x}, \boldsymbol{y}, w, \boldsymbol{z}) \equiv_{e_\ell} 0$$

for $\ell \in [1, k]$. Here, the $\alpha_i, \beta_i, \gamma_i, \alpha'_j, \beta'_j, \gamma'_j, \delta_\ell$ are linear functions with integer coefficients (and possibly constants). Let $f_i(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) := \alpha_i(\boldsymbol{x}) + \beta_i(\boldsymbol{y}) + \gamma_i(\boldsymbol{z})$ for all $i \in [1, n]$ and $f'_j(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) := \alpha'_j(\boldsymbol{x}) + \beta'_j(\boldsymbol{y}) + \gamma'_j(\boldsymbol{z})$ for all $j \in [1, m]$. In the following we assume that $n, m > 0$ and note that the other cases are simpler and can be handled similarly.

Assume $\boldsymbol{c} \in \mathbb{Z}^{|\boldsymbol{z}|}$ satisfies the formula in (i), i.e. there is an infinite sequence $(\boldsymbol{a}_i)_{i \geq 1}$ of pairwise distinct vectors over $\mathbb{Z}$ such that for all $i < j$ there exists $b_{i,j} \in \mathbb{Z}$ such that $\varphi(\boldsymbol{a}_i, \boldsymbol{a}_j, b_{i,j}, \boldsymbol{c})$ holds. By Ramsey's theorem, we can take an infinite subsequence such that we can assume that $f_1(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{c}) \leq \cdots \leq f_n(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{c})$ and $f'_1(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{c}) \leq \cdots \leq f'_m(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{c})$ for all $i < j$. Thus, it suffices to consider the greatest lower bound

$f_n(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{c})$ and the smallest upper bound $f'_1(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{c})$ on w. Let $f := f_n$, $\alpha := \alpha_n$, $\beta := \beta_n$, and $\gamma := \gamma_n$. Let $N := \operatorname{lcm}\{e_1, \dots, e_k\}$ be the least common multiple of all moduli, where we set $N := 1$ if $k = 0$. Observe that by the Chinese remainder theorem, $b_{i,j}$ can always be chosen from the interval $[f(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{c}) + 1, f(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{c}) + N]$ for all $i < j$. Since this interval has fixed length N, by Ramsey's theorem we can restrict to an infinite subsequence such that there is a constant $r \in [1, N]$ such that $f(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{c}) + r = b_{i,j}$ for all $i < j$. Now, if we set $b^1_i := \alpha(\boldsymbol{a}_i)$ and $b^2_i := \beta(\boldsymbol{a}_i) + \gamma(\boldsymbol{c}) + r$ for $i \geq 1$, the infinite sequence $(\boldsymbol{a}_i, b^1_i, b^2_i)_{i \geq 1}$ satisfies $\varphi(\boldsymbol{a}_i, \boldsymbol{a}_j, b^1_i + b^2_j, \boldsymbol{c})$ for all $i < j$ as desired. □

General Presburger formulas We now prove Lemma 5.2.3. Observe that if we can show equivalence of the formulas (i) and (ii) for quantifier-free φ with modulo constraints, then the same follows for general existential Presburger formulas φ: Let φ be an existential Presburger formula. By Proposition 2.4.2, there exists an equivalent quantifier-free formula φ' over $\langle \mathbb{Z}; <, (\equiv_e)_{e \in \mathbb{N}}, +, 0, 1 \rangle$. Thus, if the equivalence in Lemma 5.2.3 holds for φ', then it also holds for φ itself. Therefore, we may assume that φ is quantifier-free but possibly contains modulo constraints.

We now modify φ similarly as in the standard quantifier elimination procedure for Presburger arithmetic (Proposition 2.4.2). To this end, we define the w-simplification of a quantifier-free formula $\theta(\boldsymbol{u}, w)$ with modulo constraints. First observe that we can assume that θ is a Boolean combination of inequalities of the form $\boldsymbol{r} \cdot \boldsymbol{u} + c \sim sw$ for $\sim \in \{<, >\}$ and modulo constraints $\boldsymbol{r} \cdot \boldsymbol{u} + sw \equiv_e c$ for some vector $\boldsymbol{r}$ over $\mathbb{Z}$ and $c, s \in \mathbb{Z}$. (Note that in Presburger arithmetic equality can be expressed by a conjunction of two strict inequalities.) Let N be the least common multiple of all coefficients s of w in these constraints. We obtain θ' from θ by replacing each

- inequality $\boldsymbol{r} \cdot \boldsymbol{u} + c \sim sw$ with $\frac{N}{s} \cdot \boldsymbol{r} \cdot \boldsymbol{u} + \frac{N}{s} \cdot c \sim w$ and
- modulo constraint $\boldsymbol{r} \cdot \boldsymbol{u} + sw \equiv_e c$ with $\frac{N}{s} \cdot \boldsymbol{r} \cdot \boldsymbol{u} + w \equiv_{\frac{N}{s} \cdot e} \frac{N}{s} \cdot c$.

Now, the *w-simplification* of θ is the pair (ψ, N) where $\psi(\boldsymbol{u}, w) := \theta'(\boldsymbol{u}, w) \wedge w \equiv_N 0$. Then clearly, ψ is w-simple and for every integer vector $\boldsymbol{a}$ and $b \in \mathbb{Z}$ we have

$$\theta(\boldsymbol{a}, b) \text{ if and only if } \psi(\boldsymbol{a}, Nb)$$

and moreover, $\psi(\boldsymbol{a}, b)$ implies that b is a multiple of N.

Now, suppose $\varphi(\boldsymbol{x}, \boldsymbol{y}, w, \boldsymbol{z})$ is quantifier-free but contains modulo constraints. Moreover, let $\psi(\boldsymbol{x}, \boldsymbol{y}, w, \boldsymbol{z})$ and N be the w-simplification of φ. To show Lemma 5.2.3, let us assume the formula in (i) is satisfied for some integer vector $\boldsymbol{c}$. Then $\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \exists w \colon \psi(\boldsymbol{x}, \boldsymbol{y}, w, \boldsymbol{c})$ holds since we can multiply the witnessing valuations of w with N. By Lemma 5.2.4, this implies that

$$\exists^{\text{ram}} (\boldsymbol{x}, v_1, v_2), (\boldsymbol{y}, w_1, w_2) \colon \psi(\boldsymbol{x}, \boldsymbol{y}, v_1 + w_2, \boldsymbol{c})$$

is satisfied, meaning there exists an infinite sequence $(\boldsymbol{a}_i, b_i, b'_i)_{i \geq 1}$ where $\boldsymbol{a}_1, \boldsymbol{a}_2, \dots$ are pairwise distinct and $\psi(\boldsymbol{a}_i, \boldsymbol{a}_j, b_i + b'_j, \boldsymbol{c})$ for every $i < j$. By construction of ψ,

this implies that $b_i + b'_j$ is a multiple of N for every $i < j$ and therefore all the integers $b_1, b_2, \ldots$ must have the same remainder modulo N, say $r \in [0, N-1]$, and all $b'_1, b'_2, \ldots$ must be congruent to $-r$ modulo N. This means that the quotients $\bar{b}_i := (b_i - r)/N$ and $\bar{b}'_i := (b'_i + r)/N$ must be integers. Then for every $i < j$ we have $\psi(\boldsymbol{a}_i, \boldsymbol{a}_j, N(\bar{b}_i + \bar{b}'_j), \boldsymbol{c})$ and hence $\varphi(\boldsymbol{a}_i, \boldsymbol{a}_j, \bar{b}_i + \bar{b}'_j, \boldsymbol{c})$. Thus, the sequence $(\boldsymbol{a}_i, \bar{b}_i, \bar{b}'_i)_{i \geq 1}$ shows that $\exists^{\mathrm{ram}}(\boldsymbol{x}, v_1, v_2), (\boldsymbol{y}, w_1, w_2) \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, v_1 + w_2, \boldsymbol{c}) \wedge \boldsymbol{x} \neq \boldsymbol{y}$ is satisfied.

5.2.2 Linear Real Arithmetic

We now turn to the case where φ is a formula in LRA.

Theorem 5.2.5. *Let $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{z})$ be an existential formula in LRA with $|\boldsymbol{x}| = |\boldsymbol{y}|$. Then the formulas*

(i) $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \exists \boldsymbol{w} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{z})$ *and*

(ii) $\exists^{\mathrm{ram}}(\boldsymbol{x}, \boldsymbol{v}_1, \boldsymbol{v}_2), (\boldsymbol{y}, \boldsymbol{w}_1, \boldsymbol{w}_2) \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{v}_1 + \boldsymbol{w}_2, \boldsymbol{z}) \wedge \boldsymbol{x} \neq \boldsymbol{y}$

are equivalent.

As in the Presburger case, we may assume that $\boldsymbol{w}$ consists of just one variable w since Theorem 5.2.5 then follows by induction. Therefore, it suffices to prove the following lemma.

Lemma 5.2.6. *Let $\varphi(\boldsymbol{x}, \boldsymbol{y}, w, \boldsymbol{z})$ be an existential formula in LRA with $|\boldsymbol{x}| = |\boldsymbol{y}|$. Then the formulas*

(i) $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \exists w \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, w, \boldsymbol{z})$ *and*

(ii) $\exists^{\mathrm{ram}}(\boldsymbol{x}, v_1, v_2), (\boldsymbol{y}, w_1, w_2) \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, v_1 + w_2, \boldsymbol{z}) \wedge \boldsymbol{x} \neq \boldsymbol{y}$

are equivalent.

Proof. Again by eliminating the existential quantifiers in φ using Proposition 2.4.4, transforming it into disjunctive normal form, and distributing the quantifiers over the disjunctions, we can assume that φ is a conjunction of inequalities

$$\alpha_i(\boldsymbol{x}) + \beta_i(\boldsymbol{y}) + \gamma_i(\boldsymbol{z}) < w$$

for $i \in [1, n]$ and

$$w < \alpha'_j(\boldsymbol{x}) + \beta'_j(\boldsymbol{y}) + \gamma'_j(\boldsymbol{z})$$

for $j \in [1, m]$ and equality constraints

$$w = \delta_\ell(\boldsymbol{x}) + \kappa_\ell(\boldsymbol{y}) + \lambda_\ell(\boldsymbol{z})$$

for $\ell \in [1, k]$. Here, the $\alpha_i, \beta_i, \gamma_i, \alpha'_j, \beta'_j, \gamma'_j, \delta_i, \kappa_i, \lambda_i$ are linear functions with rational coefficients (and possibly constants).

Assume $\boldsymbol{c} \in \mathbb{R}^{|\boldsymbol{z}|}$ satisfies the formula in (i), i.e. there is an infinite sequence $(\boldsymbol{a}_i)_{i \geq 1}$ of pairwise distinct vectors over $\mathbb{R}$ such that for all $i < j$ there exists $b_{i,j} \in \mathbb{R}$ such

that $\varphi(\boldsymbol{a}_i, \boldsymbol{a}_j, b_{i,j}, \boldsymbol{c})$ holds. Clearly, if $k > 0$, we can eliminate w by replacing it with $\delta_1(\boldsymbol{x}) + \kappa_1(\boldsymbol{y}) + \lambda_1(\boldsymbol{z})$. Thus, setting $b_i := \delta_1(\boldsymbol{a}_i)$ and $b'_i := \kappa_1(\boldsymbol{a}_i) + \lambda_1(\boldsymbol{c})$ for $i \geq 1$, the sequence $(\boldsymbol{a}_i, b_i, b'_i)_{i\geq 1}$ satisfies $\varphi(\boldsymbol{a}_i, \boldsymbol{a}_j, b_i + b'_j, \boldsymbol{c})$ for all $i < j$. So assume $k = 0$, i.e. φ only contains lower and upper bounds on w. We further assume that $n, m > 0$ since the other cases are obvious. As in the Presburger case, we can apply Ramsey's theorem so that we only have to consider the greatest lower bound $\alpha(\boldsymbol{x}) + \beta(\boldsymbol{y}) + \gamma(\boldsymbol{z})$ and the smallest upper bound $\alpha'(\boldsymbol{x}) + \beta'(\boldsymbol{y}) + \gamma'(\boldsymbol{z})$ on w. This means that w can always be chosen to be the midpoint of this interval. Therefore, if we set $b_i := (\alpha(\boldsymbol{a}_i) + \alpha'(\boldsymbol{a}_i))/2$ and $b'_i := (\beta(\boldsymbol{a}_i) + \gamma(\boldsymbol{c}) + \beta'(\boldsymbol{a}_i) + \gamma'(\boldsymbol{c}))/2$ for $i \geq 1$, the sequence $(\boldsymbol{a}_i, b_i, b'_i)_{i\geq 1}$ satisfies $\varphi(\boldsymbol{a}_i, \boldsymbol{a}_j, b_i + b'_j, \boldsymbol{c})$ for all $i < j$. □

5.2.3 Linear Integer Real Arithmetic

We are now ready to prove Theorem 5.2.1. As mentioned above, it suffices to prove that the formula in (i) implies the one in (ii). Let $\psi(\boldsymbol{z})$ be the formula in (i) and $\boldsymbol{c}$ be a valuation of $\boldsymbol{z}$ that satisfies ψ. By Lemma 2.4.5, there exists a decomposition $\varphi'(\boldsymbol{x}^{\mathrm{i/r}}, \boldsymbol{y}^{\mathrm{i/r}}, \boldsymbol{w}^{\mathrm{i/r}}, \boldsymbol{z}^{\mathrm{i/r}})$ of $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{z})$. By definition of a decomposition, there is a valuation $\boldsymbol{c}^{\mathrm{i/r}}$ of $\boldsymbol{z}^{\mathrm{i/r}}$ with $\boldsymbol{c}^{\mathrm{int}} + \boldsymbol{c}^{\mathrm{real}} = \boldsymbol{c}$ such that

$$\exists^{\mathrm{ram}} \boldsymbol{x}^{\mathrm{i/r}}, \boldsymbol{y}^{\mathrm{i/r}} \colon \exists \boldsymbol{w}^{\mathrm{i/r}} \colon \varphi'(\boldsymbol{x}^{\mathrm{i/r}}, \boldsymbol{y}^{\mathrm{i/r}}, \boldsymbol{w}^{\mathrm{i/r}}, \boldsymbol{c}^{\mathrm{i/r}}).$$

We bring φ' into the disjunctive form

$$\varphi' \equiv \bigvee_{i=1}^{n} \alpha_i(\boldsymbol{x}^{\mathrm{int}}, \boldsymbol{y}^{\mathrm{int}}, \boldsymbol{w}^{\mathrm{int}}, \boldsymbol{z}^{\mathrm{int}}) \wedge \beta_i(\boldsymbol{x}^{\mathrm{real}}, \boldsymbol{y}^{\mathrm{real}}, \boldsymbol{w}^{\mathrm{real}}, \boldsymbol{z}^{\mathrm{real}})$$

where α_i is an existential Presburger formula and β_i is an existential formula in LRA for all $i \in [1, n]$. By Ramsey's theorem, there exists an $i \in [1, n]$ such that

$$\exists^{\mathrm{ram}} \boldsymbol{x}^{\mathrm{i/r}}, \boldsymbol{y}^{\mathrm{i/r}} \colon \exists \boldsymbol{w}^{\mathrm{int}} \colon \alpha_i(\boldsymbol{x}^{\mathrm{int}}, \boldsymbol{y}^{\mathrm{int}}, \boldsymbol{w}^{\mathrm{int}}, \boldsymbol{c}^{\mathrm{int}}) \wedge \exists \boldsymbol{w}^{\mathrm{real}} \colon \beta_i(\boldsymbol{x}^{\mathrm{real}}, \boldsymbol{y}^{\mathrm{real}}, \boldsymbol{w}^{\mathrm{real}}, \boldsymbol{c}^{\mathrm{real}}).$$

Note that the existentially quantified variables can be split at the conjunction into the real and integer part. To perform a similar splitting for the variables bound by the Ramsey quantifier, we have to distinct the two cases whether the vectors of the clique are pairwise distinct in the real components or in the integer components. We only show the case where the vectors of the clique are pairwise distinct in both the real and integer components. The other cases are similar by allowing that either the integer or real components do not change throughout the clique, i.e. either $\exists \boldsymbol{x}^{\mathrm{int}}, \boldsymbol{w}^{\mathrm{int}} \colon \alpha_i(\boldsymbol{x}^{\mathrm{int}}, \boldsymbol{x}^{\mathrm{int}}, \boldsymbol{w}^{\mathrm{int}}, \boldsymbol{c}^{\mathrm{int}})$ or $\exists \boldsymbol{x}^{\mathrm{real}}, \boldsymbol{w}^{\mathrm{real}} \colon \beta_i(\boldsymbol{x}^{\mathrm{real}}, \boldsymbol{x}^{\mathrm{real}}, \boldsymbol{w}^{\mathrm{real}}, \boldsymbol{c}^{\mathrm{real}})$ holds. So we assume that

$$\begin{aligned} &\exists^{\mathrm{ram}} \boldsymbol{x}^{\mathrm{int}}, \boldsymbol{y}^{\mathrm{int}} \colon \quad \exists \boldsymbol{w}^{\mathrm{int}} \colon \ \alpha_i(\boldsymbol{x}^{\mathrm{int}}, \boldsymbol{y}^{\mathrm{int}}, \boldsymbol{w}^{\mathrm{int}}, \boldsymbol{c}^{\mathrm{int}}) \wedge \\ &\exists^{\mathrm{ram}} \boldsymbol{x}^{\mathrm{real}}, \boldsymbol{y}^{\mathrm{real}} \colon \exists \boldsymbol{w}^{\mathrm{real}} \colon \beta_i(\boldsymbol{x}^{\mathrm{real}}, \boldsymbol{y}^{\mathrm{real}}, \boldsymbol{w}^{\mathrm{real}}, \boldsymbol{c}^{\mathrm{real}}). \end{aligned}$$

By applying Theorem 5.2.2 to the first conjunct and Theorem 5.2.5 to the second conjunct, we get

$$\begin{aligned}\exists^{\mathrm{ram}}(\boldsymbol{x}^{\mathrm{int}}, \boldsymbol{v}_1^{\mathrm{int}}, \boldsymbol{v}_2^{\mathrm{int}}), (\boldsymbol{y}^{\mathrm{int}}, \boldsymbol{w}_1^{\mathrm{int}}, \boldsymbol{w}_2^{\mathrm{int}}) \colon \quad & \alpha_i(\boldsymbol{x}^{\mathrm{int}}, \boldsymbol{y}^{\mathrm{int}}, \boldsymbol{v}_1^{\mathrm{int}} + \boldsymbol{w}_2^{\mathrm{int}}, \boldsymbol{c}^{\mathrm{int}}) \wedge \boldsymbol{x}^{\mathrm{int}} \neq \boldsymbol{y}^{\mathrm{int}} \wedge \\ \exists^{\mathrm{ram}}(\boldsymbol{x}^{\mathrm{real}}, \boldsymbol{v}_1^{\mathrm{real}}, \boldsymbol{v}_2^{\mathrm{real}}), (\boldsymbol{y}^{\mathrm{real}}, \boldsymbol{w}_1^{\mathrm{real}}, \boldsymbol{w}_2^{\mathrm{real}}) \colon & \beta_i(\boldsymbol{x}^{\mathrm{real}}, \boldsymbol{y}^{\mathrm{real}}, \boldsymbol{v}_1^{\mathrm{real}} + \boldsymbol{w}_2^{\mathrm{real}}, \boldsymbol{c}^{\mathrm{real}}) \wedge \\ & \boldsymbol{x}^{\mathrm{real}} \neq \boldsymbol{y}^{\mathrm{real}}.\end{aligned}$$

This implies that $\boldsymbol{c}$ satisfies the formula in (ii) by adding the two infinite cliques componentwise.

5.3 Eliminating Ramsey Quantifiers in Linear Integer Arithmetic

In this section we describe our procedure to eliminate the Ramsey quantifier if applied to an existential Presburger formula.

Theorem 5.3.1. *Given an existential Presburger formula $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ with $|\boldsymbol{x}| = |\boldsymbol{y}|$, one can construct in polynomial time an existential Presburger formula of linear size that is equivalent to $\exists^{\mathrm{ram}}\boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$.*

To prove Theorem 5.3.1, one could try to compute a Presburger formula $\psi(\boldsymbol{x}, \boldsymbol{z})$ such that for every valuation $\boldsymbol{c}$ of $\boldsymbol{z}$ the set $\{\boldsymbol{a} \mid \psi(\boldsymbol{a}, \boldsymbol{c})\}$ forms a clique in $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c})$ if one exists. This, however, turns out to be impossible: There exist Presburger formulas that have infinite cliques, but none of them is definable in Presburger arithmetic. For example, consider the formula $\varphi(x, y) := y \geq 2x \wedge x \geq 1$. Every infinite clique $A = \{a_0, a_1, \dots\}$ of φ with $a_0 \leq a_1 \leq \dots$ must satisfy $a_i \geq 2^i \cdot a_0$ for every $i \geq 1$ and thus cannot be ultimately periodic (i.e. there do not exist $n, k \in \mathbb{N}$ such that for all $a \geq n$ we have $a \in A$ if and only if $a + k \in A$). Since a subset of $\mathbb{N}$ is Presburger-definable if and only if it is ultimately periodic [GS66a], it follows that A is not Presburger-definable. To circumvent this problem, our strategy is to define a Presburger formula that only expresses conditions that are equivalent to the existence of an infinite clique in φ without defining the actual clique.

Let us first assume that $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ is a conjunction of the form

$$\bigwedge_{i=1}^{n} \boldsymbol{r}_i \cdot \boldsymbol{x} < \boldsymbol{s}_i \cdot \boldsymbol{y} + \boldsymbol{t}_i \cdot \boldsymbol{z} + h_i \quad \wedge \quad \bigwedge_{j=1}^{m} \boldsymbol{u}_j \cdot \boldsymbol{x} \approx_{e_j}^{j} \boldsymbol{v}_j \cdot \boldsymbol{y} + \boldsymbol{w}_j \cdot \boldsymbol{z} + d_j \tag{5.1}$$

where $\approx_{e_j}^{j} \in \{\equiv_{e_j}, \not\equiv_{e_j}\}$. It should be noted that since modulo constraints are expressible with existential quantifiers and Theorem 5.2.1 allows us to eliminate existential quantifiers under the Ramsey quantifier without introducing modulo constraints, it would even suffice to treat the case where φ has no modulo constraints. However, in practice it might be useful to be able to treat modulo constraints without first trading them in for existential quantifiers. For this reason, we describe the translation in the presence of modulo constraints.

5.3.1 Cliques in Terms of Profiles

Our goal is to construct an existential Presburger formula $\varphi'(\boldsymbol{z})$ so that $\varphi'(\boldsymbol{c})$ holds if and only if there exists an infinite sequence $(\boldsymbol{a}_i)_{i\geq 1}$ of pairwise distinct vectors such that $\varphi(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{c})$ for all $i < j$. As mentioned above, it is possible that such a sequence exists, but that none of them is definable in Presburger arithmetic. Therefore, our first step is to modify the condition "$\varphi(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{c})$ for all $i < j$" into a different condition such that (i) the new condition is equivalent in terms of existence of a sequence and (ii) the new condition can always be satisfied by an arithmetic progression. Property (ii) will allow us to express the new condition in Presburger arithmetic.

Example 5.3.2. To illustrate the idea, suppose that

$$\varphi(\boldsymbol{x}, \boldsymbol{y}, z) := y_1 > 2 \cdot x_1 \wedge x_1 + x_2 < z$$

where $|\boldsymbol{x}| = |\boldsymbol{y}| = 2$. As mentioned above, any infinite clique of φ must grow exponentially in the first component. However, such an infinite clique with respect to $c \in \mathbb{Z}$ exists if and only if there exists a sequence $(\boldsymbol{a}_i)_{i\geq 1}$ with $\boldsymbol{a}_i = (a_{i,1}, a_{i,2}) \in \mathbb{Z}^2$ such that $a_{i,1} + a_{i,2} < c$ for all $i \geq 1$ and the sequence of numbers $a_{1,1}, a_{2,1}, \ldots$ in the first components grows unboundedly: Clearly, any infinite clique of φ must satisfy this. Conversely, any infinite sequence that grows unboundedly in the first component has an infinite subsequence that grows exponentially in the first component. Note that the unboundedness condition can always be satisfied by an arithmetic progression, i.e. a sequence of the form $\boldsymbol{a}_0, \boldsymbol{a}_0 + \boldsymbol{a}, \boldsymbol{a}_0 + 2\boldsymbol{a}, \ldots$, by choosing $a_1 > 0$. This means by guessing the vectors $\boldsymbol{a}_0, \boldsymbol{a}$ using existential quantifiers, we can state $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, z)$ equivalently with the existential Presburger formula

$$\exists \boldsymbol{x}_0, \boldsymbol{x} \colon x_1 > 0 \wedge x_{0,1} + x_{0,2} < z \wedge x_1 + x_2 \leq 0$$

where $\boldsymbol{x}_0 = (x_{0,1}, x_{0,2})$.

These modified conditions on sequences are based on the notion of profiles. Essentially, a profile captures how for a sequence $(\boldsymbol{a}_k)_{k\geq 1}$ the values $\boldsymbol{r}_i \cdot \boldsymbol{a}_k$ and $\boldsymbol{s}_i \cdot \boldsymbol{a}_k + \boldsymbol{t}_i \cdot \boldsymbol{c} + h_i$ evolve for every $i \in [1, n]$. A *profile* (for φ) is a vector in $\mathbb{Z}_\omega^{2n}$ where $\mathbb{Z}_\omega := \mathbb{Z} \cup \{\omega\}$. Let $\boldsymbol{p} = (p_1, \ldots, p_{2n})$ be a profile. Then value p_{2i-1} being an integer means that the sequence $(\boldsymbol{r}_i \cdot \boldsymbol{a}_k)_{k\geq 1}$ is bounded from above by p_{2i-1}. If p_{2i-1} is ω, then the sequence $(\boldsymbol{r}_i \cdot \boldsymbol{a}_k)_{k\geq 1}$ tends to infinity. Similarly, even-indexed entries p_{2i} describe the evolution of the sequence $(\boldsymbol{s}_i \cdot \boldsymbol{a}_k + \boldsymbol{t}_i \cdot \boldsymbol{c} + h_i)_{k\geq 1}$.

Let us make this precise. If $\boldsymbol{p}$ is a profile and $\boldsymbol{c} \in \mathbb{Z}^{|\boldsymbol{z}|}$ is a vector, then a sequence $(\boldsymbol{a}_k)_{k\geq 1}$ of pairwise distinct vectors over $\mathbb{Z}$ is *compatible with* $\boldsymbol{p}$ *for* $\boldsymbol{c}$ if $\boldsymbol{u}_j \cdot \boldsymbol{a}_k \approx_{e_j}^{j} \boldsymbol{v}_j \cdot \boldsymbol{a}_\ell + \boldsymbol{w}_j \cdot \boldsymbol{c} + d_j$ for all $k < \ell$ and $j \in [1, m]$ and

$$\sup(\boldsymbol{r}_i \cdot \boldsymbol{a}_k)_{k\geq 1} \leq p_{2i-1}, \qquad p_{2i} \leq \liminf(\boldsymbol{s}_i \cdot \boldsymbol{a}_k + \boldsymbol{t}_i \cdot \boldsymbol{c} + h_i)_{k\geq 1} \tag{5.2}$$

for all $i \in [1, n]$. A profile $\boldsymbol{p} = (p_1, \ldots, p_{2n})$ is *admissible* if for every $i \in [1, n]$ we have $p_{2i-1} < p_{2i}$ or $p_{2i} = \omega$.

Lemma 5.3.3. *The formula $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ has an infinite clique with respect to $\boldsymbol{c} \in \mathbb{Z}^{|\boldsymbol{z}|}$ if and only if there exists an admissible profile $\boldsymbol{p}$ such that there is a sequence compatible with $\boldsymbol{p}$ for $\boldsymbol{c}$.*

Proof. We begin with the "only if" direction. To ease notation, we write $f_i(\boldsymbol{x}) := \boldsymbol{r}_i \cdot \boldsymbol{x}$ and $g_i(\boldsymbol{x}) := \boldsymbol{s}_i \cdot \boldsymbol{x} + \boldsymbol{t}_i \cdot \boldsymbol{c} + h_i$ for all $i \in [1, n]$. Suppose $(\boldsymbol{a}_k)_{k \geq 1}$ is an infinite clique of φ with respect to $\boldsymbol{c}$. First observe that we may assume that if for some $i \in [1, n]$ the sequence $(f_i(\boldsymbol{a}_k))_{k \geq 1}$ is bounded from above, then for its maximum M we have $M < g_i(\boldsymbol{a}_k)$ for every $k \geq 1$. Indeed, if this is not the case, we can achieve it by removing an initial segment of $(\boldsymbol{a}_k)_{k \geq 1}$. Now, we define the profile $\boldsymbol{p} = (p_1, \ldots, p_{2n})$ as

$$p_{2i-1} := \sup(f_i(\boldsymbol{a}_k))_{k \geq 1}, \qquad p_{2i} := \liminf(g_i(\boldsymbol{a}_k))_{k \geq 1}$$

for all $i \in [1, n]$. Observe that p_{2i} cannot be $-\omega$ and thus belongs to $\mathbb{Z}_\omega$: This is because the sequence $(g_i(\boldsymbol{a}_k))_{k \geq 1}$ is bounded from below by $\min\{f_i(\boldsymbol{a}_1), g_i(\boldsymbol{a}_1)\}$. Then $\boldsymbol{p}$ is admissible since otherwise, we would have $p_{2i-1} \geq p_{2i}$ and $p_{2i} \in \mathbb{Z}$ for some $i \in [1, n]$ implying that there are $k < \ell$ with $f_i(\boldsymbol{a}_k) \geq g_i(\boldsymbol{a}_\ell)$, which contradicts the fact that $(\boldsymbol{a}_k)_{k \geq 1}$ is an infinite clique of φ with respect to $\boldsymbol{c}$. Moreover, by definition of $\boldsymbol{p}$, the sequence $(\boldsymbol{a}_k)_{k \geq 1}$ is clearly compatible with $\boldsymbol{p}$ for $\boldsymbol{c}$.

Let us now prove the "if" direction. Let $\boldsymbol{p} \in \mathbb{Z}_\omega^{2n}$ be an admissible profile and $(\boldsymbol{a}_k)_{k \geq 1}$ be a sequence compatible with $\boldsymbol{p}$ for $\boldsymbol{c}$. Then, we know that for any $k < \ell$ we have $\boldsymbol{u}_j \cdot \boldsymbol{a}_k \approx^j_{e_j} \boldsymbol{v}_j \cdot \boldsymbol{a}_\ell + \boldsymbol{w}_j \cdot \boldsymbol{c} + d_j$. We claim that we can choose an infinite subsequence of $(\boldsymbol{a}_k)_{k \geq 1}$ such that $f_i(\boldsymbol{a}_k) < g_i(\boldsymbol{a}_\ell)$ for every $k < \ell$ and $i \in [1, n]$. It suffices to do this for each $i \in [1, n]$ individually since the property is preserved by taking infinite subsequences. Likewise, picking an infinite subsequence does not spoil the property of being compatible with $\boldsymbol{p}$ for $\boldsymbol{c}$. So consider some $i \in [1, n]$. We distinguish two cases, namely whether $p_{2i} \in \mathbb{Z}$ or $p_{2i} = \omega$. First, suppose $p_{2i} \in \mathbb{Z}$. Then since $\boldsymbol{p}$ is admissible, we have $p_{2i-1} < p_{2i}$. Now, compatibility implies that $p_{2i-1} < p_{2i} \leq g_i(\boldsymbol{a}_\ell)$ for almost all $\ell \geq 1$. Hence, by removing some initial segment of $(\boldsymbol{a}_k)_{k \geq 1}$, we can ensure that $f_i(\boldsymbol{a}_k) < g_i(\boldsymbol{a}_\ell)$ for every $k < \ell$. Now suppose $p_{2i} = \omega$. We successively choose indices $1 \leq i_1 < i_2 < \ldots$ such that $f_i(\boldsymbol{a}_{i_k}) < g_i(\boldsymbol{a}_{i_\ell})$ for every $k < \ell$. Let $i_1 := 1$. Suppose we have already chosen $i_1, \ldots, i_h$ for some $h \geq 1$. Let $M := \max\{f_i(\boldsymbol{a}_{i_k}) \mid 1 \leq k \leq h\}$. By compatibility of $(\boldsymbol{a}_k)_{k \geq 1}$ and since $p_{2i} = \omega$, there exists an $\ell > i_h$ with $M < g_i(\boldsymbol{a}_\ell)$, which allows us to choose $i_{h+1} := \ell$. This completes the construction of an infinite clique of φ with respect to $\boldsymbol{c}$. □

5.3.2 Compatibility in Terms of Matrices

Our next step is to express the existence of a sequence compatible with $\boldsymbol{p}$ for $\boldsymbol{c}$ in terms of certain inequalities. To this end, we define two matrices $\boldsymbol{A}_{\boldsymbol{p},\boldsymbol{c}}$ and $\boldsymbol{B}_{\boldsymbol{p}}$ and a vector $\boldsymbol{b}_{\boldsymbol{p},\boldsymbol{c}}$. Here, $\boldsymbol{A}_{\boldsymbol{p},\boldsymbol{c}} \cdot \boldsymbol{x} \geq \boldsymbol{b}_{\boldsymbol{p},\boldsymbol{c}}$ will express the compatibility conditions involving those p_{2i-1} and p_{2i} that are integers. Thus, we define $\boldsymbol{A}_{\boldsymbol{p},\boldsymbol{c}}$ and $\boldsymbol{b}_{\boldsymbol{p},\boldsymbol{c}}$ by describing the system of inequalities $\boldsymbol{A}_{\boldsymbol{p},\boldsymbol{c}} \cdot \boldsymbol{x} \geq \boldsymbol{b}_{\boldsymbol{p},\boldsymbol{c}}$. For every $i \in [1, n]$ with $p_{2i-1} \in \mathbb{Z}$ we add the inequality $\boldsymbol{r}_i \cdot \boldsymbol{x} \leq p_{2i-1}$, i.e. we add the row $-\boldsymbol{r}_i$ to $\boldsymbol{A}_{\boldsymbol{p},\boldsymbol{c}}$ and the entry $-p_{2i-1}$ to $\boldsymbol{b}_{\boldsymbol{p},\boldsymbol{c}}$. Furthermore, for every $i \in [1, n]$ with $p_{2i} \in \mathbb{Z}$ we add the inequality $p_{2i} \leq \boldsymbol{s}_i \cdot \boldsymbol{x} + \boldsymbol{t}_i \cdot \boldsymbol{c} + h_i$, i.e. we add the row $\boldsymbol{s}_i$ to $\boldsymbol{A}_{\boldsymbol{p},\boldsymbol{c}}$ and the entry $p_{2i} - \boldsymbol{t}_i \cdot \boldsymbol{c} - h_i$ to $\boldsymbol{b}_{\boldsymbol{p},\boldsymbol{c}}$.

Moreover, $\boldsymbol{B_p}$ will be used to express the unboundedness condition on the right side of Equation (5.2) if $p_{2i} = \omega$. Thus, for every $i \in [1, n]$ with $p_{2i} = \omega$ we add the row $\boldsymbol{s}_i$ to $\boldsymbol{B_p}$. We say that a function $f\colon \mathbb{Z}^r \to \mathbb{Z}^s$ is *simultaneously unbounded* on a sequence $(\boldsymbol{a}_k)_{k\geq 1}$ in $\mathbb{Z}^r$ if for every $h \in \mathbb{Z}$ we have $f(\boldsymbol{a}_k) \geq (h, \dots, h)$ for almost all $k \geq 1$. Now observe the following:

Lemma 5.3.4. *Let $\boldsymbol{p} \in \mathbb{Z}_\omega^{2n}$ be a profile for $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ and $\boldsymbol{c} \in \mathbb{Z}^{|\boldsymbol{z}|}$. Then there exists a sequence compatible with $\boldsymbol{p}$ for $\boldsymbol{c}$ if and only if there exists a sequence $(\boldsymbol{a}_k)_{k\geq 1}$ of pairwise distinct vectors such that*

(i) $\boldsymbol{u}_j \cdot \boldsymbol{a}_k \approx_{e_j}^{j} \boldsymbol{v}_j \cdot \boldsymbol{a}_\ell + \boldsymbol{w}_j \cdot \boldsymbol{c} + d_j$ *for every $k < \ell$ and $j \in [1, m]$,*

(ii) $\boldsymbol{A}_{\boldsymbol{p},\boldsymbol{c}} \cdot \boldsymbol{a}_k \geq \boldsymbol{b}_{\boldsymbol{p},\boldsymbol{c}}$ *for every $k \geq 1$, and*

(iii) $\boldsymbol{B_p}$ *is simultaneously unbounded on $(\boldsymbol{a}_k)_{k\geq 1}$.*

Proof. The "if" direction holds by construction since $\boldsymbol{A}_{\boldsymbol{p},\boldsymbol{c}} \cdot \boldsymbol{a}_k \geq \boldsymbol{b}_{\boldsymbol{p},\boldsymbol{c}}$ ensures that Equation (5.2) holds if p_{2i-1} and p_{2i} are integers and simultaneously unboundedness of $\boldsymbol{B_p}$ ensures that $(\boldsymbol{s}_i \cdot \boldsymbol{a}_k + \boldsymbol{t}_i \cdot \boldsymbol{c} + h_i)_{k\geq 1}$ grows unboundedly if $p_{2i} = \omega$.

For the "only if" direction let $(\boldsymbol{a}_k)_{k\geq 1}$ be a sequence that is compatible with $\boldsymbol{p}$ for $\boldsymbol{c}$. Condition (i) is satisfied by definition. For every $k \geq 1$ the inequalities in $\boldsymbol{A}_{\boldsymbol{p},\boldsymbol{c}} \cdot \boldsymbol{a}_k \geq \boldsymbol{b}_{\boldsymbol{p},\boldsymbol{c}}$ of the form $\boldsymbol{r}_i \cdot \boldsymbol{a}_k \leq p_{2i-1}$ are fulfilled since $\sup(\boldsymbol{r}_i \cdot \boldsymbol{a}_k)_{k\geq 1} \leq p_{2i-1}$. Moreover, the inequalities in $\boldsymbol{A}_{\boldsymbol{p},\boldsymbol{c}} \cdot \boldsymbol{a}_k \geq \boldsymbol{b}_{\boldsymbol{p},\boldsymbol{c}}$ of the form $p_{2i} \leq \boldsymbol{s}_i \cdot \boldsymbol{a}_k + \boldsymbol{t}_i \cdot \boldsymbol{c} + h_i$ are satisfied for almost all $k \geq 1$ since $p_{2i} \leq \liminf(\boldsymbol{s}_i \cdot \boldsymbol{a}_k + \boldsymbol{t}_i \cdot \boldsymbol{c} + h_i)_{k\geq 1}$. Thus, by replacing $(\boldsymbol{a}_k)_{k\geq 1}$ with an infinite subsequence, condition (ii) is satisfied. Finally, $\boldsymbol{B_p}$ is simultaneously unbounded on $(\boldsymbol{a}_k)_{k\geq 1}$ since for every row $\boldsymbol{s}_i$ of $\boldsymbol{B_p}$ we have that $p_{2i} = \omega$ and therefore $(\boldsymbol{s}_i \cdot \boldsymbol{a}_k + \boldsymbol{t}_i \cdot \boldsymbol{c} + h_i)_{k\geq 1}$ grows unboundedly. □

5.3.3 Arithmetic Progressions

The last key step is to show that there exists a sequence compatible with $\boldsymbol{p}$ for $\boldsymbol{c}$ if and only if there exists such a sequence of the form $\boldsymbol{a}_0, \boldsymbol{a}_0 + \boldsymbol{a}, \boldsymbol{a}_0 + 2\boldsymbol{a}, \dots$ for vectors $\boldsymbol{a}_0, \boldsymbol{a}$, called *arithmetic progression*. This will allow us to express existence of a sequence by the existence of suitable vectors $\boldsymbol{a}_0$ and $\boldsymbol{a}$.

Lemma 5.3.5. *Let $\boldsymbol{p} \in \mathbb{Z}_\omega^{2n}$ be a profile for $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ and $\boldsymbol{c} \in \mathbb{Z}^{|\boldsymbol{z}|}$. Then there exists a sequence compatible with $\boldsymbol{p}$ for $\boldsymbol{c}$ if and only if there are vectors $\boldsymbol{a}_0, \boldsymbol{a} \in \mathbb{Z}^{|\boldsymbol{x}|}$ with $\boldsymbol{a} \neq \boldsymbol{0}$ such that*

(i) $\boldsymbol{u}_j \cdot \boldsymbol{a}_0 \approx_{e_j}^{j} \boldsymbol{v}_j \cdot (\boldsymbol{a}_0 + \boldsymbol{a}) + \boldsymbol{w}_j \cdot \boldsymbol{c} + d_j$ *and* $\boldsymbol{u}_j \cdot \boldsymbol{a} \equiv_{e_j} \boldsymbol{v}_j \cdot \boldsymbol{a} \equiv_{e_j} 0$ *for all $j \in [1, m]$,*

(ii) $\boldsymbol{A}_{\boldsymbol{p},\boldsymbol{c}} \cdot \boldsymbol{a}_0 \geq \boldsymbol{b}_{\boldsymbol{p},\boldsymbol{c}}$ *and* $\boldsymbol{A}_{\boldsymbol{p},\boldsymbol{c}} \cdot \boldsymbol{a} \geq \boldsymbol{0}$, *and*

(iii) $\boldsymbol{B_p} \cdot \boldsymbol{a} \gg \boldsymbol{0}$.

Proof. We begin with the "if" direction. Suppose there are vectors $\boldsymbol{a}_0$ and $\boldsymbol{a}$ as described. Then we claim that the sequence $(\boldsymbol{a}_k)_{k\geq 1}$ with $\boldsymbol{a}_k := \boldsymbol{a}_0 + k\boldsymbol{a}$ is compatible with $\boldsymbol{p}$ for $\boldsymbol{c}$. We use Lemma 5.3.4 to show this. First note that since $\boldsymbol{a} \neq \boldsymbol{0}$, the $\boldsymbol{a}_k$ are pairwise

distinct. It is clear that the sequence satisfies conditions (i) and (ii) of Lemma 5.3.4. Condition (iii) holds as well because in the vector $\boldsymbol{B_p} \cdot (k\boldsymbol{a})$ every entry is at least k. Thus, $\boldsymbol{B_p}$ is simultaneously unbounded on $(\boldsymbol{a}_k)_{k\geq 1}$.

For the "only if" direction assume that $(\boldsymbol{a}_k)_{k\geq 1}$ is a sequence of pairwise distinct vectors that satisfies the conditions in Lemma 5.3.4. Since there are only finitely many possible remainders modulo e_j of the expressions $\boldsymbol{u}_j \cdot \boldsymbol{a}_k$ and $\boldsymbol{v}_j \cdot \boldsymbol{a}_k$, we can pick a subsequence such that for each $j \in [1, m]$ the maps $k \mapsto \boldsymbol{u}_j \cdot \boldsymbol{a}_k$ and $k \mapsto \boldsymbol{v}_j \cdot \boldsymbol{a}_k$ are constant modulo e_j. In a second step, we notice that since $\boldsymbol{A_{p,c}} \cdot \boldsymbol{a}_k \geq \boldsymbol{b_{p,c}}$ for all $k \geq 1$, the set $\{\boldsymbol{A_{p,c}} \cdot \boldsymbol{a}_k \mid k \geq 1\}$ together with $\leq$ forms a well-partial-ordering. Thus, we can pick an infinite subsequence of $(\boldsymbol{a}_k)_{k\geq 1}$ so that $\boldsymbol{A_{p,c}} \cdot \boldsymbol{a}_1 \leq \boldsymbol{A_{p,c}} \cdot \boldsymbol{a}_2 \leq \ldots$. Note that passing to infinite subsequences does not spoil the conditions of Lemma 5.3.4. Thus, $\boldsymbol{B_p}$ is still simultaneously unbounded on $(\boldsymbol{a}_k)_{k\geq 1}$. This allows us to pick a subsequence so that also $\boldsymbol{B_p} \cdot \boldsymbol{a}_1 \ll \boldsymbol{B_p} \cdot \boldsymbol{a}_2 \ll \ldots$. Therefore, if we set $\boldsymbol{a}_0 := \boldsymbol{a}_1$ and $\boldsymbol{a} := \boldsymbol{a}_2 - \boldsymbol{a}_1$, then $\boldsymbol{a}_0$ and $\boldsymbol{a}$ are as desired. □

5.3.4 Constructing the Formula

We are now ready to prove Theorem 5.3.1. To this end, assume that $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ with $|\boldsymbol{x}| = |\boldsymbol{y}|$ is an arbitrary existential Presburger formula. Using Theorem 5.2.1, we can assume that φ is quantifier-free. By moving all negations inwards to the atoms and possibly negating them, we may further assume that φ is a positive Boolean combination of inequality atoms $\alpha_i := \boldsymbol{r}_i \cdot \boldsymbol{x} < \boldsymbol{s}_i \cdot \boldsymbol{y} + \boldsymbol{t}_i \cdot \boldsymbol{z} + h_i$ for $i \in [1, n]$ and modulo constraint atoms $\beta_j := \boldsymbol{u}_j \cdot \boldsymbol{x} \approx^j_{e_j} \boldsymbol{v}_j \cdot \boldsymbol{y} + \boldsymbol{w}_j \cdot \boldsymbol{z} + d_j$ with $\approx^j_{e_j} \in \{\equiv_{e_j}, \not\equiv_{e_j}\}$ for $j \in [1, m]$. (Note that in Presburger arithmetic equality can be expressed by a conjunction of two strict inequalities.)

The key idea is to guess (using existentially quantified variables) a subset of the atoms in φ and check that (i) satisfying these atoms makes φ true and (ii) the conjunction of these atoms has an infinite clique. Note that there are only finitely many possible conjunctions of atoms from φ and φ is equivalent to the disjunction of these conjunctions. Thus, by Ramsey's theorem, there exists an infinite clique for φ if and only if there exists one for some conjunction of atoms. Condition (i) is easy to state. To check (ii), we then require the conditions of Lemma 5.3.5 to be satisfied for the conjunction of atoms.

For each atom α_i we introduce a variable $q_i^<$ and for each atom β_j we introduce a variable $q_j^{\approx}$. To check that (i) holds, we use the formula φ' that is obtained from φ by replacing each α_i with $q_i^< = 1$ and each β_j with $q_j^{\approx} = 1$ and adding the restrictions $q_i^< = 0 \vee q_i^< = 1$ and $q_j^{\approx} = 0 \vee q_j^{\approx} = 1$. Now, φ is equivalent to

$$\psi := \exists \boldsymbol{q}^<, \boldsymbol{q}^{\approx} \colon \varphi' \wedge \bigwedge_{i=1}^{n} (q_i^< = 1 \to \alpha_i) \wedge \bigwedge_{j=1}^{m} (q_j^{\approx} = 1 \to \beta_j).$$

Let us now construct the formula for condition (ii) above. To this end, we build formulas γ_i and δ_j that state all conditions of Lemma 5.3.5 that stem from the atom α_i

and β_j, respectively. For $i \in [1,n]$ and fresh variables $p_{2i-1}, p_{2i}, \boldsymbol{x}_0, \boldsymbol{x}$ let

$$\begin{aligned}\gamma_i := {} & (p_{2i-1} < \omega \to (\boldsymbol{r}_i \cdot \boldsymbol{x}_0 \leq p_{2i-1} \wedge \boldsymbol{r}_i \cdot \boldsymbol{x} \leq 0)) \wedge \\ & (p_{2i} < \omega \to (p_{2i} \leq \boldsymbol{s}_i \cdot \boldsymbol{x}_0 + \boldsymbol{t}_i \cdot \boldsymbol{z} + h_i \wedge \boldsymbol{s}_i \cdot \boldsymbol{x} \geq 0)) \wedge \\ & (p_{2i} = \omega \to \boldsymbol{s}_i \cdot \boldsymbol{x} > 0)\end{aligned}$$

and for all $j \in [1,m]$ let

$$\delta_j := \boldsymbol{u}_j \cdot \boldsymbol{x}_0 \approx^j_{e_j} \boldsymbol{v}_j \cdot (\boldsymbol{x}_0 + \boldsymbol{x}) + \boldsymbol{w}_j \cdot \boldsymbol{z} + d_j \wedge \boldsymbol{u}_j \cdot \boldsymbol{x} \equiv_{e_j} 0 \wedge \boldsymbol{v}_j \cdot \boldsymbol{x} \equiv_{e_j} 0.$$

Here, $p_\ell < \omega$ and $p_\ell = \omega$ is shorthand notation for $\omega_\ell = 0$ and $\omega_\ell = 1$, respectively, where ω_ℓ is a fresh variable associated with p_ℓ that is restricted to values from $\{0,1\}$. Thus, from now on we implicitly quantify $\boldsymbol{\omega}$ when $\boldsymbol{p}$ is quantified. The following requires $\boldsymbol{p}$ to be an admissible profile:

$$\theta := \bigwedge_{i=1}^{n} (p_{2i-1} < \omega \wedge p_{2i-1} < p_{2i} \vee p_{2i} = \omega)$$

Then we claim that $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ is equivalent to the existential formula

$$\chi := \exists \boldsymbol{q}^<, \boldsymbol{q}^\approx, \boldsymbol{p}, \boldsymbol{x}_0, \boldsymbol{x} \colon \varphi' \wedge \theta \wedge \boldsymbol{x} \neq \boldsymbol{0} \wedge \bigwedge_{i=1}^{n} (q_i^< = 1 \to \gamma_i) \wedge \bigwedge_{j=1}^{m} (q_j^\approx = 1 \to \delta_j).$$

We show that for any $\boldsymbol{c} \in \mathbb{Z}^{|\boldsymbol{z}|}$ we have $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c})$ if and only if $\chi(\boldsymbol{c})$. For a valuation ν of the variables $q_i^<, q_j^\approx$ to $\{0,1\}$ let $I_\nu := \{i \in [1,n] \mid \nu(q_i^<) = 1\}$ and $J_\nu := \{j \in [1,m] \mid \nu(q_j^\approx) = 1\}$. By Ramsey's theorem, we have $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c})$ if and only if there is a valuation ν of the $q_i^<, q_j^\approx$ satisfying φ' such that

$$\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \bigwedge_{i \in I_\nu} \alpha_i(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c}) \wedge \bigwedge_{j \in J_\nu} \beta_j(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c}).$$

By Lemmas 5.3.3 and 5.3.5, this is the case if and only if

$$\exists \boldsymbol{p}, \boldsymbol{x}_0, \boldsymbol{x} \colon \theta \wedge \boldsymbol{x} \neq \boldsymbol{0} \wedge \bigwedge_{i \in I_\nu} \gamma_i(\boldsymbol{p}, \boldsymbol{x}_0, \boldsymbol{x}, \boldsymbol{c}) \wedge \bigwedge_{j \in J_\nu} \delta_j(\boldsymbol{x}_0, \boldsymbol{x}, \boldsymbol{c})$$

which in turn holds for some assignment ν of the $q_i^<, q_j^\approx$ satisfying φ' if and only if $\chi(\boldsymbol{c})$.

To conclude the proof of Theorem 5.3.1, we note that the size of χ is linear in the size of φ and all constructions can be performed in polynomial time.

5.4 Eliminating Ramsey Quantifiers in Linear Real Arithmetic

Let us now consider the elimination of the Ramsey quantifier if applied to an existential formula in LRA. We prove the same result as over Presburger arithmetic.

Theorem 5.4.1. *Given an existential formula $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ in LRA with $|\boldsymbol{x}| = |\boldsymbol{y}|$, one can construct in polynomial time an existential formula in LRA of linear size that is equivalent to $\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$.*

Similar to the integer case, we first assume that φ is a conjunction of the form

$$\bigwedge_{i=1}^{n} \boldsymbol{r}_i \cdot \boldsymbol{x} < \boldsymbol{s}_i \cdot \boldsymbol{y} + \boldsymbol{t}_i \cdot \boldsymbol{z} + h_i \quad \wedge \quad \bigwedge_{j=1}^{m} \boldsymbol{u}_j \cdot \boldsymbol{x} = \boldsymbol{v}_j \cdot \boldsymbol{y} + \boldsymbol{w}_j \cdot \boldsymbol{z} + d_j. \tag{5.3}$$

Here, note that we explicitly deal with equality atoms since over the reals equality cannot be expressed with strict inequalities anymore.

5.4.1 Cliques in Terms of Profiles

We now define the notion of a profile with a similar purpose as in the integer case. In the real case, however, a profile carries more information. In the case of Presburger arithmetic, it is enough to guess whether a particular function grows or has a particular upper bound. Over the reals it is possible that a function grows strictly but is still bounded because it converges.

Example 5.4.2. Let $\varphi(\boldsymbol{x}, \boldsymbol{y}, z) := x_1 < y_1 \wedge x_1 < z \wedge x_2 < y_2 - 1$ where $|\boldsymbol{x}| = |\boldsymbol{y}| = 2$. That is, any infinite clique of φ with respect to $c \in \mathbb{R}$ must strictly grow in the first component but still be bounded by c and grow unboundedly in the second component. Clearly, if φ is interpreted as a Presburger formula, then $\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, z)$ is unsatisfiable since any strictly increasing infinite sequence over the integers must grow unboundedly. Whereas over the reals, an infinite sequence can be strictly increasing and still be bounded by some finite value. Thus, it does not suffice to only consider sequences of the form $\boldsymbol{a}_0, \boldsymbol{a}_0 + \boldsymbol{a}, \boldsymbol{a}_0 + 2\boldsymbol{a}, \dots$ as we did in the Presburger case. To deal with the convergent behavior, we introduce a third vector that is used to subtract smaller and smaller fractions. In this example, for every $c \in \mathbb{R}$ we can find the sequence

$$\boldsymbol{a} - \boldsymbol{d}_{\text{con}} + \boldsymbol{d}_\infty, \boldsymbol{a} - \frac{1}{2}\boldsymbol{d}_{\text{con}} + 2\boldsymbol{d}_\infty, \boldsymbol{a} - \frac{1}{3}\boldsymbol{d}_{\text{con}} + 3\boldsymbol{d}_\infty, \dots \tag{5.4}$$

where $\boldsymbol{a} := (c, 0)$, $\boldsymbol{d}_{\text{con}} := (1, 0)$, and $\boldsymbol{d}_\infty := (0, 2)$ that forms an infinite clique of φ with respect to c. The existence of such a sequence can be checked with the formula

$$\exists \boldsymbol{x}, \boldsymbol{x}_{\text{con}}, \boldsymbol{x}_\infty \colon x_1 \leq z \wedge x_{c,1} > 0 \wedge x_{c,2} = 0 \wedge x_{\infty,1} = 0 \wedge x_{\infty,2} > 0$$

where $\boldsymbol{x}_{\text{con}} = (x_{c,1}, x_{c,2})$ and $\boldsymbol{x}_\infty = (x_{\infty,1}, x_{\infty,2})$.

A *profile* (for φ) is a tuple $\boldsymbol{p} = (\rho, \sigma, t_\rho, t_\sigma)$ of functions where $\rho, \sigma \colon \{1, \dots, n\} \to \mathbb{R} \cup \{-\omega, \omega\}$ and $t_\rho, t_\sigma \colon \{1, \dots, n\} \to \{-\omega, -1, 0, 1, \omega\}$. For a sequence $(\boldsymbol{a}_k)_{k \geq 1}$ of vectors in $\mathbb{R}^{|\boldsymbol{x}|}$ let $\boldsymbol{\rho}_i := (\boldsymbol{r}_i \cdot \boldsymbol{a}_k)_{k \geq 1}$ and $\boldsymbol{\sigma}_i := (\boldsymbol{s}_i \cdot \boldsymbol{a}_k)_{k \geq 1}$ for all $i \in [1, n]$. We say that a sequence $(\boldsymbol{a}_k)_{k \geq 1}$ of pairwise distinct vectors is *compatible* with $\boldsymbol{p}$ if $\rho(i)$ and $\sigma(i)$ are the real values to which the sequences $\boldsymbol{\rho}_i$ and $\boldsymbol{\sigma}_i$ converge or ω (resp. $-\omega$) if the corresponding sequence is strictly increasing (resp. decreasing) and diverges to ∞ (resp. $-\infty$) and

the functions t_ρ and t_σ describe the type of convergence, where type 0 means that the corresponding sequence is constant, type 1 (resp. -1) means that it is strictly increasing (resp. decreasing) and converges from below (resp. above), and the type is ω (resp. $-\omega$) in the divergent case. A profile $\boldsymbol{p}$ is $\boldsymbol{c}$-*admissible* for a vector $\boldsymbol{c} \in \mathbb{R}^{|\boldsymbol{z}|}$ if for all $i \in [1, n]$ we have

- $\sigma(i) \neq -\omega$ and if $\rho(i) = \omega$, then $\sigma(i) = \omega$,
- $\rho(i) < \sigma(i) + \boldsymbol{t}_i \cdot \boldsymbol{c} + h_i$ if either $t_\rho(i) \in \{-1, 0\}$ and $t_\sigma(i) \in \{0, 1\}$ or $t_\rho(i) = -1$ and $t_\sigma(i) = -1$, and
- $\rho(i) \leq \sigma(i) + \boldsymbol{t}_i \cdot \boldsymbol{c} + h_i$ if either $t_\rho(i) = 0$ and $t_\sigma(i) = -1$ or $t_\rho(i) = 1$.

We say that a sequence $(\boldsymbol{a}_k)_{k \geq 1}$ satisfies the equality constraints (of φ) for $\boldsymbol{c} \in \mathbb{R}^{|\boldsymbol{z}|}$ if $\boldsymbol{u}_j \cdot \boldsymbol{a}_k = \boldsymbol{v}_j \cdot \boldsymbol{a}_\ell + \boldsymbol{w}_j \cdot \boldsymbol{c} + d_j$ for all $j \in [1, m]$ and $k < \ell$.

Let $\varphi(\boldsymbol{x}, \boldsymbol{y}, z) := x_1 < y_1 \wedge x_1 < z \wedge x_2 < y_2 - 1$ be the formula from Example 5.4.2 with three inequalities. Here, for $c \in \mathbb{R}$ we define the profile $\boldsymbol{p}$ with

- $\rho(1) := \sigma(1) := c$ and $t_\rho(1) := t_\sigma(1) := 1$,
- $\rho(2) := c$, $t_\rho(2) := 1$, $\sigma(2) := 0$, and $t_\sigma(2) := 0$, and
- $\rho(3) := \sigma(3) := \omega$ and $t_\rho(3) := t_\sigma(3) := \omega$.

Then $\boldsymbol{p}$ is c-admissible and the sequence from Equation (5.4) is compatible with $\boldsymbol{p}$. The following lemma shows that this is equivalent to φ having an infinite clique with respect to c.

Lemma 5.4.3. *The formula $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ has an infinite clique with respect to $\boldsymbol{c} \in \mathbb{R}^{|\boldsymbol{z}|}$ if and only if there exists a $\boldsymbol{c}$-admissible profile $\boldsymbol{p}$ such that there is a sequence compatible with $\boldsymbol{p}$ that satisfies the equality constraints for $\boldsymbol{c}$.*

Proof. We first show the "only if" direction. Let $(\boldsymbol{a}_k)_{k \geq 1}$ be an infinite clique of φ with respect to $\boldsymbol{c}$. For all $i \in [1, n]$ consider the sequence $\boldsymbol{\rho}_i$. By the Bolzano-Weierstrass theorem, if $\boldsymbol{\rho}_i$ is bounded, we can replace $(\boldsymbol{a}_k)_{k \geq 1}$ with an infinite subsequence such that $\boldsymbol{\rho}_i$ converges to a real value $r_i \in \mathbb{R}$. By restricting further to an infinite subsequence, we have that $\boldsymbol{\rho}_i$ is either constant, strictly increasing, or strictly decreasing. Thus, we set $\rho(i) := r_i$ and $t_\rho(i)$ to 0, 1, or -1 depending on whether $\boldsymbol{\rho}_i$ is constant, increasing, or decreasing. If $\boldsymbol{\rho}_i$ is unbounded, we replace $(\boldsymbol{a}_k)_{k \geq 1}$ with an infinite subsequence such that $\boldsymbol{\rho}_i$ is strictly increasing if it is unbounded above and strictly decreasing if it is unbounded below. Then we set $\rho(i)$ and $t_\rho(i)$ to ω or $-\omega$ depending on whether $\boldsymbol{\rho}_i$ is increasing or decreasing. Similarly, we can define $\sigma(i)$ and $t_\sigma(i)$ by considering the sequence $\boldsymbol{\sigma}_i$. Thus, there is a sequence $(\boldsymbol{a}_k)_{k \geq 1}$ that is compatible with the profile $\boldsymbol{p} := (\rho, \sigma, t_\rho, t_\sigma)$. Since $(\boldsymbol{a}_k)_{k \geq 1}$ still satisfies the equality constraints for $\boldsymbol{c}$, it remains to show that $\boldsymbol{p}$ is $\boldsymbol{c}$-admissible. First observe that $\sigma(i) \neq -\omega$ since $\boldsymbol{\sigma}_i$ is bounded from below by $\min\{\boldsymbol{r}_i \cdot \boldsymbol{a}_1, \boldsymbol{s}_i \cdot \boldsymbol{a}_1\}$. If $\rho(i) = \omega$, then also $\sigma(i) = \omega$ since otherwise there were $k < \ell$ such that $\boldsymbol{r}_i \cdot \boldsymbol{a}_k \geq \boldsymbol{s}_i \cdot \boldsymbol{a}_\ell + \boldsymbol{t}_i \cdot \boldsymbol{c} + h_i$. With a similar reasoning we can show that

if $t_\rho(i) \in \{-1, 0\}$ and $t_\sigma(i) \in \{0, 1\}$, then $\rho(i) < \sigma(i) + \boldsymbol{t}_i \cdot \boldsymbol{c} + h_i$, if $t_\rho(i) = t_\sigma(i) = -1$, then $\rho(i) < \sigma(i) + \boldsymbol{t}_i \cdot \boldsymbol{c} + h_i$, and if either $t_\rho(i) = 0$ and $t_\sigma(i) = -1$ or $t_\rho(i) = 1$, then $\rho(i) \leq \sigma(i) + \boldsymbol{t}_i \cdot \boldsymbol{c} + h_i$.

We now turn to the "if" direction. Let $\boldsymbol{p} = (\rho, \sigma, t_\rho, t_\sigma)$ be a $\boldsymbol{c}$-admissible profile and $(\boldsymbol{a}_k)_{k\geq 1}$ be a sequence compatible with $\boldsymbol{p}$ that satisfies the equality constraints for $\boldsymbol{c}$. To ease notation, we define the functions $f_i(\boldsymbol{x}) := \boldsymbol{r}_i \cdot \boldsymbol{x}$ and $g_i(\boldsymbol{x}) := \boldsymbol{s}_i \cdot \boldsymbol{x} + \boldsymbol{t}_i \cdot \boldsymbol{c} + h_i$ for all $i \in [1, n]$. We successively restrict for each $i \in [1, n]$ to a subsequence such that $f_i(\boldsymbol{a}_k) < g_i(\boldsymbol{a}_\ell)$ for all $k < \ell$. To this end, we inductively choose indices $1 \leq i_1 < i_2 < \ldots$ that define the desired subsequence $(\boldsymbol{a}_{i_k})_{k\geq 1}$. Let $i_1 := 1$. Suppose we have already chosen the indices $i_1, \ldots, i_h$ for some $h \geq 1$ such that $f_i(\boldsymbol{a}_{i_k}) < g_i(\boldsymbol{a}_{i_\ell})$ for all $1 \leq k < \ell \leq h$. Let $M := \max\{f_i(\boldsymbol{a}_k) \mid 1 \leq k \leq h\}$. If $\rho(i) < \sigma(i) + \boldsymbol{t}_i \cdot \boldsymbol{c} + h_i$, then clearly there exists an $i_{h+1} > i_h$ such that $M < g_i(\boldsymbol{a}_{i_{h+1}})$. If $\rho(i) = \sigma(i) + \boldsymbol{t}_i \cdot \boldsymbol{c} + h_i$, then by definition of $\boldsymbol{c}$-admissibility, we have that either $t_\rho(i) = 0$ and $t_\sigma(i) = -1$ or $t_\sigma(i) = 1$, or $\rho(i) = \sigma(i) = \omega$. In all of these cases we can find an index $i_{h+1} > i_h$ such that $M < g_i(\boldsymbol{a}_{i_{h+1}})$. Thus, we can extend the sequence of indices with i_{h+1}. Finally, note that passing to subsequences does not spoil the satisfaction of the equality constraints for $\boldsymbol{c}$. Thus, the constructed subsequence $(\boldsymbol{a}_{i_k})_{k\geq 1}$ is an infinite clique of φ with respect to $\boldsymbol{c}$. □

5.4.2 A General Form of Cliques

In the case of Presburger arithmetic, a key insight was that if there exists a clique compatible with a profile, then there exists one of the form $\boldsymbol{a}_0, \boldsymbol{a}_0 + \boldsymbol{a}, \boldsymbol{a}_0 + 2 \cdot \boldsymbol{a}, \ldots$. The real linear arithmetic case is more involved in this regard: There are profiles with which no arithmetic progression is compatible. Above we have already seen such a profile $\boldsymbol{p}$ for the formula in Example 5.4.2. Here, every sequence compatible with $\boldsymbol{p}$ must be strictly increasing and converging from below to a real value $c \in \mathbb{R}$ in the first component and strictly increasing and unbounded in the second component. Clearly, no arithmetic progression $\boldsymbol{a}_0, \boldsymbol{a}_0 + \boldsymbol{a}, \boldsymbol{a}_0 + 2 \cdot \boldsymbol{a}, \ldots$ has these properties since the entry in the first component of $\boldsymbol{a}$ would have to be positive which implies that it would tend to infinity. However, we will prove that if a compatible sequence exists, then we can always find one of a similar form as the sequence in Equation (5.4). That is, a sequence $(\boldsymbol{a}_k)_{k\geq 1}$ with

$$\boldsymbol{a}_k = \boldsymbol{a} - \tfrac{1}{k}\boldsymbol{d}_{\text{con}} + k\boldsymbol{d}_\infty \tag{5.5}$$

for some real vectors $\boldsymbol{a}$, $\boldsymbol{d}_{\text{con}}$, and $\boldsymbol{d}_\infty$. Here, the vector $\boldsymbol{d}_{\text{con}}$ realizes the convergence behavior: By subtracting smaller and smaller fractions of it from $\boldsymbol{a}$, the part $\boldsymbol{a} - \frac{1}{k}\boldsymbol{d}_{\text{con}}$ converges to $\boldsymbol{a}$. Whereas the vector $\boldsymbol{d}_\infty$ realizes divergence to $\pm\infty$: By adding larger and larger multiples of it, we can make sure that the sequence grow unboundedly in certain components.

Our goal is to formulate sufficient conditions on vectors $\boldsymbol{a}$, $\boldsymbol{d}_{\text{con}}$, and $\boldsymbol{d}_\infty$ such that the sequence in Equation (5.5) is compatible with a profile and satisfies the equality constraints of φ. We then show that also the converse holds, i.e. if there exists a compatible sequence, then there is one of the form as in Equation (5.5) where the defining vectors satisfy the conditions. These conditions will be expressible in terms of linear

(in)equalities, which allows us to construct an existential formula in LRA that states the existence of an infinite clique.

5.4.3 Extracting $\boldsymbol{a}$ and $\boldsymbol{d}_\infty$

Before we formulate the conditions on the vectors $\boldsymbol{a}$, $\boldsymbol{d}_{\text{con}}$, and $\boldsymbol{d}_\infty$, we present the key lemma that will yield the existence of $\boldsymbol{a}$ and $\boldsymbol{d}_\infty$ with the desired properties. Suppose we are given a sequence $(\boldsymbol{a}_k)_{k\geq 1}$ in $\mathbb{R}^d$ where for some linear maps $\boldsymbol{A}\colon \mathbb{R}^d \to \mathbb{R}^m$ and $\boldsymbol{B}\colon \mathbb{R}^d \to \mathbb{R}^n$ the sequence $(\boldsymbol{A}\cdot\boldsymbol{a}_k)_{k\geq 1}$ converges to some $\boldsymbol{v}\in\mathbb{R}^d$ and the sequence $(\boldsymbol{B}\cdot\boldsymbol{a}_k)_{k\geq 1}$ is simultaneously unbounded. If this sequence is of the form as in Equation (5.5), then $\boldsymbol{a}$ and $\boldsymbol{d}_\infty$ must satisfy (i) $\boldsymbol{A}\cdot\boldsymbol{a}=\boldsymbol{v}$, (ii) $\boldsymbol{A}\cdot\boldsymbol{d}_\infty=\boldsymbol{0}$, and (iii) $\boldsymbol{B}\cdot\boldsymbol{d}_\infty\gg\boldsymbol{0}$. Indeed, we must have that $\boldsymbol{A}\cdot\boldsymbol{d}_\infty=\boldsymbol{0}$ since if $\boldsymbol{A}\cdot\boldsymbol{d}_\infty$ has a non-zero entry, then the sequence $(\boldsymbol{A}\cdot(\boldsymbol{a}-\frac{1}{k}\boldsymbol{d}_{\text{con}}+k\boldsymbol{d}_\infty))_{k\geq 1}$ diverges in the corresponding component. Moreover, if $\boldsymbol{A}\cdot\boldsymbol{d}_\infty=\boldsymbol{0}$, then $\boldsymbol{A}\cdot(\boldsymbol{a}-\frac{1}{k}\boldsymbol{d}_{\text{con}}+k\boldsymbol{d}_\infty)=\boldsymbol{A}\cdot(\boldsymbol{a}-\frac{1}{k}\boldsymbol{d}_{\text{con}})$ for all $k\geq 1$ and therefore $(\boldsymbol{A}\cdot\boldsymbol{a}_k)_{k\geq 1}$ converges to $\boldsymbol{A}\cdot\boldsymbol{a}$, which means that we must have $\boldsymbol{A}\cdot\boldsymbol{a}=\boldsymbol{v}$. Finally, the map $\boldsymbol{B}$ is simultaneously unbounded on the sequence $(\boldsymbol{a}-\frac{1}{k}\boldsymbol{d}_{\text{con}}+k\boldsymbol{d}_\infty)_{k\geq 1}$ if and only if $\boldsymbol{B}\cdot\boldsymbol{d}_\infty\gg\boldsymbol{0}$. The following lemma yields vectors $\boldsymbol{a}$ and $\boldsymbol{d}_\infty$ that satisfy these conditions.

Lemma 5.4.4. *Let $\boldsymbol{A}\colon\mathbb{R}^d\to\mathbb{R}^m$ and $\boldsymbol{B}\colon\mathbb{R}^d\to\mathbb{R}^n$ be linear maps and $\boldsymbol{v}\in\mathbb{R}^m$. If $(\boldsymbol{a}_k)_{k\geq 1}$ is a sequence in $\mathbb{R}^d$ such that $(\boldsymbol{A}\cdot\boldsymbol{a}_k)_{k\geq 1}$ converges to $\boldsymbol{v}$ and $\boldsymbol{B}$ is simultaneously unbounded on $(\boldsymbol{a}_k)_{k\geq 1}$, then there exist $\boldsymbol{a}\in\mathbb{R}^d$ with $\boldsymbol{A}\cdot\boldsymbol{a}=\boldsymbol{v}$ and $\boldsymbol{d}_\infty\in\mathbb{R}^d$ with $\boldsymbol{A}\cdot\boldsymbol{d}_\infty=\boldsymbol{0}$ and $\boldsymbol{B}\cdot\boldsymbol{d}_\infty\gg\boldsymbol{0}$.*

Proof. To ease notation, we write $\boldsymbol{w}[i]$ for the i-th entry of a vector $\boldsymbol{w}$. For any $j\in[0,n]$ let

$$T_j := \{\boldsymbol{u}\in\mathbb{R}^d \mid (\boldsymbol{A}\cdot\boldsymbol{u})[i]=\boldsymbol{v}[i] \text{ for all } 1\leq i\leq j\}.$$

Note that T_j is an *affine subspace*, meaning that if $\boldsymbol{u}_1,\boldsymbol{u}_2,\boldsymbol{u}_3\in T_j$ and $r\in\mathbb{R}$, then $\boldsymbol{u}_1+r(\boldsymbol{u}_2-\boldsymbol{u}_3)\in T_j$.

We call a sequence $(\boldsymbol{a}_k)_{k\geq 1}$ in $\mathbb{R}^d$ *good* if $(\boldsymbol{A}\cdot\boldsymbol{a}_k)_{k\geq 1}$ converges to $\boldsymbol{v}$ and $\boldsymbol{B}$ is simultaneously unbounded on $(\boldsymbol{a}_k)_{k\geq 1}$. We shall prove that for every $j\in[0,n-1]$ if there is a good sequence contained in T_j, then there is a good sequence contained in T_{j+1}. This implies that there is a good sequence contained in T_n since by assumption, there is a good sequence in T_0. Clearly, any vector of a good sequence in T_n can be chosen as $\boldsymbol{a}$. Moreover, any such sequence contains two vectors whose difference vector can be chosen as $\boldsymbol{d}_\infty$.

Thus, let $(\boldsymbol{a}_k)_{k\geq 1}$ be a good sequence in T_j for $j\in[0,n-1]$. Let us assume that the sequence $((\boldsymbol{A}\cdot\boldsymbol{a}_k)[j+1])_{k\geq 1}$ converges to $\boldsymbol{v}[j+1]$ from below and is strictly increasing: All other cases are symmetric or already yield a subsequence in T_{j+1}. To construct a good sequence in T_{j+1}, it suffices to show that for all $N\in\mathbb{N}$ and $\varepsilon>0$ there exists a vector $\boldsymbol{a}'\in T_{j+1}$ such that $\boldsymbol{B}\cdot\boldsymbol{a}'\geq(N,\ldots,N)$ and $\|\boldsymbol{A}\cdot\boldsymbol{a}'-\boldsymbol{v}\|<\varepsilon$. Since the sequence $(\boldsymbol{a}_k)_{k\geq 1}$ is good, we can find indices $1\leq k<\ell$ such that

(i) $\boldsymbol{B}\cdot\boldsymbol{a}_k\geq(N,\ldots,N)$,

(ii) $\boldsymbol{B}\cdot\boldsymbol{a}_k\ll\boldsymbol{B}\cdot\boldsymbol{a}_\ell$,

(iii) $\|\boldsymbol{A} \cdot \boldsymbol{a}_k - \boldsymbol{v}\|, \|\boldsymbol{A} \cdot \boldsymbol{a}_\ell - \boldsymbol{v}\| < \delta$, and

(iv) $(\boldsymbol{v} - \boldsymbol{A} \cdot \boldsymbol{a}_\ell)[j+1] < \frac{1}{2}((\boldsymbol{v} - \boldsymbol{A} \cdot \boldsymbol{a}_k)[j+1])$

where $\delta > 0$ will be chosen later. By property (iv), there must be some $r \in \mathbb{R}$ with $1 < r < 2$ such that

$$(\boldsymbol{A} \cdot \boldsymbol{a}_k + r(\boldsymbol{A} \cdot \boldsymbol{a}_\ell - \boldsymbol{A} \cdot \boldsymbol{a}_k))[j+1] = \boldsymbol{v}[j+1].$$

We now set $\boldsymbol{a}' := \boldsymbol{a}_k + r(\boldsymbol{a}_\ell - \boldsymbol{a}_k)$. By the previous equation and the fact that T_j is an affine subspace, we have $\boldsymbol{a}' \in T_{j+1}$. Moreover, (i) and (ii) imply that $\boldsymbol{B} \cdot \boldsymbol{a}' \geq (N, \ldots, N)$. Furthermore, using (iii) we have

$$\begin{aligned}\|\boldsymbol{A} \cdot \boldsymbol{a}' - \boldsymbol{v}\| &= \|\boldsymbol{A} \cdot \boldsymbol{a}_k + r(\boldsymbol{A} \cdot \boldsymbol{a}_\ell - \boldsymbol{A} \cdot \boldsymbol{a}_k) - \boldsymbol{v}\| \\ &\leq \|\boldsymbol{A} \cdot \boldsymbol{a}_k - \boldsymbol{v}\| + r\|\boldsymbol{A} \cdot \boldsymbol{a}_\ell - \boldsymbol{A} \cdot \boldsymbol{a}_k\| < \delta + 2r\delta < 5\delta.\end{aligned}$$

Thus, picking $\delta := \varepsilon/5$ yields the desired $\boldsymbol{a}'$. □

5.4.4 Compatibility in Terms of Inequalities

We are now ready to describe the conditions for the vectors $\boldsymbol{a}$, $\boldsymbol{d}_{\text{con}}$, and $\boldsymbol{d}_\infty$. We define matrices and vectors over $\mathbb{Q}$ to describe systems of linear (in)equalities that are needed to express compatibility with a profile $\boldsymbol{p}$.

Limit values Let $\boldsymbol{L}_{\boldsymbol{p}}$ be a matrix and $\boldsymbol{\ell}_{\boldsymbol{p}}$ be a vector such that $\boldsymbol{L}_{\boldsymbol{p}} \cdot \boldsymbol{x} = \boldsymbol{\ell}_{\boldsymbol{p}}$ if and only if

$$\begin{aligned}\boldsymbol{r}_i \cdot \boldsymbol{x} &= \rho(i) && \text{for all } i \in [1,n] \text{ with } t_\rho(i) \in \{-1, 1\}, \\ \boldsymbol{s}_i \cdot \boldsymbol{x} &= \sigma(i) && \text{for all } i \in [1,n] \text{ with } t_\sigma(i) \in \{-1, 1\}.\end{aligned}$$

Then, as discussed above, our vectors need to satisfy $\boldsymbol{L}_{\boldsymbol{p}} \cdot \boldsymbol{a} = \boldsymbol{\ell}_{\boldsymbol{p}}$ and $\boldsymbol{L}_{\boldsymbol{p}} \cdot \boldsymbol{d}_\infty = \boldsymbol{0}$.

Constant values Let $\boldsymbol{C}_{\boldsymbol{p}}$ be a matrix and $\boldsymbol{c}_{\boldsymbol{p}}$ be a vector such that $\boldsymbol{C}_{\boldsymbol{p}} \cdot \boldsymbol{x} = \boldsymbol{c}_{\boldsymbol{p}}$ if and only if

$$\begin{aligned}\boldsymbol{r}_i \cdot \boldsymbol{x} &= \rho(i) && \text{for all } i \in [1,n] \text{ with } t_\rho(i) = 0, \\ \boldsymbol{s}_i \cdot \boldsymbol{x} &= \sigma(i) && \text{for all } i \in [1,n] \text{ with } t_\sigma(i) = 0.\end{aligned}$$

Since components $i \in [1,n]$ with $t_\rho(i) = 0$ (resp. $t_\sigma(i) = 0$) are those where the sequence $(\boldsymbol{r}_i \cdot \boldsymbol{a}_k)_{k \geq 1}$ (resp. $(\boldsymbol{s}_i \cdot \boldsymbol{a}_k)_{k \geq 1}$) is constant, our vectors clearly need to satisfy $\boldsymbol{C}_{\boldsymbol{p}} \cdot \boldsymbol{d}_{\text{con}} = \boldsymbol{0}$ and $\boldsymbol{C}_{\boldsymbol{p}} \cdot \boldsymbol{d}_\infty = \boldsymbol{0}$.

Convergence Let $\boldsymbol{D}_{\boldsymbol{p}}$ be a matrix such that $\boldsymbol{D}_{\boldsymbol{p}} \cdot \boldsymbol{x} \gg \boldsymbol{0}$ if and only if

$$\begin{aligned}\boldsymbol{r}_i \cdot \boldsymbol{x} > 0 \text{ (resp. } < 0) && \text{for all } i \in [1,n] \text{ with } t_\rho(i) = 1 \text{ (resp. } = -1), \\ \boldsymbol{s}_i \cdot \boldsymbol{x} > 0 \text{ (resp. } < 0) && \text{for all } i \in [1,n] \text{ with } t_\sigma(i) = 1 \text{ (resp. } = -1).\end{aligned}$$

Since the components $i \in [1,n]$ with $t_\rho(i) = 1$ (resp. $t_\rho(i) < 0$) are those where the sequence $(\boldsymbol{r}_i \cdot \boldsymbol{a}_k)_{k \geq 1}$ is strictly increasing (resp. decreasing) and converges to a real number from below (resp. from above), and similarly for $(\boldsymbol{s}_i \cdot \boldsymbol{a}_k)_{k \geq 1}$, we must have $\boldsymbol{D}_{\boldsymbol{p}} \cdot \boldsymbol{d}_{\text{con}} \gg \boldsymbol{0}$.

Unboundedness Let $\boldsymbol{U_p}$ be a matrix such that $\boldsymbol{U_p} \cdot \boldsymbol{x} \gg \boldsymbol{0}$ if and only if

$$\begin{aligned} \boldsymbol{r}_i \cdot \boldsymbol{x} > 0 \text{ (resp. } < 0) \qquad & \text{for all } i \in [1,n] \text{ with } t_\rho(i) = \omega \text{ (resp. } = -\omega), \\ \boldsymbol{s}_i \cdot \boldsymbol{x} > 0 \text{ (resp. } < 0) \qquad & \text{for all } i \in [1,n] \text{ with } t_\sigma(i) = \omega \text{ (resp. } = -\omega). \end{aligned}$$

Since the components $i \in [1,n]$ with $t_\rho(i) = \pm\omega$ (resp. $t_\sigma(i) = \pm\omega$) are those where the sequence $(\boldsymbol{r}_i \cdot \boldsymbol{a}_k)_{k\geq 1}$ (resp. $(\boldsymbol{s}_i \cdot \boldsymbol{a}_k)_{k\geq 1}$) diverges to $\pm\infty$, we must have $\boldsymbol{U_p} \cdot \boldsymbol{d}_\infty \gg \boldsymbol{0}$.

Let us now formally provide a list of necessary and sufficient conditions on $\boldsymbol{a}$, $\boldsymbol{d}_{\text{con}}$, and $\boldsymbol{d}_\infty$ for the existence of a sequence compatible with $\boldsymbol{p}$ that satisfies the equality constraints for $\boldsymbol{c}$.

Lemma 5.4.5. *Let $\boldsymbol{p}$ be a profile for $\varphi(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ and $\boldsymbol{c} \in \mathbb{R}^{|\boldsymbol{z}|}$. Then there exists a sequence compatible with $\boldsymbol{p}$ that satisfies the equality constraints for $\boldsymbol{c}$ if and only if there are vectors $\boldsymbol{a}, \boldsymbol{d}_{\text{con}}, \boldsymbol{d}_\infty \in \mathbb{R}^{|\boldsymbol{x}|}$ with $\boldsymbol{d}_{\text{con}} \neq \boldsymbol{0}$ such that*

(i) $\boldsymbol{L_p} \cdot \boldsymbol{a} = \boldsymbol{\ell_p}$ *and* $\boldsymbol{C_p} \cdot \boldsymbol{a} = \boldsymbol{c_p}$,

(ii) $\boldsymbol{D_p} \cdot \boldsymbol{d}_{\text{con}} \gg \boldsymbol{0}$ *and* $\boldsymbol{C_p} \cdot \boldsymbol{d}_{\text{con}} = \boldsymbol{0}$,

(iii) $\boldsymbol{L_p} \cdot \boldsymbol{d}_\infty = \boldsymbol{0}$, $\boldsymbol{C_p} \cdot \boldsymbol{d}_\infty = \boldsymbol{0}$, *and* $\boldsymbol{U_p} \cdot \boldsymbol{d}_\infty \gg \boldsymbol{0}$, *and*

(iv) $\boldsymbol{u}_j \cdot \boldsymbol{d}_{\text{con}} = \boldsymbol{u}_j \cdot \boldsymbol{d}_\infty = 0$, $\boldsymbol{v}_j \cdot \boldsymbol{d}_{\text{con}} = \boldsymbol{v}_j \cdot \boldsymbol{d}_\infty = 0$, *and* $(\boldsymbol{u}_j - \boldsymbol{v}_j) \cdot \boldsymbol{a} = \boldsymbol{w}_j \cdot \boldsymbol{c} + d_j$ *for all* $j \in [1,m]$.

Proof. We start with the "only if" direction. Let $(\boldsymbol{a}_k)_{k\geq 1}$ be a sequence compatible with $\boldsymbol{p}$ that satisfies the equality constraints for $\boldsymbol{c}$. First observe that the equality constraints imply that $\boldsymbol{u}_j \cdot \boldsymbol{a}_k = \boldsymbol{u}_j \cdot \boldsymbol{a}_\ell$, $\boldsymbol{v}_j \cdot \boldsymbol{a}_k = \boldsymbol{v}_j \cdot \boldsymbol{a}_\ell$, and $(\boldsymbol{u}_j - \boldsymbol{v}_j) \cdot \boldsymbol{a}_k = \boldsymbol{w}_j \cdot \boldsymbol{c} + d_j$ for all $2 \leq k < \ell$. Thus, by removing the first vector of the sequence, we can assume that $(\boldsymbol{a}_k)_{k\geq 1}$ fulfills this property already for $1 \leq k < \ell$. For $\boldsymbol{d}_{\text{con}}$ we choose $\boldsymbol{a}_2 - \boldsymbol{a}_1$, where $\boldsymbol{d}_{\text{con}} \neq \boldsymbol{0}$ since we assume that the vectors of compatible sequences are pairwise distinct. This fulfills (ii) since the sequences $\boldsymbol{\rho}_i$ and $\boldsymbol{\sigma}_i$ are strictly increasing/decreasing if $t_\rho, t_\sigma \in \{-1,1\}$ and constant if $t_\rho = t_\sigma = 0$. Moreover, $\boldsymbol{d}_{\text{con}}$ fulfills (iv) since $\boldsymbol{u}_j \cdot \boldsymbol{d}_{\text{con}} = \boldsymbol{u}_j \cdot \boldsymbol{a}_2 - \boldsymbol{u}_j \cdot \boldsymbol{a}_1 = 0$ and $\boldsymbol{v}_j \cdot \boldsymbol{d}_{\text{con}} = \boldsymbol{v}_j \cdot \boldsymbol{a}_2 - \boldsymbol{v}_j \cdot \boldsymbol{a}_1 = 0$ for all $j \in [1,m]$. Let $\boldsymbol{A}$ be the matrix obtained by concatenating $\boldsymbol{L_p}$ and $\boldsymbol{C_p}$ vertically and adding the rows $\boldsymbol{u}_j$, $\boldsymbol{v}_j$, and $\boldsymbol{u}_j - \boldsymbol{v}_j$ for all $j \in [1,m]$. In parallel, we define the vector $\boldsymbol{v}$ as the vertical concatenation of $\boldsymbol{\ell_p}$ and $\boldsymbol{c_p}$ extended with the entry $\boldsymbol{u}_j \cdot \boldsymbol{a}_1$ in the row of $\boldsymbol{u}_j$, the entry $\boldsymbol{v}_j \cdot \boldsymbol{a}_1$ in the row of $\boldsymbol{v}_j$, and the entry $\boldsymbol{w}_j \cdot \boldsymbol{c} + d_j$ in the row of $\boldsymbol{u}_j - \boldsymbol{v}_j$. Then we have that the sequence $(\boldsymbol{A} \cdot \boldsymbol{a}_k)_{k\geq 1}$ converges to $\boldsymbol{v}$. We can now apply Lemma 5.4.4 for $\boldsymbol{A}$, $\boldsymbol{v}$, and $\boldsymbol{B} := \boldsymbol{U_p}$ to obtain vectors $\boldsymbol{a}$ and $\boldsymbol{d}_\infty$ with the desired properties.

For the "if" direction let $\boldsymbol{a}$, $\boldsymbol{d}_{\text{con}}$, and $\boldsymbol{d}_\infty$ be as in the statement of the lemma. We claim that the sequence $(\boldsymbol{a}_k)_{k\geq k_0}$ with $\boldsymbol{a}_k := \boldsymbol{a} - \frac{1}{k}\boldsymbol{d}_{\text{con}} + k\boldsymbol{d}_\infty$ for sufficiently large $k_0 \in \mathbb{N}$ is as desired.

We first show that the sequence is compatible with $\boldsymbol{p}$. Since $\boldsymbol{d}_{\text{con}} \neq \boldsymbol{0}$, the $\boldsymbol{a}_k$ are pairwise distinct for all $k \geq k_0$ and sufficiently large k_0. If $t_\rho(i) = 1$ (resp. $t_\rho(i) = -1$),

then $\boldsymbol{r}_i \cdot \boldsymbol{a} = \rho(i)$, $\boldsymbol{r}_i \cdot \boldsymbol{d}_{\text{con}} > 0$ (resp. < 0), and $\boldsymbol{r}_i \cdot \boldsymbol{d}_\infty = 0$, which implies that the sequence $\boldsymbol{\rho}_i$ is strictly increasing (resp. decreasing) and converges to $\rho(i)$ from below (resp. above). If $t_\rho(i) = 0$, then $\boldsymbol{\rho}_i$ is constantly $\rho(i)$ since $\boldsymbol{r}_i \cdot \boldsymbol{a} = \rho(i)$, $\boldsymbol{r}_i \cdot \boldsymbol{d}_{\text{con}} = 0$, and $\boldsymbol{r}_i \cdot \boldsymbol{d}_\infty = 0$. Finally, if $t_\rho(i) = \omega$ (resp. $t_\rho(i) = -\omega$), then $\boldsymbol{r}_i \cdot \boldsymbol{d}_\infty > 0$ (resp. < 0), which means that for sufficiently large k_0, the sequence $\boldsymbol{\rho}_i$ starting at index k_0 is strictly increasing (resp. decreasing) and diverges to ∞ (resp. $-\infty$). The statement can be shown analogously for $\boldsymbol{\sigma}_i$.

It remains to prove that the sequence satisfies the equality constraints for $\boldsymbol{c}$. By (iv), we have that $\boldsymbol{u}_j \cdot (\boldsymbol{a} - \frac{1}{k}\boldsymbol{d}_{\text{con}} + k\boldsymbol{d}_\infty) = \boldsymbol{v}_j \cdot (\boldsymbol{a} - \frac{1}{\ell}\boldsymbol{d}_{\text{con}} + \ell\boldsymbol{d}_\infty) + \boldsymbol{w}_j \cdot \boldsymbol{c} + d_j$ for $k < \ell$ if and only if $\boldsymbol{u}_j \cdot \boldsymbol{a} = \boldsymbol{v}_j \cdot \boldsymbol{a} + \boldsymbol{w}_j \cdot \boldsymbol{c} + d_j$ which holds if and only if $(\boldsymbol{u}_j - \boldsymbol{v}_j) \cdot \boldsymbol{a} = \boldsymbol{w}_j \cdot \boldsymbol{c} + d_j$ which is fulfilled by (iv). □

5.4.5 Constructing the Formula

We now prove Theorem 5.4.1. To this end, let $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ be an arbitrary existential LRA formula. If φ is a conjunction of (in)equalities, Lemma 5.4.5 essentially tells us how to construct an existential formula for $\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. Moreover, by Theorem 5.2.1, we may assume φ to be quantifier-free. Thus, it remains to treat the case where φ is a Boolean combination of constraints as in Equation (5.3).

We first move all negations inward and, if necessary, negate atoms so that we are left with a positive Boolean combination of strict inequality and equality atoms. Let $\alpha_i := \boldsymbol{r}_i \cdot \boldsymbol{x} < \boldsymbol{s}_i \cdot \boldsymbol{y} + \boldsymbol{t}_i \cdot \boldsymbol{z} + h_i$ for $i \in [1, n]$ be the inequality atoms and $\beta_j := \boldsymbol{u}_j \cdot \boldsymbol{x} = \boldsymbol{v}_j \cdot \boldsymbol{y} + \boldsymbol{w}_j \cdot \boldsymbol{z} + d_j$ for $j \in [1, m]$ be the equality atoms in φ.

As in the Presburger case, we guess a subset of the atoms and assert that (i) satisfying all these atoms makes φ true and (ii) there exists a clique satisfying the conjunction of these atoms.

Let φ' be the formula obtained from φ by replacing each α_i with $q_i^< = 1$, for a fresh variable $q_i^<$, for all $i \in [1, n]$ and each β_j with $q_j^= = 1$, for a fresh variable $q_j^=$, for all $j \in [1, m]$ and adding the restrictions $q_i^< = 0 \vee q_i^< = 1$ and $q_j^= = 0 \vee q_j^= = 1$. Now, φ is equivalent to

$$\psi := \exists \boldsymbol{q}^<, \boldsymbol{q}^= \colon \varphi' \wedge \bigwedge_{i=1}^{n} (q_i^< = 1 \to \alpha_i) \wedge \bigwedge_{j=1}^{m} (q_j^= = 1 \to \beta_j).$$

We represent a profile $\boldsymbol{p}$ by the variables ρ_i, σ_i, $t_{\rho,i}$, and $t_{\sigma,i}$ for all $i \in [1, n]$ where ρ_i, σ_i range over $\mathbb{R}$ and $t_{\rho,i}, t_{\sigma,i}$ range over $\{-2, -1, 0, 1, 2\}$. Here, -2 and 2 represent $-\omega$ and ω, respectively.

We now define formulas for the inequalities and equality constraints from Lemma 5.4.5. For $i \in [1, n]$ let $\rho_i, \sigma_i, t_{\rho,i}, t_{\sigma,i}, \boldsymbol{x}, \boldsymbol{x}_{\text{con}}, \boldsymbol{x}_\infty$ be fresh variables. Our first formula λ_i contains all the constraints from $\boldsymbol{L}_p \cdot \boldsymbol{x} = \boldsymbol{\ell}_{\boldsymbol{p}}$ and $\boldsymbol{L}_{\boldsymbol{p}} \cdot \boldsymbol{x}_\infty = \boldsymbol{0}$ that stem from the atom α_i:

$$\begin{aligned} \lambda_i := {} & ((t_{\rho,i} = -1 \vee t_{\rho,i} = 1) \to \boldsymbol{r}_i \cdot \boldsymbol{x} = \rho_i \wedge \boldsymbol{r}_i \cdot \boldsymbol{x}_\infty = 0) \wedge \\ & ((t_{\sigma,i} = -1 \vee t_{\sigma,i} = 1) \to \boldsymbol{s}_i \cdot \boldsymbol{x} = \sigma_i \wedge \boldsymbol{s}_i \cdot \boldsymbol{x}_\infty = 0) \end{aligned}$$

Next, the formula χ_i states the constraints about constant values, meaning those from $\boldsymbol{C_p} \cdot \boldsymbol{x} = \boldsymbol{c_p}$ and $\boldsymbol{C_p} \cdot \boldsymbol{x}_{\text{con}} = \boldsymbol{C_p} \cdot \boldsymbol{x}_\infty = \boldsymbol{0}$, that stem from α_i:

$$\begin{aligned}\chi_i := (t_{\rho,i} = 0 \to \boldsymbol{r}_i \cdot \boldsymbol{x} = \rho_i \wedge \boldsymbol{r}_i \cdot \boldsymbol{x}_{\text{con}} = 0 \wedge \boldsymbol{r}_i \cdot \boldsymbol{x}_\infty = 0) \wedge \\ (t_{\sigma,i} = 0 \to \boldsymbol{s}_i \cdot \boldsymbol{x} = \sigma_i \wedge \boldsymbol{s}_i \cdot \boldsymbol{x}_{\text{con}} = 0 \wedge \boldsymbol{s}_i \cdot \boldsymbol{x}_\infty = 0)\end{aligned}$$

With δ_i we express the convergence constraints from $\boldsymbol{D_p} \cdot \boldsymbol{x}_{\text{con}} \gg \boldsymbol{0}$ required by α_i:

$$\begin{aligned}\delta_i := (t_{\rho,i} = -1 \to \boldsymbol{r}_i \cdot \boldsymbol{x}_{\text{con}} < 0) \wedge (t_{\rho,i} = 1 \to \boldsymbol{r}_i \cdot \boldsymbol{x}_{\text{con}} > 0) \wedge \\ (t_{\sigma,i} = -1 \to \boldsymbol{s}_i \cdot \boldsymbol{x}_{\text{con}} < 0) \wedge (t_{\sigma,i} = 1 \to \boldsymbol{s}_i \cdot \boldsymbol{x}_{\text{con}} > 0)\end{aligned}$$

Furthermore, μ_i states the unboundedness condition $\boldsymbol{U}_p \cdot \boldsymbol{x}_\infty \gg \boldsymbol{0}$:

$$\begin{aligned}\mu_i := (t_{\rho,i} = -2 \to \boldsymbol{r}_i \cdot \boldsymbol{x}_\infty < 0) \wedge (t_{\rho,i} = 2 \to \boldsymbol{r}_i \cdot \boldsymbol{x}_\infty > 0) \wedge \\ (t_{\sigma,i} = -2 \to \boldsymbol{s}_i \cdot \boldsymbol{x}_\infty < 0) \wedge (t_{\sigma,i} = 2 \to \boldsymbol{s}_i \cdot \boldsymbol{x}_\infty > 0)\end{aligned}$$

Finally, ε_j expresses the equality constraints (iv) in Lemma 5.4.5: For all $j \in [1, m]$ let

$$\begin{aligned}\varepsilon_j := \boldsymbol{u}_j \cdot \boldsymbol{x}_{\text{con}} = 0 \wedge \boldsymbol{u}_j \cdot \boldsymbol{x}_\infty = 0 \wedge \boldsymbol{v}_j \cdot \boldsymbol{x}_{\text{con}} = 0 \wedge \boldsymbol{v}_j \cdot \boldsymbol{x}_\infty = 0 \wedge \\ (\boldsymbol{u}_j - \boldsymbol{v}_j) \cdot \boldsymbol{x} = \boldsymbol{w}_j \cdot \boldsymbol{z} + d_j.\end{aligned}$$

To check if $\boldsymbol{p}$ is a $\boldsymbol{z}$-admissible profile, we define the formula

$$\begin{aligned}\theta := \bigwedge_{i=1}^{n} t_{\rho,i} \in \{-2,-1,0,1,2\} \wedge t_{\sigma,i} \in \{-1,0,1,2\} \wedge (t_{\rho,i} = 2 \to t_{\sigma,i} = 2) \wedge \\ \big((t_{\rho,i} \in \{-1,0\} \wedge t_{\sigma,i} \in \{0,1\} \vee t_{\rho,i} = -1 \wedge t_{\sigma,i} = -1) \to \rho_i < \sigma_i + \boldsymbol{t}_i \cdot \boldsymbol{z} + h_i\big) \wedge \\ \big((t_{\rho,i} = 0 \wedge t_{\sigma,i} = -1 \vee t_{\rho,i} = 1) \to \rho_i \leq \sigma_i + \boldsymbol{t}_i \cdot \boldsymbol{z} + h_i\big)\end{aligned}$$

where we use set notation as a shorthand. Then we claim that $\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ is equivalent to

$$\begin{aligned}\gamma := \exists \boldsymbol{q}^{<}, \boldsymbol{q}^{=}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{x}_{\text{con}}, \boldsymbol{x}_\infty \colon \varphi' \wedge \theta \wedge \boldsymbol{x}_{\text{con}} \neq \boldsymbol{0} \wedge \\ \bigwedge_{i=1}^{n} (q_i^{<} = 1 \to \lambda_i \wedge \chi_i \wedge \delta_i \wedge \mu_i) \wedge \bigwedge_{j=1}^{m} (q_j^{=} = 1 \to \varepsilon_j).\end{aligned}$$

We show that for any $\boldsymbol{c} \in \mathbb{R}^{|\boldsymbol{z}|}$ we have $\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c})$ if and only if $\gamma(\boldsymbol{c})$. For a valuation ν of the $q_i^{<}, q_j^{=}$ to $\{0,1\}$ let $I_\nu := \{i \in [1,n] \mid \nu(q_i^{<}) = 1\}$ and $J_\nu := \{j \in [1,m] \mid \nu(q_j^{=}) = 1\}$. By Ramsey's theorem, $\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c})$ holds if and only if there is a valuation ν of the $q_i^{<}, q_j^{=}$ satisfying φ' such that

$$\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \bigwedge_{i \in I_\nu} \alpha_i(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c}) \wedge \bigwedge_{j \in J_\nu} \beta_j(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c}).$$

By Lemmas 5.4.3 and 5.4.5, this is equivalent to

$$\exists \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{x}_{\text{con}}, \boldsymbol{x}_\infty \colon \theta(\boldsymbol{p}, \boldsymbol{c}) \wedge \boldsymbol{x}_{\text{con}} \neq \boldsymbol{0} \wedge \bigwedge_{i \in I_\nu} \lambda_i \wedge \chi_i \wedge \delta_i \wedge \mu_i \wedge \bigwedge_{j \in J_\nu} \varepsilon_j(\boldsymbol{x}, \boldsymbol{x}_{\text{con}}, \boldsymbol{x}_\infty, \boldsymbol{c}). \quad (5.6)$$

This holds for some valuation ν of the $q_i^{<}, q_j^{=}$ satisfying φ' if and only if $\gamma(\boldsymbol{c})$.

Note that γ is an existential formula in LRA of linear size. Moreover, all constructions can be performed in polynomial time.

5.4.6 Linear Rational Arithmetic

In this section we observe that if we consider rational numbers instead of real numbers, then the analogue of Theorem 5.4.1 still holds. We define *linear rational arithmetic* as the structure $\langle \mathbb{Q}; <, +, 0, 1 \rangle$. It follows from [FR75] that linear real arithmetic and linear rational arithmetic are *elementarily equivalent*, i.e. their theories coincide. In other words, a first-order sentence is true in linear real arithmetic if and only if it is true in linear rational arithmetic. In particular, this implies that since all rational numbers are definable in linear real/rational arithmetic, a formula $\varphi(\boldsymbol{x})$ is satisfied by $\boldsymbol{c} \in \mathbb{Q}^{|\boldsymbol{x}|}$ in linear real arithmetic if and only if it is satisfied by $\boldsymbol{c}$ in linear rational arithmetic. Note that a priori this does not necessarily imply that over the rationals the Ramsey quantifier can be eliminated with the same procedure as in the real case. However, an inspection of the proof of Theorem 5.4.1 shows that this is indeed the case.

Theorem 5.4.6. *Given an existential formula $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ in linear rational arithmetic with $|\boldsymbol{x}| = |\boldsymbol{y}|$, one can construct in polynomial time an existential formula in linear rational arithmetic of linear size that is equivalent to $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$.*

Proof. We can view $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ as a formula in LRA and thus obtain a formula $\psi(\boldsymbol{z})$ that is equivalent to $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ in LRA using Theorem 5.4.1. We claim that $\psi(\boldsymbol{z})$ is equivalent to $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ in linear rational arithmetic as well.

Assume $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c})$ holds in linear rational arithmetic for some vector $\boldsymbol{c} \in \mathbb{Q}^{|\boldsymbol{z}|}$. Then in particular, $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c})$ holds in LRA and thus by construction, $\psi(\boldsymbol{c})$ holds in LRA. Since LRA and linear rational arithmetic are elementarily equivalent, this implies that $\psi(\boldsymbol{c})$ holds in linear rational arithmetic.

Conversely, assume that $\psi(\boldsymbol{c})$ holds for some $\boldsymbol{c} \in \mathbb{Q}^{|\boldsymbol{z}|}$ in linear rational arithmetic. Recall from the proof of Theorem 5.4.1 that ψ is equivalent to a disjunction $\bigvee_{i=1}^{n} \varphi_i$ where each φ_i is a conjunction as in Equation (5.6). To each φ_i we associate the system S_i of linear (in)equalities described in Lemma 5.4.5. Since $\psi(\boldsymbol{c})$ holds in linear rational arithmetic, some system S_i has a rational solution where also the valuations of the variables in $\boldsymbol{p}$ are rational. Moreover, the valuation of $\boldsymbol{p}$ is a $\boldsymbol{c}$-admissible profile. Now, an inspection of the "if" direction in the proof of Lemma 5.4.5 yields that the constructed compatible sequence based on rational solutions of S_i and $\boldsymbol{p}$ will be rational as well. Thus, since the "if" direction in Lemma 5.4.3 only picks an infinite subsequence, there exists a rational infinite clique that witnesses that $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c})$ holds in linear rational arithmetic. □

5.5 Eliminating Ramsey Quantifiers in Linear Integer Real Arithmetic

We are now ready to prove our main theorem of this chapter. We show that the Ramsey quantifier can be eliminated in LIRA with the same complexity as in the cases where we either only deal with integers or reals.

Theorem 5.5.1. *Given an existential formula $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ in LIRA, where $\boldsymbol{x}$ and $\boldsymbol{y}$ have the same type, one can construct in polynomial time an existential formula in LIRA of linear size that is equivalent to $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$.*

Proof. Recall that a decomposition of an existential LIRA formula φ is an existentially quantified Boolean combination of LIA and LRA formulas that is equivalent to the separation of φ, i.e. where each variable is split into its integer and real part. It suffices to prove the theorem for a decomposition of φ: Given φ, we first compute a decomposition $\varphi'(\boldsymbol{x}^{\mathrm{i/r}}, \boldsymbol{y}^{\mathrm{i/r}}, \boldsymbol{z}^{\mathrm{i/r}})$ using Lemma 2.4.5. We then show how to compute a formula $\psi'(\boldsymbol{z}^{\mathrm{i/r}})$ in LIRA that is equivalent to $\exists^{\mathrm{ram}} \boldsymbol{x}^{\mathrm{i/r}}, \boldsymbol{y}^{\mathrm{i/r}} \colon \varphi'(\boldsymbol{x}^{\mathrm{i/r}}, \boldsymbol{y}^{\mathrm{i/r}}, \boldsymbol{z}^{\mathrm{i/r}})$. Let $\psi(\boldsymbol{z})$ be the formula obtained from ψ' by replacing every z_i^{int} with $\lfloor z_i \rfloor$ and every z_i^{real} with $z_i - \lfloor z_i \rfloor$. Then ψ is equivalent to $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ since $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c})$ if and only if $\boldsymbol{c} = \boldsymbol{c}^{\mathrm{int}} + \boldsymbol{c}^{\mathrm{real}}$ and $\exists^{\mathrm{ram}} \boldsymbol{x}^{\mathrm{i/r}}, \boldsymbol{y}^{\mathrm{i/r}} \colon \varphi'(\boldsymbol{x}^{\mathrm{i/r}}, \boldsymbol{y}^{\mathrm{i/r}}, \boldsymbol{c}^{\mathrm{i/r}})$.

Thus, we now assume that $\varphi'(\boldsymbol{x}^{\mathrm{i/r}}, \boldsymbol{y}^{\mathrm{i/r}}, \boldsymbol{z}^{\mathrm{i/r}})$ is a decomposition of φ. By Theorem 5.2.1, we can assume that φ' is quantifier-free. We further assume that all negations are moved directly into the atoms. Let $\alpha_1, \dots, \alpha_n$ be the Presburger atoms and $\beta_1, \dots, \beta_m$ be the atoms in LRA of φ'. For fresh integer variables $p_1, \dots, p_n$ and real variables $q_1, \dots, q_m$ let σ be the formula obtained from φ' by replacing every α_i with $p_i = 1$ and every β_j with $q_j = 1$ and adding the constraints $p_i = 0 \vee p_i = 1$ and $q_j = 0 \vee q_j = 1$. Then φ' is equivalent to

$$\delta := \exists \boldsymbol{p}, \boldsymbol{q} \colon \sigma \wedge \bigwedge_{i=1}^{n} (p_i = 1 \to \alpha_i) \wedge \bigwedge_{j=1}^{m} (q_j = 1 \to \beta_j).$$

Since each p_i and q_j has only two possible valuations, Ramsey's theorem implies that $\exists^{\mathrm{ram}} \boldsymbol{x}^{\mathrm{i/r}}, \boldsymbol{y}^{\mathrm{i/r}} \colon \delta(\boldsymbol{x}^{\mathrm{i/r}}, \boldsymbol{y}^{\mathrm{i/r}}, \boldsymbol{z}^{\mathrm{i/r}})$ is equivalent to

$$\exists \boldsymbol{p}, \boldsymbol{q} \colon \sigma \wedge \exists^{\mathrm{ram}} \boldsymbol{x}^{\mathrm{i/r}}, \boldsymbol{y}^{\mathrm{i/r}} \colon \bigwedge_{i=1}^{n} (p_i = 1 \to \alpha_i) \wedge \bigwedge_{j=1}^{m} (q_j = 1 \to \beta_j).$$

Let $\alpha := \bigwedge_{i=1}^{n} (p_i = 1 \to \alpha_i)$ and $\beta := \bigwedge_{j=1}^{m} (q_j = 1 \to \beta_j)$. To split the vectors of the Ramsey quantified variables into integer and real components, we have to allow that in the infinite clique either the integer components or the real components do not change throughout the clique. To this end, we introduce a fresh variable r that is either 0 or 1 and get the equivalent formula

$$\begin{aligned}
&\exists \boldsymbol{p}, \boldsymbol{q}, r \colon \sigma \wedge (r = 0 \vee r = 1) \wedge \\
&\Big(\big(\exists^{\mathrm{ram}} \boldsymbol{x}^{\mathrm{int}}, \boldsymbol{y}^{\mathrm{int}} \colon \alpha(\boldsymbol{p}, \boldsymbol{x}^{\mathrm{int}}, \boldsymbol{y}^{\mathrm{int}}, \boldsymbol{z}^{\mathrm{int}}) \big) \vee r = 0 \wedge \exists \boldsymbol{x}^{\mathrm{int}} \colon \alpha(\boldsymbol{p}, \boldsymbol{x}^{\mathrm{int}}, \boldsymbol{x}^{\mathrm{int}}, \boldsymbol{z}^{\mathrm{int}}) \Big) \wedge \\
&\Big(\big(\exists^{\mathrm{ram}} \boldsymbol{x}^{\mathrm{real}}, \boldsymbol{y}^{\mathrm{real}} \colon \beta(\boldsymbol{q}, \boldsymbol{x}^{\mathrm{real}}, \boldsymbol{y}^{\mathrm{real}}, \boldsymbol{z}^{\mathrm{real}}) \big) \vee r = 1 \wedge \exists \boldsymbol{x}^{\mathrm{real}} \colon \beta(\boldsymbol{q}, \boldsymbol{x}^{\mathrm{real}}, \boldsymbol{x}^{\mathrm{real}}, \boldsymbol{z}^{\mathrm{real}}) \Big)
\end{aligned}$$

where the Ramsey quantifiers can be eliminated using Theorems 5.3.1 and 5.4.1. □

Let us mention a simple consequence of Theorem 5.5.1.

Corollary 5.5.2. *The infinite clique problem for existential formulas in LIRA is* NP*-complete.*

Proof. The NP lower bound already holds for existential formulas in Presburger arithmetic and LRA by a reduction from the respective satisfiability problem, which by Propositions 2.4.1 and 2.4.3 is NP-complete. Indeed, let $\varphi(\boldsymbol{x})$ be an existential formula in Presburger arithmetic or LRA. Then φ is satisfiable if and only if $\exists^{\mathrm{ram}}(\boldsymbol{x}, u), (\boldsymbol{y}, v) \colon \varphi(\boldsymbol{x})$.

For the upper bound let $\varphi(\boldsymbol{x}, \boldsymbol{y})$ be an existential formula in LIRA, where $\boldsymbol{x}$ and $\boldsymbol{y}$ have the same type. By Theorem 5.5.1, one can compute an existential formula ψ in LIRA that is equivalent to $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y})$ in polynomial time. Then satisfiability can be checked in NP using Proposition 2.4.6. □

To conclude this section, we observe that we can use the same procedure as in Theorem 5.5.1 to eliminate the Ramsey quantifier in *linear integer rational arithmetic*, i.e. over the structure $\langle \mathbb{Q}; <, \lfloor \cdot \rfloor, +, 0, 1 \rangle$. The only difference is that we use Theorem 5.4.6 in place of Theorem 5.4.1.

Theorem 5.5.3. *Given an existential formula $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ in linear integer rational arithmetic, where $\boldsymbol{x}$ and $\boldsymbol{y}$ have the same type, one can construct in polynomial time an existential formula in linear integer rational arithmetic of linear size that is equivalent to $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$.*

5.6 Applications

In this section we present concrete applications of the results obtained in this chapter. We will see that our elimination procedure of the Ramsey quantifier in linear arithmetics can be used to prove precise complexity results for various problems in the context of verification. These results significantly improve the best currently known upper bounds.

5.6.1 Monadic Decomposability

Recall that a formula is monadically decomposable if it is equivalent to a Boolean combination of monadic formulas, i.e. formulas with only a single free variable. As already motivated in the introduction, the problem of monadic decomposability has applications in a wide range of areas. Essentially, it can help whenever simplification by removing dependencies between variables leads to more efficient algorithms. This is for example the case in the context of constraint databases [GRS99; Cho+03; Lib03], string analysis [Vea+17; Hag+20], theorem proving [KV14], compiler optimization [ASU86], and quantifier elimination [Hag+20]. For the purpose of enabling the use of SMT solvers, Veanes et al. [Vea+17] considered monadic decomposability in the quantifier-free fragment of theories. Recall that a formula is monadically decomposable in the quantifier-free fragment if it can be written as a Boolean combination of quantifier-free monadic formulas.

Monadic decomposability for LIA was shown to be decidable (under slightly different terms) by Ginsburg and Spanier [GS66b]. More generally, Libkin [Lib03] provides sufficient conditions on the theory under which monadic decomposability is decidable

(see Theorem 3.3.9). Since LIA and LRA both satisfy these conditions, Theorem 3.3.9 implies that the problem is decidable for both theories. In terms of complexity, monadic decomposability in the quantifier-free fragment of LIA was shown to be coNP-complete by Hague et al. [Hag+20]. The precise complexity in the case of LRA and LIRA, however, remained open. Using Theorems 5.4.1 and 5.5.1, we can show that coNP-completeness also holds for LRA and LIRA. As discussed in Sections 2.4.2 and 2.4.4, LIA and LIRA only admit quantifier elimination if we allow modulo constraints. For this reason, in the following we assume that formulas in the quantifier-free fragments of LIA and LIRA can contain modulo constraints.

Corollary 5.6.1. *Given a quantifier-free formula in LIA, LRA, or LIRA, deciding monadic decomposability in the respective quantifier-free fragment is* coNP*-complete.*

Proof. We start with the coNP lower bounds. To show that monadic decomposability in the quantifier-free fragment of LRA is coNP-hard, we reduce from the unsatisfiability problem for quantifier-free formulas in LRA, which by Proposition 2.4.3 is coNP-complete. Let $\psi(\boldsymbol{x})$ be a quantifier-free formula in LRA. We define the formula

$$\varphi(\boldsymbol{x}, y, z) := \neg\psi(\boldsymbol{x}) \vee y = z$$

where y, z are fresh variables. We claim that ψ is unsatisfiable if and only if φ is monadically decomposable. For the "only if" direction assume that ψ is unsatisfiable. Then $\varphi(\boldsymbol{x}, y, z)$ is satisfied for all valuations of $\boldsymbol{x}, y, z$, which means that φ is trivially monadically decomposable. For the "if" direction assume that $\psi(\boldsymbol{c})$ holds for some $\boldsymbol{c} \in \mathbb{R}^{|\boldsymbol{x}|}$. Since LRA and linear rational arithmetic are elementarily equivalent, we may assume that $\boldsymbol{c} \in \mathbb{Q}^{|\boldsymbol{x}|}$. Towards a contradiction, suppose φ is monadically decomposable. Then the formula $\varepsilon(y, z) := \varphi(\boldsymbol{c}, y, z)$ must also be monadically decomposable. However, we have $\varepsilon(y, z)$ is equivalent to $y = z$, which is clearly not monadically decomposable.

The lower bound for LIA can be shown analogously. This implies that monadic decomposability is also coNP-hard for LIRA since any atom of a monadic quantifier-free LIRA formula with free integer variable can be converted to an LIA atom.

The coNP upper bounds follow from Theorems 5.3.1, 5.4.1, and 5.5.1 as follows. Let $\varphi(\boldsymbol{x})$ with $k := |\boldsymbol{x}|$ be a quantifier-free formula in LIA, LRA, or LIRA. Let A be either the set of integers or reals depending on whether φ is a formula in LIA, LRA, or LIRA. Recall that $\approx_j^{[\![\varphi]\!]}$ is the equivalence relation on A defined by

$$a \approx_j^{[\![\varphi]\!]} b \quad :\iff \quad \forall \boldsymbol{c} \in A^{k-1} \colon (a \odot_j \boldsymbol{c} \in [\![\varphi]\!] \iff b \odot_j \boldsymbol{c} \in [\![\varphi]\!])$$

for all $j \in [1, k]$. That is, a and b satisfy φ for the same vectors $\boldsymbol{c}$ when placed at the j-th position surrounded by the components of $\boldsymbol{c}$. We define the formula for the complement of $\approx_j^{[\![\varphi]\!]}$ as

$$x \not\approx_j y := \exists \boldsymbol{z} \colon \varphi(x \odot_j \boldsymbol{z}) \wedge \neg\varphi(y \odot_j \boldsymbol{z}) \vee \neg\varphi(x \odot_j \boldsymbol{z}) \wedge \varphi(y \odot_j \boldsymbol{z})$$

for all $j \in [1, k]$. Now by Proposition 3.3.10, we have that φ is *not* monadically decomposable if and only if $\approx_j^{[\![\varphi]\!]}$ has infinite index for some $j \in [1, k-1]$, which is the case if

and only if $\not\approx_j^{[\![\varphi]\!]}$ has an infinite clique for some $j \in [1, k-1]$. Thus, it remains to check whether there exists $j \in [1, k-1]$ such that

$$\exists^{\mathrm{ram}} x, y \colon x \not\approx_j y$$

is satisfiable, i.e. to solve the infinite clique problem for the formula $\not\approx_j$. Since φ is quantifier-free, we observe that $x \not\approx_j y$ is an existential formula. Moreover, if φ is an LIRA formula and contains modulo constraints, we can remove the modulo constraints in favor of more existentially quantified variables as mentioned in Section 2.4.2. Thus, we can apply Corollary 5.5.2 to decide the infinite clique problem for $\not\approx_j$ in NP. This results in an NP-algorithm for the problem of monadic non-decomposability. □

Let us mention that although a coNP-algorithm for monadic decomposability in LIA was known, our new procedure is asymptotically much more efficient than the one by Hague et al. [Hag+20]: They construct for each variable x a formula ν_x that contains an exponential constant B. They choose $B = 2^{dmn+3}$ where d is the number of bits needed to encode the constants in φ, n is the number of linear inequalities in any disjunct in φ, and m is the number of variables in φ. Thus, this constant requires $dmn + 3$ bits, meaning ν_x is of length $\mathcal{O}(dmn)$, which is cubic in the size of the input formula. This means that they have to check satisfiability of formulas of cubic size. In contrast, each of our formulas that we check for satisfiability after the elimination of the Ramsey quantifier is of only linear size.

However, the algorithm by Hague et al. [Hag+20] *computes* a monadic decomposition if it exists. It is even shown that the size of the decomposition is at most exponential, which is optimal if one restricts to decompositions in DNF/CNF. Nevertheless, our approach can be used as a termination check also in case of LRA and LIRA.

5.6.2 Variadic Decomposability

In the previous section we considered the problem of checking whether a formula can be decomposed into a Boolean combination of formulas that only depend on one variable. More generally, instead of only allowing a single free variable in the subformulas of the decomposition, we can demand that all the free variables stem from the same block of some given partition of the variables. We say that formula $\varphi(x_1, \ldots, x_k)$ over a structure $\mathfrak{A}$ is P-decomposable for a partition P of $[1, k]$ if φ is equivalent to a Boolean combination of formulas each of whose free variables having indices that are contained in a single block of P. For instance, if $P = \{\{1\}, \ldots, \{k\}\}$ is the discrete partition of $[1, k]$, then P-decomposability is the same as monadic decomposability. We call the problem of deciding whether a given formula is P-decomposable for a given partition P *variadic decomposability*.

Example 5.6.2. Consider the formula

$$\varphi(x_1, x_2, x_3) := x_1 + x_2 \geq x_3 \wedge x_1 \geq x_3 - 2 \wedge x_1 \geq 0 \wedge x_2 \geq 0$$

in Presburger arithmetic. Clearly, φ is not monadically decomposable. However, for the partition $P := \{\{1,3\},\{2\}\}$ we have that φ is P-decomposable since it is equivalent to

$$x_1 \geq 0 \wedge x_2 \geq 0 \wedge \big((x_1 = x_3 - 2 \wedge x_2 \geq 2) \vee (x_1 = x_3 - 1 \wedge x_2 \geq 1) \vee x_2 \geq x_3\big)$$

where each atom either only contains variables from $\{x_1, x_3\}$ or $\{x_2\}$.

The problem of variadic decomposability was already considered by Libkin [Lib03]. In fact, Theorem 3.3.9 was shown in [Lib03] in the more general setting. Thus, variadic decomposability is decidable for LIA and LRA since both structures satisfy the conditions of Theorem 3.3.9. In terms of complexity, it was shown by Hague et al. [Hag+20] that the problem is coNP-complete for the quantifier-free fragment of LIA, thus, matching the complexity of monadic decomposability. We show that also for the quantifier-free fragments of LRA and LIRA the complexity of variadic decomposability is still coNP. For the proof we need the analogous characterization via finite-index equivalence relations as for monadic decomposability. Recall that for a formula $\varphi(x_1, \ldots, x_k)$ over a structure $\mathfrak{A}$ with domain A and $I \subseteq [1,k]$ we define the equivalence relation $\approx_I^{[\![\varphi]\!]_\mathfrak{A}}$ on $A^{|I|}$ such that

$$\boldsymbol{a} \approx_I^{[\![\varphi]\!]_\mathfrak{A}} \boldsymbol{b} \quad :\iff \quad \forall \boldsymbol{c} \in A^{k-|I|} \colon (\boldsymbol{a} \odot_I \boldsymbol{c} \in [\![\varphi]\!]_\mathfrak{A} \iff \boldsymbol{b} \odot_I \boldsymbol{c} \in [\![\varphi]\!]_\mathfrak{A}).$$

Here, $\boldsymbol{a} \odot_I \boldsymbol{c}$ denotes the unique k-tuple whose projection to the components in I is $\boldsymbol{a}$ and whose projection to $[1,k] \setminus I$ is $\boldsymbol{c}$.

Proposition 5.6.3. *Let $\mathfrak{A} = (A, I)$ be a structure such that for every finite-index equivalence relation $E \subseteq A^n \times A^n$ definable over $\mathfrak{A}$ each equivalence class contains a representative that is definable by ground terms over $\mathfrak{A}$. Then a quantifier-free formula $\varphi(x_1, \ldots, x_k)$ is P-decomposable for a partition P of $[1,k]$ in the quantifier-free fragment of $\mathfrak{A}$ if and only if $\approx_B^{[\![\varphi]\!]_\mathfrak{A}}$ has finite index for all blocks $B \in P$.*

Proof. The proof is similar to the proof of Proposition 3.3.10 by using Lemma 3.3.6. For this we observe that a partition $P = \{B_1, \ldots, B_m\}$ is generated by its blocks $B_1, \ldots, B_m$. □

Recall from Example 3.3.11 that LIA, LRA, and LIRA all satisfy the condition of Proposition 5.6.3. Hence, it can be used to prove coNP-completeness of variadic decomposability similarly as in Corollary 5.6.1.

Corollary 5.6.4. *Given a quantifier-free formula in LIA, LRA, or LIRA, deciding variadic decomposability in the respective quantifier-free fragment is coNP-complete.*

Proof. The lower bounds are inherited from monadic decomposability by choosing the discrete partition. For the upper bounds let $\varphi(x_1, \ldots, x_k)$ be a quantifier-free formula in LIA, LRA, or LIRA and P be a partition of $[1,k]$. For every block $B \in P$ we define the formula

$$\boldsymbol{x} \not\approx_B \boldsymbol{y} := \exists \boldsymbol{z} \colon \big(\varphi(\boldsymbol{x} \odot_B \boldsymbol{z}) \wedge \neg\varphi(\boldsymbol{y} \odot_B \boldsymbol{z})\big) \vee \big(\neg\varphi(\boldsymbol{x} \odot_B \boldsymbol{z}) \wedge \varphi(\boldsymbol{y} \odot_B \boldsymbol{z})\big)$$

that expresses the complement of $\approx_B^{[\![\varphi]\!]}$, where $|\boldsymbol{x}| = |\boldsymbol{y}| = |B|$ and $|\boldsymbol{z}| = k - |B|$. Note that the formulas $\boldsymbol{x} \not\approx_B \boldsymbol{y}$ are existential and can be easily computed in polynomial time. By Proposition 5.6.3, we have that φ is *not* P-decomposable if and only if $\approx_B^{[\![\varphi]\!]}$ has infinite index for some $B \in P$, which is the case if and only if $\not\approx_B^{[\![\varphi]\!]}$ has an infinite clique for some $B \in P$. Thus, it remains to check whether there exists $B \in P$ such that

$$\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \boldsymbol{x} \not\approx_B \boldsymbol{y}$$

is satisfiable, i.e. to decide the infinite clique problem for the formula $\not\approx_B$. This can be done in NP for each $B \in P$ using Corollary 5.5.2. □

5.6.3 Linear Liveness for Systems with Counters and Clocks

We have already seen in Section 4.4.1 that the Ramsey quantifier can be used to check liveness properties of transition systems, provided that the reachability relation is expressible in the respective logic (in Section 4.4.1 over an (tree-)automatic structure). We now consider similar liveness problems for systems whose reachability relation can be expressed in linear integer/real arithmetic. More specifically, the systems that we consider will involve counters and/or clocks. The reason is that configurations of such systems will be tuples of integers representing the counter values and reals representing the clock values. Then the reachability relation can in some cases be expressed as a formula over linear integer/real arithmetic.

More precisely, we consider transition systems with set of configurations $C := Q \times \mathbb{Z}^k \times D^\ell$ where Q is a finite set of control states and D is either $\mathbb{R}$ or $\mathbb{Q}$. Moreover, we are interested in transition systems with step relation $\to \subseteq C \times C$ such that for all $p, q \in Q$ one can effectively construct an existential formula $\varphi_{p,q}(\boldsymbol{x}, \boldsymbol{y})$ in LIRA for its reachability relation $\to^+$ restricted to initial state p and target state q. This means $\varphi_{p,q}(\boldsymbol{a}, \boldsymbol{b})$ if and only if $(p, \boldsymbol{a}) \to^+ (q, \boldsymbol{b})$ for all $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}^k \times D^\ell$. We will see some concrete examples of classes of such transition systems below.

Recall from Section 3.2.2 that by assuming that $Q = \{1, \ldots, |Q|\}$, we can construct a formula φ for the reachability relation that includes the state component of the configurations:

$$\varphi(x_0, \boldsymbol{x}, y_0, \boldsymbol{y}) := \bigvee_{p,q \in Q} x_0 = p \wedge y_0 = q \wedge \varphi_{p,q}(\boldsymbol{x}, \boldsymbol{y})$$

Then for all configurations $(p, \boldsymbol{a}), (q, \boldsymbol{b}) \in C$ we have that $\varphi(p, \boldsymbol{a}, q, \boldsymbol{b})$ if and only if $(p, \boldsymbol{a}) \to^+ (q, \boldsymbol{b})$. Note that the size of φ is polynomial in the sizes of the $\varphi_{p,q}$. Thus, in the following we may just combine the state and the vector of counter/clock values.

Let us recall the definition of the *linear liveness problem*:

Given A description of a transition system and existential LIRA formulas $\varepsilon(\boldsymbol{x}_0)$ and $\psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$.

Question Is there an infinite sequence $\boldsymbol{a}_0, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots$ of configurations and a vector $\boldsymbol{c}$ such that

- $\varepsilon(\boldsymbol{a}_0)$,
- $\boldsymbol{a}_i \to^+ \boldsymbol{a}_j$ for all $0 \leq i < j$, and
- $\psi(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{c})$ for all $1 \leq i < j$?

Here, ε defines a set of initial configurations and ψ relates pairs of configurations while involving some vector of free variables.

If we assume as above that we are dealing with systems where the existential LIRA formulas $\varphi_{p,q}$ for the reachability relation can be effectively constructed from the system description, then linear liveness can be decided using the Ramsey quantifier: There exists a run satisfying the properties of the problem definition if and only if

$$\begin{aligned}\exists \boldsymbol{x}_0, \boldsymbol{z} \colon \varepsilon(\boldsymbol{x}_0) \wedge \Big(\big(\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \varphi(\boldsymbol{x}_0, \boldsymbol{x}) \wedge \varphi(\boldsymbol{x}, \boldsymbol{y}) \wedge \psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\big) \vee \\ \big(\exists \boldsymbol{x} \colon \varphi(\boldsymbol{x}_0, \boldsymbol{x}) \wedge \varphi(\boldsymbol{x}, \boldsymbol{x}) \wedge \psi(\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{z})\big)\Big).\end{aligned} \tag{5.7}$$

Here, the first disjunct states the existence of a sequence of pairwise distinct configurations, whereas the second disjunct expresses the existence of a lasso. Since both φ and ψ are existential formulas in LIRA, we can use Theorem 5.5.1 to eliminate the Ramsey quantifier. Then satisfiability of the resulting existential formula can be checked with Proposition 2.4.6. In the following we will see several applications of this observation for concrete classes of transition systems.

Timed (pushdown) automata The model of timed automata was introduced by Alur and Dill [AD94]. Intuitively, a timed automaton is equipped with a vector of real valued clocks and in each step some time can elapse or, depending on the satisfaction of clock constraints, some clocks can be reset while changing the control state. Before we can give the formal definition, we have to fix some notation.

Let $\boldsymbol{x} = (x_1, \ldots, x_d)$ be a tuple of variables, called *clocks*. We define the set $\Phi(\boldsymbol{x})$ of *clock constraints* as the set of conjunctions of LRA atoms of the form

$$\top, \qquad x_i < k, \qquad x_i = k, \qquad x_i > k \tag{5.8}$$

where $k \in \mathbb{N}_0$, $i \in [1, d]$, and $\top$ denotes the formula that is satisfied for every valuation. Recall that for a set $I \subseteq [1, d]$ of indices $\pi_I(\boldsymbol{x})$ denotes the projection of $\boldsymbol{x}$ to the components in I.

A *timed automaton* is a tuple $\mathcal{A} = (Q, \boldsymbol{x}, \Delta)$ where Q is a finite set of states, $\boldsymbol{x} = (x_1, \ldots, x_d)$ is a tuple of clocks, and $\Delta \subseteq Q \times \Phi(\boldsymbol{x}) \times 2^{[1,d]} \times Q$ is a finite set of transitions. The set of configurations of $\mathcal{A}$ is defined as $C := Q \times \mathbb{R}^d_{\geq 0}$ where $\mathbb{R}_{\geq 0}$ denotes the set of non-negative real numbers. We define the step relation $\to \in C \times C$ such that $(p, \boldsymbol{a}) \to (q, \boldsymbol{b})$ if and only if

(i) $p = q$ and $\boldsymbol{a} + \boldsymbol{r} = \boldsymbol{b}$ for some vector $\boldsymbol{r} = (r, \ldots, r) \in \mathbb{R}^d_{\geq 0}$ or

(ii) there is a transition $(p, \varphi, I, q) \in \Delta$ such that $\varphi(\boldsymbol{a})$, $\pi_I(\boldsymbol{b}) = \boldsymbol{0}$, and $\pi_{[1,d]\setminus I}(\boldsymbol{b}) = \pi_{[1,d]\setminus I}(\boldsymbol{a})$.

A step of the form (i) is called *delay transition* and a step of the form (ii) is called *discrete transition*. We assume that any infinite run $(q_1, \boldsymbol{a}_1) \to (q_2, \boldsymbol{a}_2) \to \cdots$ contains both infinitely many delay and infinitely many discrete transitions and for the added values $r_1, r_2, \ldots$ of the delay transitions we have that $\sum_{i=1}^{\infty} r_i$ diverges [QSW17].

It was shown by Alur and Dill [AD94] that liveness with a Büchi condition on states is PSPACE-complete for timed automata. Later Comon and Jurski [CJ99] showed that the reachability relation $\to^+$ between configurations of timed automata is effectively expressible in LIRA. This result was revisited by Quaas, Shirmohammadi, and Worrell [QSW17], who prove the following:

Proposition 5.6.5. *Given a timed automaton $\mathcal{A} = (Q, \boldsymbol{x}, \Delta)$ and states $p, q \in Q$, one can compute in exponential time an existential formula $\varphi_{p,q}(\boldsymbol{x}, \boldsymbol{y})$ in LIRA with $|\boldsymbol{x}| = |\boldsymbol{y}|$ such that for all configurations $(p, \boldsymbol{a}), (q, \boldsymbol{b}) \in C$ we have that $(p, \boldsymbol{a}) \to^+ (q, \boldsymbol{b})$ if and only if $\varphi_{p,q}(\boldsymbol{a}, \boldsymbol{b})$.*

Proposition 5.6.5 implies that the formula in Equation (5.7) can be constructed in exponential time. Then eliminating the Ramsey quantifier in polynomial time using Theorem 5.5.1 and checking satisfiability of the resulting existential LIRA formula in NP using Proposition 2.4.6 yields:

Corollary 5.6.6. *The linear liveness problem for timed automata is in* NEXP.

Note that the main difference between linear livenss and liveness with a Büchi condition on states, which as mentioned above is PSPACE-complete, is that linear liveness can reason about configurations rather than only states. It can even express conditions between configurations.

In order to model timed behavior of recursive programs, timed automata have been extended with stacks. A *timed pushdown automaton* is a timed automaton enriched with a stack and multiple stack clocks [CL18]. With each push operation the symbol pushed on the stack is associated with a new copy of the stack clocks that are initialized with values specified by a stack constraint. Similarly, on a pop operation the final values of the stack clocks associated with the popped stack symbol are checked against a stack constraint. Here, stack constraints are formulas that can depend on both the global clocks and the stack clocks. Earlier extensions of timed automata with stacks either did not allow stack clocks [BER94] or only allowed a single stack clock where stack constraints could not involve both the stack clock and the global clocks [AAS12]. The two models turned out to be equivalent [CL15]. Clemente and Lasota [CL18] showed that timed pushdown automata as described above are strictly more expressive than the models in [BER94] and [AAS12].

The set of configurations with empty stack of a timed pushdown automaton $\mathcal{A}$ is defined as $C := Q \times \mathbb{Q}_{\geq 0}^d$ where Q is the state set and d is the number of global clocks of $\mathcal{A}$. Note that Clemente and Lasota [CL18] considered rational clock values instead of real ones as defined above for timed automata. The following theorem is shown in [CL18].

Proposition 5.6.7. *Given a timed pushdown automaton $\mathcal{A}$ and states $p, q \in Q$, one can compute in double exponential time an existential formula $\varphi_{p,q}(\boldsymbol{x}, \boldsymbol{y})$ in linear integer rational arithmetic with $|\boldsymbol{x}| = |\boldsymbol{y}|$ such that for all configurations $(p, \boldsymbol{a}), (q, \boldsymbol{b}) \in C$ we have that $(p, \boldsymbol{a}) \to^+ (q, \boldsymbol{b})$ if and only if $\varphi_{p,q}(\boldsymbol{a}, \boldsymbol{b})$.*

Thus, with the same reasoning as for Corollary 5.6.6, where we apply Theorem 5.5.3 instead of Theorem 5.5.1, we obtain the following corollary.

Corollary 5.6.8. *The linear liveness problem for timed pushdown automata is in* 2-NEXP.

Continuous vector addition systems with states Vector addition systems with states (VASSs) are arguably the most popular formal model for concurrent systems. Their configurations consist of a state and a vector of natural numbers representing counter values. These counter values can be updated in each step by adding a vector of integers, where a transition is only enabled if the counter updates do not result in negative counter values. Since the reachability problem for VASSs is Ackermann-complete [LS19a; CO21; Ler21] and the coverability problem is EXPSPACE-complete [Rac78; Lip76], there has been substantial interest in finding overapproximations where these problems become easier.

A particularly successful overapproximation is the continuous semantics. A *continuous vector addition system with states* (CVASS) is a pair $\mathcal{V} = (Q, \Delta)$ where Q is a finite set of states and $\Delta \subseteq Q \times \mathbb{Z}^d \times Q$ is a finite transition relation. We call d the *dimension* of $\mathcal{V}$. The set of configurations of $\mathcal{V}$ is defined as $C := Q \times \mathbb{Q}^d_{\geq 0}$ where the rational components are thought of as the counter values. The step relation $\to \in C \times C$ of $\mathcal{V}$ is defined such that $(p, \boldsymbol{a}) \to (q, \boldsymbol{b})$ if and only if there exist a transition $(p, \boldsymbol{v}, q) \in \Delta$ and a rational number $0 < r \leq 1$ such that $\boldsymbol{a} + r\boldsymbol{v} = \boldsymbol{b}$.

The continuous semantics has been used for reachability to speed up the backward search procedure by pruning configurations that cannot cover the target continuously [Blo+16]. Blondin and Haase [BH17] show that reachability for CVASSs is in NP. More specifically, they show the following:

Proposition 5.6.9. *Given a CVASS $\mathcal{V} = (Q, \Delta)$ of dimension d and states $p, q \in Q$, one can compute in logspace an existential formula $\varphi_{p,q}(\boldsymbol{x}, \boldsymbol{y})$ in linear rational arithmetic with $|\boldsymbol{x}| = |\boldsymbol{y}| = d$ such that for all configurations $(p, \boldsymbol{a}), (q, \boldsymbol{b}) \in C$ we have that $(p, \boldsymbol{a}) \to^+ (q, \boldsymbol{b})$ if and only if $\varphi_{p,q}(\boldsymbol{a}, \boldsymbol{b})$.*

Now, Equation (5.7) together with Theorem 5.4.6 imply that linear liveness for CVASSs has the same complexity as reachability.

Corollary 5.6.10. *The linear liveness problem for CVASSs is* NP*-complete.*

Here, the lower bound follows from NP-hardness of the reachability problem for CVASSs [BH17].

Succinct one-counter automata In the presence of discrete counters there are several classes of systems whose reachability relation is expressible in existential Presburger arithmetic. The simplest example of such systems are automata equipped with a single discrete counter.

Fix a variable x, called *counter*. We denote the set of Boolean combinations of Presburger atoms as in Equation (5.8) with variable x by Φ. A *succinct one-counter automaton* (SOCA) is a tuple $\mathcal{A} = (Q, \Delta)$ where Q is a finite set of states and $\Delta \subseteq Q \times \Phi \times \mathbb{Z} \times Q$ is a finite transition relation. We define the set of configurations of $\mathcal{A}$ as $C := Q \times \mathbb{N}_0$. The step relation $\to \in C \times C$ is defined such that $(p, a) \to (q, b)$ if and only if there is a transition $(p, \varphi, v, q) \in \Delta$ with $\varphi(a)$ and $a + v = b$. Note that we call the model *succinct* since we assume that the counter updates and the constants in the formulas from Φ are encoded in binary.

It was shown by Haase et al. [Haa+09] that the reachability problem for SOCAs is NP-complete. Based on that, Li et al. [Li+20] proved that an existential Presburger formula for the reachability relation can be computed in polynomial time.

Proposition 5.6.11. *Given an SOCA $\mathcal{A} = (Q, \Delta)$ and states $p, q \in Q$, one can compute in polynomial time an existential Presburger formula $\varphi_{p,q}(x, y)$ such that for all configurations $(p, a), (q, b) \in C$ we have that $(p, a) \to^+ (q, b)$ if and only if $\varphi_{p,q}(a, b)$.*

Thus, we can use Equation (5.7) and Theorem 5.3.1 to decide linear liveness for SOCAs in NP.

Corollary 5.6.12. *The linear liveness problem for SOCAs is* NP*-complete.*

Again, NP-hardness follows from the the reachability problem.

Reversal-bounded counter machines If we want to equip an automaton with more than one counter, we have to impose restrictions on the additional counters since an automaton with two unrestricted counters is already Turing-complete [Min67], meaning that the reachability problem is undecidable. The most prominent example of automata with restricted counters are reversal-bounded counter machines [Iba78].

Let $\boldsymbol{x} = (x_1, \ldots, x_d)$ be a tuple of variables, called *counters*. We denote the set of Boolean combinations of Presburger atoms as in Equation (5.8) by $\Phi(\boldsymbol{x})$. A sequence in $\mathbb{N}_0^d$ is called *r-reversal-bounded* for some $r \in \mathbb{N}_0$ if the sequence switches between non-decreasing and non-increasing at most r times in each component. For example, the sequence $1, 1, 2, 5, 5, 3, 1, 2$ has two reversals.

A *reversal-bounded counter machine* (RBCM) is a tuple $\mathcal{A} = (r, Q, \boldsymbol{x}, \Delta)$ where $r \in \mathbb{N}_0$ is the *reversal bound*, Q is a finite set of states, $\boldsymbol{x} = (x_1, \ldots, x_d)$ is a tuple of counters, and $\Delta \subseteq Q \times \Phi(\boldsymbol{x}) \times \mathbb{Z}^d \times Q$ is a finite transition relation. The set of configurations of $\mathcal{A}$ is defined as $C := Q \times \mathbb{N}_0^d$. We define the step relation $\to \in C \times C$ such that $(p, \boldsymbol{a}) \to (q, \boldsymbol{b})$ if and only if there is a transition $(p, \varphi, \boldsymbol{v}, q) \in \Delta$ with $\varphi(\boldsymbol{a})$ and $\boldsymbol{a} + \boldsymbol{v} = \boldsymbol{b}$. This time, the reachability relation $\to^+ \in C \times C$ of $\mathcal{A}$ is *not* the transitive closure of $\to$. For configurations $(p, \boldsymbol{a}), (q, \boldsymbol{b}) \in C$ we let $(p, \boldsymbol{a}) \to^+ (q, \boldsymbol{b})$ if and only if there is a run $(q_1, \boldsymbol{a}_1) \to \cdots \to (q_n, \boldsymbol{a}_n)$ of $\mathcal{A}$ with $q_1 = p$, $\boldsymbol{a}_1 = \boldsymbol{a}$, $q_n = q$, and $\boldsymbol{a}_n = \boldsymbol{b}$ such

that the sequence $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ is r-reversal-bounded. Note that $\to^+$ is not transitive in general. We say that an RBCM has *one unrestricted counter* if one of the counters can have arbitrarily many reversals. Here, note that the updates in Δ and the constants in formulas from $\Phi(\boldsymbol{x})$ that refer to the unrestricted counter are encoded in unary (and not in binary as for the reversal-bounded counters).

It was shown by Ibarra et al. [Iba+00] that the reachability relation of RBCMs with one unrestricted counter is definable by an existential Presburger formula. Hague and Lin [HL11] proved that the corresponding existential Presburger formula can even be computed in polynomial time.

Proposition 5.6.13. *Given an RBCM $\mathcal{A} = (r, Q, \boldsymbol{x}, \Delta)$ with one unrestricted counter and states $p, q \in Q$, one can compute in polynomial time an existential Presburger formula $\varphi_{p,q}(\boldsymbol{x}, \boldsymbol{y})$ with $|\boldsymbol{x}| = |\boldsymbol{y}| = d$ such that for all configurations $(p, \boldsymbol{a}), (q, \boldsymbol{b}) \in C$ we have that $(p, \boldsymbol{a}) \to^+ (q, \boldsymbol{b})$ if and only if $\varphi_{p,q}(\boldsymbol{a}, \boldsymbol{b})$.*

Thus, we can use Equation (5.7) and Theorem 5.3.1 to decide the linear liveness problem.

Corollary 5.6.14. *The linear liveness problem for RBCMs with one unrestricted counter is* NP*-complete.*

Here, the lower bound follows again from a reduction from the reachability problem, which is already NP-hard for RBCMs with a single counter with reversal bound one [HL11].

Parikh automata A model that is equally expressive as reversal-bounded counter machines are so-called Parikh automata [KR03]. Intuitively, a Parikh automaton is an NFA equipped with multiple counters to which in each transition natural numbers can be added. However, transitions are taken independently of the current counter values and only at the end of the run it can be checked whether the vector of counter values is contained in a semilinear set.

We call a set $L \subseteq \mathbb{N}_0^d$ *linear* if there are vectors $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_n \in \mathbb{N}_0^d$ such that $L = \{\boldsymbol{v}_0 + \sum_{i=1}^n c_i \boldsymbol{v}_i \mid c_1, \ldots, c_n \in \mathbb{N}_0\}$. A set $S \subseteq \mathbb{N}_0^d$ is *semilinear* if it is a finite union of linear sets. Let Σ be an alphabet and $D \subseteq \mathbb{N}_0^d$ be finite. For a word $w = (a_1, \boldsymbol{v}_1) \ldots (a_n, \boldsymbol{v}_n) \in (\Sigma \times D)^*$ we write $\pi_\Sigma(w) := a_1 \ldots a_n \in \Sigma^*$ for the projection to the Σ-component and $\Phi(w) := \boldsymbol{v}_1 + \cdots + \boldsymbol{v}_n \in \mathbb{N}_0^d$ for the *extended Parikh image*, where $\Phi(\varepsilon) := \boldsymbol{0}$.

A *Parikh automaton* is a pair $(\mathcal{A}, S)$ where $\mathcal{A} = (Q, \Sigma \times D, \Delta, q_0, F)$ is an NFA and $S \subseteq \mathbb{N}_0^d$ is a semilinear set. The language accepted by $(\mathcal{A}, S)$ is

$$L(\mathcal{A}, S) := \{\pi_\Sigma(w) \mid w \in L(\mathcal{A}) \text{ and } \Phi(w) \in S\}.$$

The set of configurations of $(\mathcal{A}, S)$ is $C := Q \times \mathbb{N}_0^d$. We defined the step relation such that $(p, \boldsymbol{a}) \to (q, \boldsymbol{b})$ if and only if there exists a transition $(p, (a, \boldsymbol{v}), q) \in \Delta$ with $\boldsymbol{a} + \boldsymbol{v} = \boldsymbol{b}$. It was shown by Seidl et al. [Sei+04] that one can compute in polynomial time an existential Presburger formula for the Parikh image of a regular language given by an NFA (even of context-free languages [VSS05]). It is easy to see that the proof can be adapted to the

extended Parikh image. Thus, since $(p, \boldsymbol{a}) \to^+ (q, \boldsymbol{b})$ if and only if there is a run from p to q reading $w \in (\Sigma \times D)^*$ such that $\boldsymbol{a} + \Phi(w) = \boldsymbol{b}$, the reachability relation can also be expressed by an existential Presburger formula.

Proposition 5.6.15. *Given a Parikh automaton $(\mathcal{A}, S)$ and states $p, q \in Q$, one can compute in polynomial time an existential Presburger formula $\varphi_{p,q}(\boldsymbol{x}, \boldsymbol{y})$ with $|\boldsymbol{x}| = |\boldsymbol{y}| = d$ such that for all configurations $(p, \boldsymbol{a}), (q, \boldsymbol{b}) \in C$ we have that $(p, \boldsymbol{a}) \to^+ (q, \boldsymbol{b})$ if and only if $\varphi_{p,q}(\boldsymbol{a}, \boldsymbol{b})$.*

Again, using Equation (5.7) and Theorem 5.3.1, it follows:

Corollary 5.6.16. *The linear liveness problem for Parikh automata is* NP*-complete.*

Parikh automata over infinite words were recently introduced by Guha et al. [Guh+22]. The authors show that the non-emptiness problem for Büchi Parikh automata is NP-complete, where in the asynchronous semantics an infinite run $(q_1, \boldsymbol{a}_1) \to (q_2, \boldsymbol{a}_2) \to \dots$ is considered accepting if there exist infinitely many $i \geq 1$ with $q_i \in F$ and $\boldsymbol{a}_i \in S$. Since semilinear sets and Presburger-definable sets over $\mathbb{N}_0$ coincide [GS66a] and from any semilinear set one can compute in polynomial time an existential Presburger formula, non-emptiness can be checked using linear liveness. Thus, Corollary 5.6.16, in particular, yields the precise complexity for the non-emptiness problem for Büchi Parikh automata.

5.6.4 Deciding Whether a Relation Is a WQO

The notion of *well-structured transition systems* (WSTSs) [Abd+96; FS01] plays an important role in the verification of infinite-state systems. They are used to impose sufficient conditions on transition systems such that certain verification problems become decidable. Loosely speaking, a WSTS is a transition system equipped with a well-quasi-ordering (WQO) on its configurations that is compatible with the step relation. One well-known example of WSTSs are VASSs, which we already encountered above. Recently, Finkel and Gupta [FG19a] considered the problem of automatically checking whether a transition system is well-structured. In particular, they raise the question of how to decide whether a relation specified by a Presburger formula is a WQO. Using Theorem 5.3.1, we can show that this problem is coNP-complete if the relation is given by a quantifier-free Presburger formula.

Corollary 5.6.17. *Given a quantifier-free Presburger formula $\varphi(\boldsymbol{x}, \boldsymbol{y})$ with $|\boldsymbol{x}| = |\boldsymbol{y}| = d$, it is* coNP*-complete to decide whether $[\![\varphi]\!] \subseteq \mathbb{Z}^d \times \mathbb{Z}^d$ is a WQO.*

Proof. Recall that a relation $R \subseteq \mathbb{Z}^d \times \mathbb{Z}^d$ is a WQO if and only if it is reflexive and transitive and for every infinite sequence $(\boldsymbol{a}_i)_{i \geq 1}$ in $\mathbb{Z}^d$ there are $1 \leq i < j$ with $(\boldsymbol{a}_i, \boldsymbol{a}_j) \in R$. Thus, φ violates the conditions of a WQO if and only if

(i) $\exists \boldsymbol{x} \colon \neg\varphi(\boldsymbol{x}, \boldsymbol{x})$ (reflexivity violation) or

(ii) $\exists \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \colon \varphi(\boldsymbol{x}, \boldsymbol{y}) \wedge \varphi(\boldsymbol{y}, \boldsymbol{z}) \wedge \neg\varphi(\boldsymbol{x}, \boldsymbol{z})$ (transitivity violation) or

(iii) $\exists^{\mathrm{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \neg\varphi(\boldsymbol{x}, \boldsymbol{y})$ (violation of the sequence condition).

Thus, using Theorem 5.3.1 and Proposition 2.4.1, we can check in NP whether φ does not define a WQO.

To prove the lower bound, we reduce from the unsatisfiability problem of quantifier-free Presburger formulas, which by Proposition 2.4.1 is coNP-complete. Suppose we are given a quantifier-free Presburger formula $\psi(\boldsymbol{x})$ with $|\boldsymbol{x}| = d$. We define the formula

$$\begin{aligned}\varphi((x,\boldsymbol{x}),(y,\boldsymbol{y})) := (x = 0 \wedge y = 0) \vee (x < 0 \wedge y < 0) \vee (x > 0 \wedge y > 0) \vee \\ (x < 0 \wedge y = 0) \vee (x = 0 \wedge y > 0 \wedge \psi(\boldsymbol{y}))\end{aligned}$$

where $|\boldsymbol{y}| = d$. We claim that ψ is unsatisfiable if and only if $[\![\varphi]\!] \subseteq \mathbb{Z}^{d+1} \times \mathbb{Z}^{d+1}$ is a WQO.

Suppose ψ is unsatisfiable. Then $\varphi((a,\boldsymbol{a}),(b,\boldsymbol{b}))$ holds for some $a, b \in \mathbb{Z}$ and $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}^d$ if and only if a and b have the same sign (or are both zero) or $a < 0$ and $b = 0$. Let $A \cup B \cup C = \mathbb{Z}^{k+1}$ be a partition such that A contains the vectors with negative first component, B contains the vectors with 0 in the first component, and C contains the vectors with positive first component. Within each set A, B, C we have that φ relates every pair of vectors. Moreover, for all $(a,\boldsymbol{a}) \in A$ and $(b,\boldsymbol{b}) \in B$ we have $\varphi((a,\boldsymbol{a}),(b,\boldsymbol{b}))$ and the vectors in C are unrelated to those in $A \cup B$. Thus, $[\![\varphi]\!]$ is clearly a WQO.

Conversely, if ψ is satisfiable, say $\psi(\boldsymbol{a})$ holds for some $\boldsymbol{a} \in \mathbb{Z}^d$, then $[\![\varphi]\!] \subseteq \mathbb{Z}^{d+1} \times \mathbb{Z}^{d+1}$ is not transitive: For example, we have $\varphi((-1,\boldsymbol{0}),(0,\boldsymbol{0}))$ and $\varphi((0,\boldsymbol{0}),(1,\boldsymbol{a}))$ but not $\varphi((-1,\boldsymbol{0}),(1,\boldsymbol{a}))$. □

Decidability was already shown in [FG19b], where they use the fact that Presburger arithmetic is an automatic structure and Ramsey quantifiers in automatic structures can be evaluated. This approach however leads to higher complexity: We first have to construct in polynomial space an automaton for the negation of a given quantifier-free Presburger formula and then check condition (iii) in the proof of Corollary 5.6.17, where the Ramsey quantifier can be evaluated using Theorem 4.2.1. This results in a PSPACE-algorithm.

5.7 Experiments

We have implemented a prototype, which can be found at [Ber+23], of our Ramsey quantifier elimination algorithms for LIA, LRA, and LIRA in Python using the Z3 [MB08] interface Z3Py. We have tested it against two sets of micro-benchmarks. The first benchmark set contains the following examples, where the dimension d of $\boldsymbol{x}$ and $\boldsymbol{y}$ is a parameter:

(a) $\varphi_{\text{half}} := \exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon 2\boldsymbol{y} \leq \boldsymbol{x} \wedge \boldsymbol{x} \geq \boldsymbol{t}$ for parameter $t \in \mathbb{Z}$

(b) $\varphi_{\text{eq\_ex}} := \exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \exists \boldsymbol{z} \colon \boldsymbol{x} \ll \boldsymbol{y} \wedge \boldsymbol{x} = \boldsymbol{z}$

(c) $\varphi_{\text{eq\_free}} := \exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \boldsymbol{x} \ll \boldsymbol{y} \wedge \boldsymbol{x} = \boldsymbol{z}$

(d) $\varphi_{\text{dickson}} := \exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \boldsymbol{x} \geq \boldsymbol{0} \wedge (\boldsymbol{x} > \boldsymbol{y} \vee \boldsymbol{x} \not\leq \boldsymbol{y} \wedge \boldsymbol{y} \not\leq \boldsymbol{x})$, where unsatisfiability over $\mathbb{Z}$ proves Dickson's lemma

formula	dom	sat	input		output						
			#vars	#atoms	#vars	#atoms	$d=1$	$d=10$	$d=20$	$d=50$	$d=100$
φ_{half}	$\mathbb{Z}$	no	$2d$	$2d$	$22d$	$130d$	0.04s	0.33s	0.84s	3.35s	11.00s
	$\mathbb{R}$	$t \leq 0$			$25d$	$284d$	0.06s	0.75s	2.10s	10.31s	38.48s
$\varphi_{\text{eq\_ex}}$	$\mathbb{Z}$	yes	$3d$	$2d$	$31d$	$166d$	0.05s	0.67s	2.01s	10.23s	39.44s
	$\mathbb{R}$	yes			$25d$	$213d$	0.05s	1.03s	3.53s	20.01s	82.46s
$\varphi_{\text{eq\_free}}$	$\mathbb{Z}$	no	$3d$	$2d$	$28d$	$162d$	0.05s	0.44s	1.12s	4.73s	16.11s
	$\mathbb{R}$	no			$20d$	$209d$	0.08s	0.54s	1.64s	8.18s	31.03s
φ_{dickson}	$\mathbb{Z}$	no	$2d$	$5d$	$37d$	$226d$	0.06s	0.58s	1.52s	6.33s	21.60s
	$\mathbb{R}$	yes			$40d$	$482d$	0.08s	1.17s	4.48s	17.18s	66.46s
φ_{prog}	$\mathbb{R},\mathbb{Z}$	yes	$6d$	$14d$	$426d+1$	$3858d+4$	0.84s	68.28s	445.89s	>500s	>500s

Table 5.1: Experiments for the elimination of the Ramsey quantifier with a 500 seconds timeout.

(e) $\varphi_{\text{prog}} := \exists^{\text{ram}}(\boldsymbol{x}_1,\boldsymbol{x}_2),(\boldsymbol{y}_1,\boldsymbol{y}_2)\colon \boldsymbol{x}_1 \gg \mathbf{0} \wedge \boldsymbol{x}_2 \gg \mathbf{0} \wedge \boldsymbol{y}_1 \geq \mathbf{0.5}\boldsymbol{x}_1 + \mathbf{0.5} \wedge \boldsymbol{y}_2 \leq \boldsymbol{x}_2 - \lfloor \mathbf{0.5}\boldsymbol{x}_1 + \mathbf{0.5} \rfloor$ expressing the reachability relation of the non-terminating program in Algorithm 1, where $\boldsymbol{x}_1, \boldsymbol{y}_1$ are vectors of real variables and $\boldsymbol{x}_2, \boldsymbol{y}_2$ are vectors of integer variables

Here, $\lfloor \boldsymbol{v} \rfloor$ for a vector $\boldsymbol{v} = (v_1, \ldots, v_d)$ denotes the vector $(\lfloor v_1 \rfloor, \ldots, \lfloor v_d \rfloor)$.

The experiments were conducted on an Intel(R) Core(TM) i7-10510U CPU with 16GB of RAM running on Windows 10. The results are summarized in Table 5.1. We observe that the number of output variables and atoms linearly depends on the number of input variables and atoms. In the first three cases, the output formula has ca. 5 times as many variables as the input has variables plus atoms. The choice of parameter $t \in \mathbb{Z}$ has no notable effect on the size of the output formula or the running time since it only changes constants. For φ_{prog} our prototype implementation assumes the formula to be already decomposed into a Boolean combination of LIA and LRA formulas, whose size is given in the input column of Table 5.1. Then the running time is dominated by the Z3 satisfiability check due to the large number of variables and atoms in the output.

For the second benchmark set we used our elimination procedure to check monadic decomposability, as described in Section 5.6.1, of the following formulas:

(a) $\varphi_{\text{imp}} := \bigwedge_{i=1}^{d} x_i \geq 0 \rightarrow x_i + y_i \geq k \wedge y_i \geq 0$ for parameter $k \in \mathbb{N}$

(b) $\varphi_{\text{diagonal}} := \mathbf{0} \leq \boldsymbol{x} \leq \boldsymbol{k} \wedge x_1 = \cdots = x_d$ for parameter $k \in \mathbb{N}$

(c) $\varphi_{\text{cubes2d}} := x_1 + x_2 \leq k \wedge \left(\bigvee_{i=1}^{k} i \leq x_1 \leq i+2 \wedge i \leq x_2 \leq i+2 \right)$ where parameter $k \in \mathbb{N}$ is the number of cubes

(d) $\varphi_{\text{cubes10}} := \bigvee_{i=1}^{10} \boldsymbol{i} \leq \boldsymbol{x} \leq \boldsymbol{i} + \mathbf{2}$

(e) $\varphi_{\text{mixed}} := \boldsymbol{x} = \lfloor \boldsymbol{y} \rfloor \wedge \mathbf{0} \leq \boldsymbol{y} \leq \boldsymbol{k}$ over LIRA with parameter $k \in \mathbb{N}$

The results are shown in Table 5.2, where either the dimension d or parameter k is varied. The size of the input refers to the formula $\not\approx_{[1,d]\setminus\{j\}}$ for $j = 1$ that expresses the complement of the relation $\approx_{[1,d]\setminus\{j\}}$. The reason why we do not use $\not\approx_j$ as in the proof of Corollary 5.6.1 is that $\not\approx_{[1,d]\setminus\{j\}}$ only has one existentially quantified variable, which

formula	dom	mondec	input		output					
			#vars	#atoms	#vars	#atoms				
φ_{imp}	$\mathbb{Z}$	yes	$4d-1$	$12d$	$84d-8$	$516d-62$	$d=1$	$d=5$	$d=10$	$d=20$
							0.12s	6.19s	34.11s	224.53s
	$\mathbb{R}$	no			$136d-7$	$1678d-130$	0.32s	2.27s	6.69s	19.01s
$\varphi_{\text{diagonal}}$							$d=2$	$d=10$	$d=20$	$d=30$
	$\mathbb{Z}$	yes	$2d-1$	$4d+4$	$52d-8$	$322d-62$	0.15s	8.20s	46.48s	151.42s
	$\mathbb{R}$	no			$35d+59$	$416d+716$	0.32s	1.33s	3.82s	7.37s
φ_{cubes2d}							$k=50$	$k=100$	$k=150$	$k=250$
	$\mathbb{Z}$	yes	3	$16k+4$	$80k+36$	$512k+198$	7.74s	18.77s	37.73s	109.17s
	$\mathbb{R}$	no			$176k+63$	$2256k+702$	210.12s	>500s	>500s	>500s
φ_{cubes10}							$d=2$	$d=10$	$d=15$	$d=20$
	$\mathbb{Z}$	yes	$2d-1$	$80d$	$412d-8$	$2626d-62$	1.18s	66.40s	231.67s	482.39s
	$\mathbb{R}$	yes			$893d-7$	$11414d-130$	5.83s	>500s	>500s	>500s
φ_{mixed}							$d=1$	$d=2$	$d=3$	$d=4$
	$\mathbb{R},\mathbb{Z}$	yes	$6d-1$	$28d$	$842d-28$	$7710d-198$	3.67s	42.84s	192.76s	>500s

Table 5.2: Experiments for monadic decomposability with a 500 seconds timeout.

has the advantage that the algorithm only has to eliminate one existential quantifier before eliminating the Ramsey quantifier. For the output we measure the size of the first formula given to Z3, i.e. $\exists^{\text{ram}} \boldsymbol{x}, \boldsymbol{y} \colon \boldsymbol{x} \not\approx_{[2,d]} \boldsymbol{y}$ after the elimination of the Ramsey quantifier. We observe that if n is the number of input variables plus atoms, on these instances the number of output variables can be estimated by $5 \cdot n$ over $\mathbb{Z}$ and $10 \cdot n$ over $\mathbb{R}$. The slowdown compared to Table 5.1 is partly due to the larger input formulas, on which the Ramsey quantifier is eliminated, but also to the fact that in case the formula is monadically decomposable, one has to consider $\not\approx_{[1,d]\setminus\{j\}}$ for *all* $j \in [1,d]$.

For φ_{imp} and $\varphi_{\text{diagonal}}$ we observe that, despite the larger output formula, over $\mathbb{R}$ the algorithm terminates significantly faster than over $\mathbb{Z}$ since it only needs to consider $\not\approx_{[2,d]}$ to detect that φ is not monadically decomposable. The first four examples over the integers are taken from [MSL21], where the authors compare their tool to mondec_1 from [Vea+17] that *computes* a monadic decomposition if one exists. We observe that on these instances our *decision* algorithm is significantly faster than mondec_1, especially for φ_{imp} and $\varphi_{\text{diagonal}}$ when only the parameter k is varied (and $d=1$ respectively $d=2$ as in [MSL21]). The reason for this is that mondec_1 computes the monadic decomposition, whose size grows exponentially in the encoding of k, whereas in our approach, where we only decide if a decomposition exists, k only changes a constant in the formulas where the Ramsey quantifier is eliminated. Therefore, changing k in the two examples (and also in φ_{mixed}) does not have any notable effect on the running time in Table 5.2. In this case, our algorithm is also faster than the one developed in [MSL21], that outputs the decomposition in form of cubes. Since both algorithms in [Vea+17] and [MSL21] only terminate if the input formula is monadically decomposable, our algorithm can be used as a termination check. Finally, note that the increase of the running time for φ_{cubes2d}, φ_{cubes10} over $\mathbb{R}$ and φ_{mixed} is due to the large number of atoms in the output, which is problematic not only for the elimination procedure but especially for the satisfiability check with Z3. We observe that for large instances the running time is dominated by the satisfiability check.

6 Recognizability in Subclasses of Rational Relations

Parts of this chapter were published by the author in [BG23].

We already observed in previous chapters that our results on infinite cliques can be applied to (tree-)reconizability of (tree-)regular relations and the related problem of monadic decomposability. For this, we exploited the connection between recognizability and checking finite index of certain equivalence relations. We will see that the same strategy can be used to obtain precise complexity results for ω-recognizability of ω-regular relations. Here, similar to tree-recognizability, we make use of the fact that we only have to deal with complements of equivalence relations. In fact, checking for an infinite clique in arbitrary ω-regular relations is a longstanding open problem. One central observation is that a co-equivalence relation has an infinite clique if and only if it has unbounded cliques, i.e. arbitrarily large but finite cliques. This allows us to focus on special cliques consisting of ultimately periodic words, whose existence can be checked in the automaton for the co-equivalence relation.

Over finite words we already determined the precise complexity of the recognizability problem for regular relations. For the more general class of deterministic rational relations, however, the precise complexity is not known. For this problem one could try to use the same technique as for (ω-)regular relations and reduce the problem to the infinite clique problem. Here, however, we encounter two issues. Firstly, deterministic rational relations do not enjoy the same closure properties as regular relations and the co-equivalence relations $\not\approx_j$ are not necessarily deterministic rational. In fact, for relations of arity greater than two it is not even clear whether $\not\approx_j$ is rational. For binary relations at least it can be observed that $\not\approx_1$ is a rational relation. The second issue is that it is not clear whether the infinite clique problem for rational relations is decidable at all, even if restricted to co-equivalence relations. For this reason, we use a more direct approach, where we find characterizations of when $\not\approx_j$ has finite index. These alternative characterizations will allow us to identify patterns in the deterministic multitape automaton that witness *non*-recognizability of the accepted relation. Checking absence of these patterns will then lead to improved complexity upper bounds.

A problem that is not even known to be decidable is checking whether a deterministic rational relation is regular. We show that this problem can be reduced to recognizability of deterministic rational relations, which yields the first decision procedure for it, solving an open problem from [CCG06; IT12]. The idea is that as long as the delay between heads of the automaton is bounded, the computation can be simulated synchronously. If two heads can move arbitrarily far apart, then we show that they have to be independent from each other, which can be checked using the notion of recognizability.

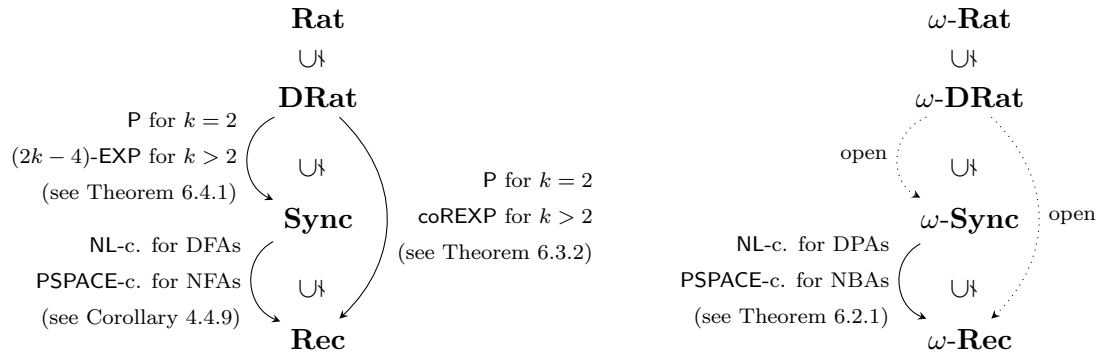


Figure 6.1: The complexity landscape of deciding membership to subclasses of (ω-)rational relations. An arrow from $\mathcal{C}_2$ to $\mathcal{C}_1$ refers to the problem of deciding whether a given relation from $\mathcal{C}_2$ belongs to $\mathcal{C}_1$. Membership of (ω-)rational relations in any one of the three subclasses is undecidable. Dotted arrows mean that decidability of the problem is unknown. The parameter k denotes the arity of the given relation.

6.1 Main Results

As we have already seen in Section 4.4.2, Ramsey quantifiers or, more specifically, the infinite clique problem can be used to decide recognizability for regular relations. That is, given a k-ary regular relation R, we can check whether $R = \bigcup_{i=1}^{n} L_{i,1} \times \cdots \times L_{i,k}$ for regular languages $L_{i,j}$. In other words, we showed that the *membership problem* of relations from **Reg** in the strict subclass **Rec** is decidable. A natural question to ask is whether the infinite clique problem can also help to solve other membership problems in the hierarchies of classes of relations defined in Sections 2.3.5 and 2.3.6 over finite and infinite words. Figure 6.1 provides an overview of the complexity landscape that we obtain after this chapter. Here, all membership problems of (ω-)rational relations in any of the three subclasses are known to be undecidable [FR68; Lis79].

Recognizability of ω-regular relations Corollary 4.4.9 shows that, given a regular relation R by a DFA (resp. NFA), one can decide in nondeterministic logspace (resp. polynomial space) whether R is recognizable. For the proof we used the characterization from Proposition 3.3.7 of recognizable relations via finite-index equivalence relations $\approx_j^R$. Then we applied Corollary 4.2.2 to check for an infinite clique in the complements $\not\approx_j^R$. Decidability of the infinite clique problem for arbitrary ω-regular relations, however, is a longstanding open problem. Nevertheless, in Section 6.2 we are able to prove the precise complexity of ω-recognizability for ω-regular relations. Interestingly, the complexity matches the complexity of the recognizability problem for regular relations over finite words.

Theorem 6.2.1. *Given an ω-regular relation $R \subseteq (\Sigma^\omega)^k$ by a DPA (resp. NBA), it is* NL*-complete (resp.* PSPACE*-complete) to decide whether R is ω-recognizable.*

For the proof we use the fact that $\not\approx_j^R$ is the complement of an equivalence relation rather than being an arbitrary ω-regular relation. This is similar to the tree case, where in order to prove the precise complexity in Corollary 4.4.10, we could not just apply the generic algorithm for the infinite clique problem in arbitrary tree-regular relations, but we had to develop a specialized algorithm for co-equivalence relations (Theorem 4.3.13). We will see that such a specialized algorithm for ω-regular co-equivalence relations can also be utilized to prove Theorem 6.2.1.

Theorem 6.2.2. *The infinite clique problem over ω-regular co-equivalence relations $\overline{E} \subseteq (\Sigma^\omega)^{2d}$ is* NL*-complete.*

To prove Theorem 6.2.2, we follow the approach by Löding and Spinrath [LS19b] and solve the *unbounded cliques problem* for ω-regular co-equivalence relations $\overline{E}$. This means that we check whether $\overline{E}$ contains cliques of arbitrary size, which in case of co-equivalence relations is equivalent to the existence of an infinite clique. The algorithm in [LS19b] constructs an automaton for a regular set of ultimately periodic representatives of E, whose size is double (triple) exponential in the size of a (non)deterministic automaton for $\overline{E}$. We circumvent construction of this large automaton and identify a simple pattern directly in the automaton for $\overline{E}$ which witnesses the existence of unbounded cliques.

Recognizability of deterministic rational relations The remaining results of this chapter concern membership problems for deterministic rational relations over finite words. In Section 6.3 we consider the recognizability problem for deterministic rational relations. We will revisit the proof by Carton, Choffrut, and Grigorieff [CCG06], who showed decidability for relations of arbitrary arity and that for binary relations a double exponential-time upper bound can be derived from [Val75]. With our approach we obtain the following improved complexity bounds.

Theorem 6.3.2. *Given a deterministic rational relation $R \subseteq (\Sigma^*)^k$, one can decide whether R is recognizable (i) in* P *if $k = 2$, (ii) in* coREXP *if $k > 2$ is fixed, and (iii) in* coNEXP *if k is part of the input.*

Here, coREXP $\subseteq$ coNEXP is the class of all decision problems that can be solved by a randomized algorithm in exponential time, which may err on negative instances with probability at most $1/2$.

For the proof of Theorem 6.3.2 we identify patterns in the deterministic multitape automaton for R that are present if and only if R is not recognizable. We then reduce the check for absence of these patterns to the equivalence problem for deterministic multitape automata. If R is binary, the reduction works in logspace. For higher arity it requires polynomial space. To decide recognizability, we can then use known upper bounds on the complexity of the equivalence problem. It was shown by Friedman and Greibach [FG82] that equivalence of two deterministic 2-tape automata can be checked in polynomial time. Whereas for deterministic k-tape automata, where k is part of the input, only a coNP upper bound is known [KKS15]. If the arity $k < 2$ is fixed, then equivalence can be decided by a randomized polynomial-time algorithm that may

err with probability at most 1/2 if the given deterministic k-tape automata are non-equivalent [Wor13].

Interestingly, there is also a reduction in the reverse direction, i.e. a reduction from the equivalence problem for deterministic multitape automata to recognizability of deterministic rational relations. We show that this reduction works in logspace, which means that in the binary case the two problems are logspace equivalent.

Theorem 6.3.8. *Let $k \geq 2$. The equivalence problem for deterministic k-tape automata is logspace reducible to the recognizability problem for k-ary deterministic rational relations.*

The proof essentially follows from a result by Friedman and Greibach [FG78], which reduces the equivalence problem for DPDAs restricted to a subclass $\mathcal{C}$ to the membership problem of DPDAs to $\mathcal{C}$.

So far, we only considered the problem of *deciding* whether a relation is recognizable. Barceló et al. [Bar+19] raised the question of how to actually *output* a witness for recognizability using some appropriate representation. We will answer this question for deterministic rational relations, where the witness will be an independent multitape automaton, i.e. given a deterministic multitape automaton accepting a recognizable relation R, we compute an independent multitape automaton for R. Recall that independent multitape automata accept precisely the recognizable relations. Note that this also yields an algorithm in the case of regular relations (considered in Section 4.4.2 and also in [Bar+19]) since they form a subclass of deterministic rational relations.

Theorem 6.3.9. *Given a deterministic k-tape automaton for a recognizable relation R, one can compute in double exponential time an independent k-tape automaton for R.*

The proof is based on ideas from [CCG06] and imitates the construction by Valiant [Val75] of a double exponentially large DFA from a DPDA accepting a regular language. It seems that the missing piece for the construction is our characterization of recognizability via the equivalence relations $\approx_j$ instead of $\approx_{[1,j]}$. Moreover, we show that the double exponential blow-up in Theorem 6.3.9 is unavoidable in general.

Regularity of deterministic rational relations As a final result of this chapter, we close the remaining gap in the decidability landscape of the membership problems over finite-word relations (Figure 6.1), which was left open in [CCG06] and answers a question by Ibarra and Trân [IT12]. In Section 6.4 we prove that regularity of deterministic rational relations is decidable by reducing the problem to the recognizability problem for deterministic rational relations, enabling us to use Theorem 6.3.2.

Theorem 6.4.1. *Given a deterministic rational relation $R \subseteq (\Sigma^*)^k$, one can decide whether R is regular (i) in* P *if $k = 2$ and (ii) in $(2k-4)$-*EXP *if $k > 2$ is fixed.*

Intuitively, in order to accept a regular relation, the heads of a deterministic multitape automaton either only have *bounded delay* throughout the computation, i.e. they cannot move arbitrarily far apart, or are independent from each other. It is easy to see that

bounded delay implies that the corresponding tapes can be read synchronously by storing the symbols that are "read ahead" in a queue of bounded length and the heads of the deterministic multitape automaton can therefore be simulated by an NFA. If two heads are independent, then an NFA can also simulate them in a synchronous manner. To check whether heads have bounded delay, it suffices to ensure that every cycle reads the same number of symbols in the corresponding components. In case there is a cycle where the number of read symbols differs between components, we can use the recognizability test from Theorem 6.3.2 to check whether the heads with unbounded delay are independent.

6.2 ω-Recognizability of ω-Regular Relations

In this section we settle the precise complexity of the ω-recognizability problem for ω-regular relations. The complexity is the same as over finite words (Corollary 4.4.9), i.e. it depends on whether the given automaton is deterministic or nondeterministic. Decidability in double exponential time if the relation is given by a DPA was already shown in [LS19b].

Theorem 6.2.1. *Given an ω-regular relation $R \subseteq (\Sigma^\omega)^k$ by a DPA (resp. NBA), it is* NL*-complete (resp.* PSPACE*-complete) to decide whether R is ω-recognizable.*

The lower bounds are inherited from the finite-word case using an infinite padding: Let $R \subseteq (\Sigma^*)^k$ be a regular relation and $\bot \notin \Sigma$. Then R is recognizable if and only if $R' := \{(w_1\bot^\omega, \ldots, w_k\bot^\omega) \mid (w_1, \ldots, w_k) \in R\}$ is ω-recognizable. Moreover, a DPA (resp. NBA) for R' can be constructed in logspace from a DFA (resp. NFA) for R.

To prove the upper bounds, we use the same technique as for Corollary 4.4.9. That is, we use the characterization from Proposition 3.3.7 that an ω-regular relation $R \subseteq (\Sigma^\omega)^k$ is ω-recognizable if and only if the equivalence relation $\approx_j^R$ has finite index for all $j \in [1, k-1]$. Since $\approx_j^R$ has infinite index if and only if its complement $\not\approx_j^R$ has an infinite clique, it remains to decide the infinite clique problem for the co-equivalence relation $\not\approx_j^R$. Recall that $\not\approx_j^R \subseteq \Sigma^\omega \times \Sigma^\omega$ is defined such that $u \not\approx_j^R v$ if and only if

$$\exists \boldsymbol{w} \in (\Sigma^\omega)^{k-1} \colon (u \odot_j \boldsymbol{w} \in R \wedge v \odot_j \boldsymbol{w} \notin R) \vee (u \odot_j \boldsymbol{w} \notin R \wedge v \odot_j \boldsymbol{w} \in R).$$

If R is given by a DPA (resp. NBA), we can construct an NBA for $\not\approx_j^R$ in logspace (resp. polynomial space) using Propositions 2.3.10 and 2.3.30. Note that in case of a DPA, before we perform the projection using Proposition 2.3.30, we convert the DPA into an NBA using Proposition 2.3.9. Thus, to conclude the proof of Theorem 6.2.1, it suffices to show the following theorem.

Theorem 6.2.2. *The infinite clique problem over ω-regular co-equivalence relations $\overline{E} \subseteq (\Sigma^\omega)^{2d}$ is* NL*-complete.*

Here, it is important that $\overline{E}$ is the complement of an equivalence relation E. Decidability of the infinite clique problem for general ω-regular relations is a longstanding open problem [Kus10]. However, Löding and Spinrath [LS19b] showed that under the

assumption that the relation is a co-equivalence relation the problem is decidable in double exponential time. The rest of this section is devoted to proving Theorem 6.2.2. Note that the lower bound again follows from the finite-word case (Corollary 4.2.2) using an infinite padding. For the upper bound we use a similar approach as in [LS19b].

Recall that to solve the infinite clique problem for regular relations, we constructed a Büchi automaton that accepts encodings of infinite cliques. Over infinite words, however, it is not clear how such an encoding would look like since it would have to encode an infinite sequence of infinite words. A strong indicator that this is indeed difficult is that there are ω-regular relations which have infinite cliques but no *ω-regular* infinite clique. One such example is the complement of the *equal ends* equivalence relation $\sim_{\mathsf{e}} \subseteq \Sigma^\omega \times \Sigma^\omega$ where $u \sim_{\mathsf{e}} v$ if and only if there exist $u_0, v_0 \in \Sigma^*$ with $|u_0| = |v_0|$ and $w \in \Sigma^\omega$ such that $u = u_0 w$ and $v = v_0 w$. Kuske and Lohrey [KL08] observed that although $\not\sim_{\mathsf{e}}$ has infinite cliques, it does not have any ω-regular infinite cliques.

Instead of checking whether $\overline{E}$ has an infinite clique, we follow the approach by Löding and Spinrath [LS19b] and equivalently decide whether $\overline{E}$ has *unbounded cliques*: For every $n \geq 1$ there exists a clique $(w_1, \ldots, w_n)$ of size n in $\overline{E}$, i.e. $(w_i, w_j) \in \overline{E}$ for all $1 \leq i < j \leq n$. The important observation made in [LS19b] is that it suffices to search for unbounded cliques consisting of ultimately periodic words. Since an ultimately periodic word uv^ω can be encoded by the finite word $u\#v$, this allows us to reduce the infinite clique problem over ω-regular co-equivalence relations to a question on regular relations over finite words. Note that for general relations it is not necessarily the case that the existence of unbounded cliques implies the existence of an infinite clique. But for the complement of an equivalence relation both statements are equivalent to the equivalence relation having infinite index.

6.2.1 Reduction to Slenderness

For the rest of this section fix an NBA $\mathcal{A} = (Q, \Sigma^2, q_0, \Delta, F)$ for the complement $\overline{E} \subseteq \Sigma^\omega \times \Sigma^\omega$ of an equivalence relation E. Let us first recall the results from [LS19b] that will be used in our proof. We define the equivalence relation $E_\# \subseteq (\Sigma^*\{\#\}\Sigma^*)^2$ where $\# \notin \Sigma$ by

$$E_\# := \{(u\#v, x\#y) \mid (uv^\omega, xy^\omega) \in E, |u| = |x|, |v| = |y|\}.$$

To show that $E_\#$ is ω-regular, Löding and Spinrath [LS19b] use the notion of transition profiles. A *transition profile* $\tau = (\Rightarrow, \overset{F}{\Rightarrow})$ over $\mathcal{A}$ consists of two binary relations $\Rightarrow, \overset{F}{\Rightarrow} \in Q \times Q$. For each word $w \in (\Sigma^2)^*$ we define the transition profile $\tau(w)$ such that $p \Rightarrow q$ if and only if there exists a run $p \xrightarrow{w} q$ in $\mathcal{A}$ and $p \overset{F}{\Rightarrow} q$ if and only if there exists a run $p \xrightarrow{w} q$ in $\mathcal{A}$ visiting a final state. It is easy to see that $\tau(uv)$ for $u, v \in (\Sigma^2)^*$ is determined by $\tau(u)$ and $\tau(v)$ and therefore the set $\mathrm{TP}(\mathcal{A}) := \{\tau(w) \mid w \in (\Sigma^2)^*\}$ forms a finite monoid, called *transition monoid*, with the well-defined operation $\tau(u) \cdot \tau(v) := \tau(uv)$ and neutral element $\tau(\varepsilon)$. Note that $|\mathrm{TP}(\mathcal{A})|$ may be exponential in the size of $\mathcal{A}$.

Lemma 6.2.3. *The relation $E_\#$ is regular.*

Proof. The NFA $\mathcal{A}_\#$ for $E_\#$ checks whether on an input $u\#v \otimes x\#y$ with $|u| = |x|$ and $|v| = |y|$ there is a state q in $\mathcal{A}$ such that $q_0 \xrightarrow{u\otimes x} q$ and $q \xrightarrow{v\otimes y} q$ where the cycle visits a final state. Such a state exists if and only if $\mathcal{A}$ accepts the infinite word $uv^\omega \otimes xy^\omega$. Thus, since $\mathcal{A}$ accepts the complement of E, $\mathcal{A}_\#$ should accept if and only if such a state does not exist. To this end, $\mathcal{A}_\#$ first computes and stores the transition profile $\tau(u \otimes x)$ and then computes and stores the transition profile $\tau(v \otimes y)$ and accepts if there is no $q \in Q$ such that $q_0 \Rightarrow q$ in $\tau(u \otimes x)$ and $q \overset{F}{\Rightarrow} q$ in $\tau(v \otimes y)$. To compute the transition profiles, $\mathcal{A}_\#$ uses the monoid structure of $\mathrm{TP}(\mathcal{A})$, i.e. it starts with $\tau(\varepsilon)$ and if it reads $a \in \Sigma^2$ in a state with $\tau(w)$ stored, it switches to the state that stores $\tau(wa)$. □

Note that the above construction is effective, but the resulting NFA for $E_\#$ may be exponentially larger than $\mathcal{A}$ since it stores elements of $\mathrm{TP}(\mathcal{A})$ in its states. Using the techniques from [CCG06], one can show that $E_\#$ has a regular set of representatives.

Lemma 6.2.4. *The equivalence relation $E_\#$ has a regular complete set of representatives $L_\#(E)$. Moreover, $L_\#(E)$ can be written as*

$$L_\#(E) = \bigcup_{(i,j)\in I} P_i\{\#\}S_j$$

where $I \subseteq \mathbb{N}^2$ is a finite index set and $P_i, S_j \subseteq \Sigma^*$ are non-empty regular languages for all $(i,j) \in I$.

Proof. Let $<_{\mathrm{lex}}$ be some lexicographic order on words from $\Sigma^*\{\#\}\Sigma^*$. Clearly, $<_{\mathrm{lex}}$ is a regular relation. Then the language

$$L_\#(E) := \{w \in \Sigma^*\{\#\}\Sigma^* \mid \neg\exists w' \in \Sigma^*\{\#\}\Sigma^* \colon (w,w') \in E_\# \wedge w' <_{\mathrm{lex}} w\}$$

of lexicographically smallest representatives is regular and an NFA $\mathcal{B}_\#$ for it can be constructed from $\mathcal{A}_\#$ using Propositions 2.3.4 and 2.3.27. Note that $\mathcal{B}_\#$ may be exponentially larger than $\mathcal{A}_\#$.

For the "moreover" part let $\mathcal{B}_\# = (Q', \Sigma \cup \{\#\}, \Delta', q_0', F')$, then

$$L_\#(E) = \bigcup_{(p',\#,q')\in\Delta'} L_{q_0',p'}\{\#\}L_{p',F'}$$

where

$$L_{q_0',p'} := \{w \in \Sigma^* \mid q_0' \xrightarrow{w} p'\} \text{ and}$$
$$L_{p',F'} := \{w \in \Sigma^* \mid \exists f' \in F' \colon p' \xrightarrow{w} f'\}$$

are clearly regular languages. □

Let us fix the decomposition of $L_\#$ from Lemma 6.2.4. The next step in [LS19b] is to reduce the infinite clique problem for $\overline{E}$ to checking a property for all P_i, S_j. A language $L \subseteq \Sigma^*$ is *slender* if there exists a $k \in \mathbb{N}$ such that for all $\ell \in \mathbb{N}$ it holds that $|L \cap \Sigma^\ell| \leq k$.

Lemma 6.2.5. *The co-equivalence relation $\overline{E}$ has unbounded cliques if and only if there exists $(i,j) \in I$ such that P_i or S_j is not slender.*

Proof. The proof proceeds in three steps. We first show that $\overline{E}$ has unbounded cliques if and only if the relation

$$\overline{E}_{\#} := \{(u\#v, x\#y) \mid (uv^\omega, xy^\omega) \in \overline{E}, |u| = |x|, |v| = |y|\}$$

has unbounded cliques. The "if" direction is clear. For the converse we first observe that since E is an ω-regular equivalence relation that has by assumption infinite index, Lemma 2.3.31 implies that there are infinitely many ultimately periodic words $u_i v_i^\omega$ for $i \geq 1$ such that $(u_i v_i^\omega, u_j v_j^\omega) \in \overline{E}$ for all $1 \leq i < j$. We show that for every $n \geq 1$ the words $u_1\#v_1, \ldots, u_n\#v_n$ can be rewritten such that they form a clique in $\overline{E}_{\#}$ of size n. W.l.o.g. assume that $|u_1| \geq |u_i|$ for all $i \in [1,n]$. Write $v_i = \hat{v}_i\tilde{v}_i$ such that $|\hat{v}_i| = (|u_1| - |u_i|) \mod |v_i|$. Then define $u_i' := u_i v_i^{p_i}\hat{v}_i$ where $p_i := \lfloor \frac{|u_1|-|u_i|}{|v_i|} \rfloor$ for all $i \in [1,n]$. Now, we have that $|u_i'| = |u_1'|$ for all $i \in [1,n]$. To ensure that that the periods have the same length, let $\ell := \operatorname{lcm}(|v_1|, \ldots, |v_n|)$ and $\ell_i := \frac{\ell}{|v_i|}$ for all $i \in [1,n]$. Then we set $v_i' := (\tilde{v}_i\hat{v}_i)^{\ell_i}$ for all $i \in [1,n]$. It is easy to verify that now $|v_i'| = \ell$ and $u_i'(v_i')^\omega = u_i v_i^\omega$ for all $i \in [1,n]$ as desired.

As a second step we show that $\overline{E}_{\#}$ has unbounded cliques if and only if for all $k \in \mathbb{N}$ there exist $n, m \in \mathbb{N}$ such that $|L_{\#}(E) \cap \Sigma^n\{\#\}\Sigma^m| > k$. If $\overline{E}_{\#}$ has unbounded cliques, then for any $k \in \mathbb{N}$ there exists a clique $(u_1\#v_1, \ldots, u_{k+1}\#v_{k+1})$ in $\overline{E}_{\#}$ of size $k+1$. By definition of $\overline{E}_{\#}$, there are $n, m \in \mathbb{N}$ such that $|u_i| = n$ and $|v_i| = m$ for all $i \in [1, k+1]$ and $(u_i\#v_i, u_j\#v_j) \notin E_{\#}$ for all $1 \leq i < j \leq k+1$. Thus, $L_{\#}(E)$ contains a representative for each of the $k+1$ pairwise distinct $E_{\#}$-equivalence classes with the desired properties. The converse can be shown analogously.

As a third step it remains to show that for all $k \in \mathbb{N}$ there exist $n, m \in \mathbb{N}$ such that $|L_{\#}(E) \cap \Sigma^n\{\#\}\Sigma^m| > k$ if and only if in the decomposition $L_{\#}(E) = \bigcup_{(i,j)\in I} P_i\{\#\}S_j$ there is $(i,j) \in I$ such that P_i or S_j is not slender. Since I is finite, we have that for all $k \in \mathbb{N}$ there exist $n, m \in \mathbb{N}$ such that $|L_{\#}(E) \cap \Sigma^n\{\#\}\Sigma^m| > k$ if and only if there exists $(i,j) \in I$ such that for all $k \in \mathbb{N}$ there exist $n, m \in \mathbb{N}$ such that $|P_i\{\#\}S_j \cap \Sigma^n\{\#\}\Sigma^m| > k$. It is not hard to see that the latter condition is satisfied for some $(i,j) \in I$ if and only if P_i or S_j is not slender. □

With Lemma 6.2.5 at hand, the approach by Löding and Spinrath [LS19b] for checking whether $\overline{E}$ has unbounded cliques is to construct automata for each of the languages P_i and S_j as shown in Lemma 6.2.4 and check whether they are slender. However, the construction uses an automaton for $L_{\#}(E)$, which might be double exponentially large since its size is exponential in the size of the automaton for $E_{\#}$, which in turn may be exponential in the size of $\mathcal{A}$ for $\overline{E}$ as shown in Lemma 6.2.3. Hence, this results in a double exponential-time algorithm given $\mathcal{A}$. We deviate from this approach and use the slenderness property of the languages P_i and S_j only to identify the shape of unbounded cliques in $\overline{E}$. In a second step, we search for patterns in $\mathcal{A}$ that witness the existence of unbounded cliques in $\overline{E}$. The existence of these patterns can be checked in nondeterministic logspace given $\mathcal{A}$.

6.2.2 Special Unbounded Cliques

It is well-known (see e.g. [Pin22]) that slenderness of a regular language can be characterized by the absence of two distinct cycles in the automaton. We show that we can even restrict to pairs of cycles whose lengths are the same as the length of the connecting word. For convenience, in the following we drop the set braces to denote a singleton language $\{w\}$ and just write w. For instance, this allows us to write uv^*wx^*y for words $u, v, w, x, y \in \Sigma^*$ to denote the language $\{uv^n wx^m y \mid n, m \geq 0\}$.

Lemma 6.2.6. *A regular language $L \subseteq \Sigma^*$ is not slender if and only if there are words $u, v, w, x, y \in \Sigma^*$ with $|v| = |w| = |x| > 0$ and $v \neq w$ such that $uv^*wx^*y \subseteq L$.*

Proof. The "if" direction holds since for any $k \in \mathbb{N}$ we have that L contains the pairwise distinct words $uv^{i|x|}wx^{(k-i)|v|}y$ for $i \in [0, k]$ of the same length, which means that L is not slender.

For the "only if" direction consider a trimmed DFA for L, i.e. a DFA in which every state is reachable from the initial state and can reach a final state, and assume that L is not slender. We call two runs *distinct* if their sets of transitions are distinct. We first show that the DFA must contain two simple cycles that are distinct and connected. For the sake of contradiction, suppose that every run of the DFA traverses at most one simple cycle up to non-distinctness, i.e. simple cycles with different sets of transitions cannot be part of the same run. Let n be the number of states. Then any word of length greater than n iterates exactly one simple cycle up to non-distinctness, which means that it has the form $w_1 w_2^k w_3$ with $|w_1|, |w_2|, |w_3| \leq n$. This, however, implies that L is slender since there are only finitely many choices for w_1, w_2, w_3. So let $p \xrightarrow{v} p$ and $q \xrightarrow{x} q$ with $|v|, |x| > 0$ be two distinct simple cycles such that there is a run $p \xrightarrow{w} q$. Note that the sets of transitions of two distinct simple cycles are incomparable with respect to the subset relation. By replacing w with wx, we can assume that the runs $p \xrightarrow{v} p$ and $p \xrightarrow{w} q$ are distinct since the simple cycle $q \xrightarrow{x} q$ must contain a transition that is not part of $p \xrightarrow{v} p$. Then the language uv^*wx^*y is contained in L, where u and y are words read from the initial state to p and from q to some final state. We can ensure that $|w| \leq |v| = |x|$ by replacing v and x with $v^{k|x|}$ and $x^{k|v|}$, respectively, for sufficiently large $k \in \mathbb{N}$. Furthermore, we can ensure $|v| = |w| = |x|$ by extending the run $p \xrightarrow{w} q$ to $p \xrightarrow{wx_1} r$ for some prefix x_1 of x with $|wx_1| = |x|$, replacing the cycle $q \xrightarrow{x} q$ with $r \xrightarrow{x_2 x_1} r$ for $x = x_1 x_2$, and replacing y with $x_2 y$. Note that we also have that $v \neq w$ since the runs $p \xrightarrow{v} p$ and $p \xrightarrow{w} q$ are still distinct and in a DFA distinct runs starting in the same state cannot read the same word. □

The following lemma distinguishes two types of cliques: On the one hand, there are cliques whose infinite words differ in a finite prefix but have equal ends. On the other hand, there are cliques whose infinite words do not have equal ends. For example, consider the equality relation $=$ on infinite words over $\{a, b\}$. In the complement $\neq$ we can find the cliques $(a^i b^{n-i} a^\omega)_{0 \leq i \leq n}$ for all $n \in \mathbb{N}$ of infinite words that only differ in a finite prefix. On the other hand, if we consider the equal ends equivalence relation $\sim_{\mathsf{e}}$, then we observe that it does not suffice to look at the finite prefixes of infinite words

to determine whether they are in relation or not. In the complement $\not\sim_{\mathsf{e}}$ we can find the cliques $((a^i b^{n-i})^\omega)_{0\leq i\leq n}$ for all $n \in \mathbb{N}$ of ultimately periodic words that differ in the periodic part.

Lemma 6.2.7. *The co-equivalence relation $\overline{E}$ has unbounded cliques if and only if it contains cliques of the form*

(i) $(uv^i wx^{n-i} yz^\omega)_{0\leq i\leq n}$ *or*

(ii) $(z(uv^i wx^{n-i} y)^\omega)_{0\leq i\leq n}$

for all $n \in \mathbb{N}$, where $u, v, w, x, y, z \in \Sigma^*$ with $|v| = |w| = |x| > 0$ and $v \neq w$.

Proof. The "if" direction is clear. For the "only if" direction assume that $\overline{E}$ has unbounded cliques. Then by Lemma 6.2.5, there are non-empty regular languages $P, S \subseteq \Sigma^*$ with $P\{\#\}S \subseteq L_\#(E)$ such that P or S is not slender. Applying Lemma 6.2.6, this means that there are $u, v, w, x, y \in \Sigma^*$ with $|v| = |w| = |x| > 0$ and $v \neq w$ such that $uv^*wx^*y \subseteq P$ or $uv^*wx^*y \subseteq S$. If $uv^*wx^*y \subseteq P$, we pick a word $z \in S$ from the non-empty language S. Then $uv^*wx^*y\#z \subseteq L_\#(E)$. Since all words of the form $uv^i wx^j y$ for $i, j \geq 0$ are pairwise different, $\overline{E}$ contains the clique $(uv^i wx^{n-i} yz^\omega)_{0\leq i\leq n}$ for all $n \in \mathbb{N}$. Similarly, if $uv^*wx^*y \subseteq S$, we pick a word $z \in P$ and find the cliques $(z(uv^i wx^{n-i} y)^\omega)_{0\leq i\leq n}$ in $\overline{E}$. □

6.2.3 Patterns Witnessing Unbounded Cliques

To detect whether $\overline{E}$ contains cliques as in Lemma 6.2.7, we can search for certain patterns in its automaton $\mathcal{A}$. A *3-cycles pattern* consists of states $q_1, q_2, q_3, q_4, q_5 \in Q$ and words $u, v, w, x, y \in \Sigma^*$ with $|v| = |w| = |x| > 0$ and $v \neq w$ such that

$$q_1 \xrightarrow{\left[\begin{smallmatrix} u \\ u \end{smallmatrix}\right]} q_2, \quad q_2 \xrightarrow{\left[\begin{smallmatrix} v \\ v \end{smallmatrix}\right]} q_2, \quad q_2 \xrightarrow{\left[\begin{smallmatrix} w \\ v \end{smallmatrix}\right]} q_3, \quad q_3 \xrightarrow{\left[\begin{smallmatrix} x \\ v \end{smallmatrix}\right]} q_3,$$
$$q_3 \xrightarrow{\left[\begin{smallmatrix} x \\ w \end{smallmatrix}\right]} q_4, \quad q_4 \xrightarrow{\left[\begin{smallmatrix} x \\ x \end{smallmatrix}\right]} q_4, \quad q_4 \xrightarrow{\left[\begin{smallmatrix} y \\ y \end{smallmatrix}\right]} q_5.$$

We say that the above is a 3-cycles pattern from q_1 to q_5. The 3-cycles pattern is called *final* if one of the runs $q_1 \xrightarrow{\left[\begin{smallmatrix} u \\ u \end{smallmatrix}\right]} q_2$, $q_2 \xrightarrow{\left[\begin{smallmatrix} w \\ v \end{smallmatrix}\right]} q_3$, $q_3 \xrightarrow{\left[\begin{smallmatrix} x \\ w \end{smallmatrix}\right]} q_4$, $q_4 \xrightarrow{\left[\begin{smallmatrix} y \\ y \end{smallmatrix}\right]} q_5$ visits a final state. If there exists a (final) 3-cycles pattern from q_1 to q_5, we write $q_1 \xrightarrow{\mathsf{3CP}} q_5$ and $q_1 \xrightarrow{\mathsf{3CP}}_F q_5$, respectively.

Clearly, $p \xrightarrow{\mathsf{3CP}} q$ implies that $\mathcal{A}$ contains for any $n \in \mathbb{N}$ a run from p to q reading $uv^i wx^{n-i} y \otimes uv^j wx^{n-j} y$ for all $0 \leq i < j \leq n$, where $u, v, w, x, y \in \Sigma^*$ with $|v| = |w| = |x| > 0$ and $v \neq w$. That is, if $p = q_0$ and $\mathcal{A}$ accepts $z^\omega \otimes z^\omega$ from q for some $z \in \Sigma^*$, then $\overline{E}$ contains unbounded cliques of the form (i) in Lemma 6.2.7. To prove that the converse also holds, we use the fact that $\mathrm{TP}(\mathcal{A})$ is a finite monoid. An element s in a monoid M is *idempotent* if $s^2 = s$. It is well-known that every finite monoid M has an *idempotent exponent*, i.e. a number $e \geq 1$ such that s^e is idempotent for all $s \in M$. Recall that by definition of $\mathrm{TP}(\mathcal{A})$, we have that $\tau\left(\left[\begin{smallmatrix} u^n \\ v^n \end{smallmatrix}\right]\right) = \tau\left(\left[\begin{smallmatrix} u \\ v \end{smallmatrix}\right]\right)^n$ for all $n \geq 0$ and transition profile $\tau\left(\left[\begin{smallmatrix} u \\ v \end{smallmatrix}\right]\right) \in \mathrm{TP}(\mathcal{A})$. In the following we just write $\tau\left[\begin{smallmatrix} u \\ v \end{smallmatrix}\right]$ instead of $\tau\left(\left[\begin{smallmatrix} u \\ v \end{smallmatrix}\right]\right)$.

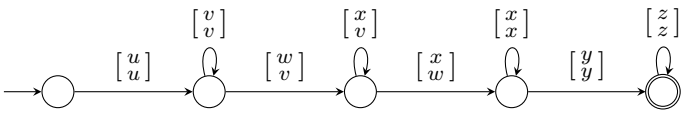


Figure 6.2: Illustration of the pattern for detecting cliques of the form (i) in Lemma 6.2.7.

Lemma 6.2.8. *Let $e \geq 1$ be the idempotent exponent of $\mathrm{TP}(\mathcal{A})$ and $n := |Q|$. If for words $u, v, w, x, y, z \in \Sigma^*$ with $|v| = |w| = |x| > 0$ and $v \neq w$ and states $q_1, q_5 \in Q$ there exists a run ρ in $\mathcal{A}$ from q_1 to q_5 reading*

$$\left[\begin{smallmatrix} u \\ u \end{smallmatrix}\right]\left[\begin{smallmatrix} v^{en} \\ v^{en} \end{smallmatrix}\right]\left[\begin{smallmatrix} v^{e-1}w \\ v^{e} \end{smallmatrix}\right]\left[\begin{smallmatrix} x^{en} \\ v^{en} \end{smallmatrix}\right]\left[\begin{smallmatrix} x^{e} \\ v^{e-1}w \end{smallmatrix}\right]\left[\begin{smallmatrix} x^{en} \\ x^{en} \end{smallmatrix}\right]\left[\begin{smallmatrix} y \\ y \end{smallmatrix}\right], \tag{6.1}$$

then $q_1 \xrightarrow{\mathsf{3CP}} q_5$. Moreover, if ρ visits a final state, then $q_1 \xrightarrow{\mathsf{3CP}}_F q_5$.

Proof. We will use the fact that $\tau\left[\begin{smallmatrix} v^e \\ v^e \end{smallmatrix}\right]$, $\tau\left[\begin{smallmatrix} x^e \\ v^e \end{smallmatrix}\right]$, and $\tau\left[\begin{smallmatrix} x^e \\ x^e \end{smallmatrix}\right]$ are idempotent in $\mathrm{TP}(\mathcal{A})$. Let us replace v with v^e, w with $v^{e-1}w$, and x with x^e. Now, the transition profiles $\tau\left[\begin{smallmatrix} v \\ v \end{smallmatrix}\right]$, $\tau\left[\begin{smallmatrix} x \\ v \end{smallmatrix}\right]$, and $\tau\left[\begin{smallmatrix} x \\ x \end{smallmatrix}\right]$ are idempotent in $\mathrm{TP}(\mathcal{A})$ and ρ becomes a run reading

$$\left[\begin{smallmatrix} u \\ u \end{smallmatrix}\right]\left[\begin{smallmatrix} v^n \\ v^n \end{smallmatrix}\right]\left[\begin{smallmatrix} w \\ v \end{smallmatrix}\right]\left[\begin{smallmatrix} x^n \\ v^n \end{smallmatrix}\right]\left[\begin{smallmatrix} x \\ w \end{smallmatrix}\right]\left[\begin{smallmatrix} x^n \\ x^n \end{smallmatrix}\right]\left[\begin{smallmatrix} y \\ y \end{smallmatrix}\right].$$

The sequence of $n+1$ states visited before and after reading each of the n factors $\left[\begin{smallmatrix} v \\ v \end{smallmatrix}\right]$ must contain a repeated state and similarly for the factors $\left[\begin{smallmatrix} x \\ v \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} x \\ x \end{smallmatrix}\right]$. Therefore, we find intermediate states $q_2, q_3, q_4 \in Q$ so that ρ has the form

$$\begin{aligned} \rho_1 &\colon q_1 \xrightarrow{\left[\begin{smallmatrix} uv^{i_1} \\ uv^{i_1} \end{smallmatrix}\right]} q_2, & \sigma_2 &\colon q_2 \xrightarrow{\left[\begin{smallmatrix} v^{i_2} \\ v^{i_2} \end{smallmatrix}\right]} q_2, \\ \rho_2 &\colon q_2 \xrightarrow{\left[\begin{smallmatrix} v^{i_3} \\ v^{i_3} \end{smallmatrix}\right]\left[\begin{smallmatrix} w \\ v \end{smallmatrix}\right]\left[\begin{smallmatrix} x^{j_1} \\ v^{j_1} \end{smallmatrix}\right]} q_3, & \sigma_3 &\colon q_3 \xrightarrow{\left[\begin{smallmatrix} x^{j_2} \\ v^{j_2} \end{smallmatrix}\right]} q_3 \\ \rho_3 &\colon q_3 \xrightarrow{\left[\begin{smallmatrix} x^{j_3} \\ v^{j_3} \end{smallmatrix}\right]\left[\begin{smallmatrix} x \\ w \end{smallmatrix}\right]\left[\begin{smallmatrix} x^{k_1} \\ x^{k_1} \end{smallmatrix}\right]} q_4, & \sigma_4 &\colon q_4 \xrightarrow{\left[\begin{smallmatrix} x^{k_2} \\ x^{k_2} \end{smallmatrix}\right]} q_4, \\ \rho_4 &\colon q_4 \xrightarrow{\left[\begin{smallmatrix} x^{k_3}y \\ x^{k_3}y \end{smallmatrix}\right]} q_5 \end{aligned} \tag{6.2}$$

for some numbers $i_1, j_1, k_1, i_3, j_3, k_3 \geq 0$ and $i_2, j_2, k_2 \geq 1$. Since $\tau\left[\begin{smallmatrix} v \\ v \end{smallmatrix}\right]$, $\tau\left[\begin{smallmatrix} x \\ v \end{smallmatrix}\right]$, and $\tau\left[\begin{smallmatrix} x \\ x \end{smallmatrix}\right]$ are idempotent in $\mathrm{TP}(\mathcal{A})$, there exist runs $\tilde{\rho}_1, \tilde{\sigma}_2, \tilde{\rho}_2, \tilde{\sigma}_3, \tilde{\rho}_3, \tilde{\sigma}_4, \tilde{\rho}_4$ as in Equation (6.2) such that $i_\ell = j_\ell = k_\ell = 1$ for all $\ell \in [1, 3]$. Then the five words uv, v^3, vwx, x^3, xy form the required 3-cycles pattern from q_1 to q_5.

For the "moreover" part assume that ρ visits a final state. We can ensure that the final state occurs in one of the subruns ρ_i in Equation (6.2): If the final state occurs in one of the cycles σ_i, then we can append the cycle σ_i to ρ_i. Using the relation $\overset{F}{\Rightarrow}$ of the respective transition profile, we can then choose the run $\tilde{\rho}_i$ such that it also visits a final state and therefore $q_1 \xrightarrow{\mathsf{3CP}}_F q_5$. □

The following lemma shows that a 3-cycles pattern can be used to detect unbounded cliques in $\overline{E}$ that are of the form (i) in Lemma 6.2.7. The pattern is illustrated in Figure 6.2.

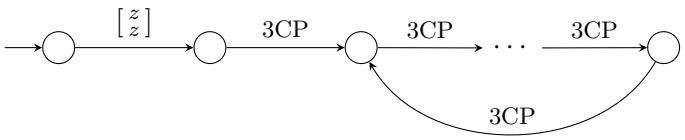


Figure 6.3: Illustration of the pattern for detecting cliques of the form (ii) in Lemma 6.2.7.

Lemma 6.2.9. *The co-equivalence relation $\overline{E}$ contains cliques $(uv^iwx^{n-i}yz^\omega)_{0\leq i\leq n}$ for all $n \in \mathbb{N}$, where $u, v, w, x, y, z \in \Sigma^*$ with $|v| = |w| = |x| > 0$ and $v \neq w$, if and only if $q_0 \xrightarrow{\mathsf{3CP}} q$ for some state $q \in Q$ such that $\left[\begin{smallmatrix} z'^\omega \\ z'^\omega \end{smallmatrix}\right]$ is accepted from q for some word $z' \in \Sigma^*$.*

Proof. We first observe that $\overline{E}$ contains cliques $(uv^iwx^{n-i}yz^\omega)_{0\leq i\leq n}$ as in the lemma if and only if

$$\left[\begin{smallmatrix} u \\ u \end{smallmatrix}\right]\left[\begin{smallmatrix} v \\ v \end{smallmatrix}\right]^*\left[\begin{smallmatrix} w \\ v \end{smallmatrix}\right]\left[\begin{smallmatrix} x \\ v \end{smallmatrix}\right]^*\left[\begin{smallmatrix} x \\ w \end{smallmatrix}\right]\left[\begin{smallmatrix} x \\ x \end{smallmatrix}\right]^*\left[\begin{smallmatrix} y \\ y \end{smallmatrix}\right]\left[\begin{smallmatrix} z \\ z \end{smallmatrix}\right]^\omega \subseteq \overline{E} \tag{6.3}$$

for some $u, v, w, x, y, z \in \Sigma^*$ with $|v| = |w| = |x| > 0$ and $v \neq w$. Then the "if" direction of the lemma follows directly. For the "only if" direction assume that Equation (6.3) holds. Let $n := |Q|$ and $e \geq 1$ be the idempotent exponent of $\mathrm{TP}(\mathcal{A})$. By Equation (6.3), there is a run of $\mathcal{A}$ on

$$\left[\begin{smallmatrix} u \\ u \end{smallmatrix}\right]\left[\begin{smallmatrix} v \\ v \end{smallmatrix}\right]^{en+e-1}\left[\begin{smallmatrix} w \\ v \end{smallmatrix}\right]\left[\begin{smallmatrix} x \\ v \end{smallmatrix}\right]^{en+e-1}\left[\begin{smallmatrix} x \\ w \end{smallmatrix}\right]\left[\begin{smallmatrix} x \\ x \end{smallmatrix}\right]^{en}\left[\begin{smallmatrix} y \\ y \end{smallmatrix}\right] = \left[\begin{smallmatrix} u \\ u \end{smallmatrix}\right]\left[\begin{smallmatrix} v^{en} \\ v^{en} \end{smallmatrix}\right]\left[\begin{smallmatrix} v^{e-1}w \\ v^e \end{smallmatrix}\right]\left[\begin{smallmatrix} x^{en} \\ v^{en} \end{smallmatrix}\right]\left[\begin{smallmatrix} x^e \\ v^{e-1}w \end{smallmatrix}\right]\left[\begin{smallmatrix} x^{en} \\ x^{en} \end{smallmatrix}\right]\left[\begin{smallmatrix} y \\ y \end{smallmatrix}\right]$$

from q_0 to some state $q \in Q$ such that $\left[\begin{smallmatrix} z^\omega \\ z^\omega \end{smallmatrix}\right]$ is accepted from q. Applying Lemma 6.2.8 yields the desired 3-cycles pattern. □

We now consider a slightly more complicated pattern that occurs in $\mathcal{A}$ if $\overline{E}$ has cliques of the form (ii) in Lemma 6.2.7. Essentially, we can always find a lasso of 3-cycles patterns. For an illustration see Figure 6.3.

Lemma 6.2.10. *If $\overline{E}$ contains cliques $(z(uv^iwx^{n-i}y)^\omega)_{0\leq i\leq n}$ for all $n \in \mathbb{N}$, where $u, v, w, x, y, z \in \Sigma^*$ with $|v| = |w| = |x| > 0$ and $v \neq w$, then there are states $q_1, \ldots, q_\ell \in Q$ such that*

- $q_0 \xrightarrow{\left[\begin{smallmatrix} z \\ z \end{smallmatrix}\right]} q_1$,
- $q_1 \xrightarrow{\mathsf{3CP}} q_2 \xrightarrow{\mathsf{3CP}} \cdots \xrightarrow{\mathsf{3CP}} q_{\ell-1} \xrightarrow{\mathsf{3CP}_F} q_\ell$, *and*
- $q_\ell = q_k$ *for some* $1 \leq k < \ell$.

Proof. Suppose that $\overline{E}$ contains cliques $(z(uv^iwx^{n-i}y)^\omega)_{0\leq i\leq n}$ with $|v| = |w| = |x| > 0$ and $v \neq w$. Let t be the word from Equation (6.1). Since $\left[\begin{smallmatrix} z \\ z \end{smallmatrix}\right]t^\omega$ is accepted by $\mathcal{A}$, it has an accepting run of the following form:

$$q_0 \xrightarrow{\left[\begin{smallmatrix} z \\ z \end{smallmatrix}\right]} q_1 \xrightarrow{t} q_2 \xrightarrow{t} q_3 \xrightarrow{t} \ldots$$

Let $m \geq 1$ such that $\{q_1, \ldots, q_m\} = \{q_i \mid i \geq 1\}$, i.e. all states q_i have been visited at least once after reaching q_m. Since the run visits some final state infinitely often, there exists $\ell > m$ such that the run $q_{\ell-1} \xrightarrow{t} q_\ell$ visits a final state. Furthermore, there exists $1 \leq k \leq m$ such that $q_k = q_\ell$. Applying Lemma 6.2.8 yields $q_i \xrightarrow{\mathsf{3CP}} q_{i+1}$ for all $1 \leq i < \ell$ and in particular $q_{\ell-1} \xrightarrow{\mathsf{3CP}}_F q_\ell$. □

The following lemma shows that the pattern from Lemma 6.2.10 is also sufficient to detect the existence of unbounded cliques in $\overline{E}$.

Lemma 6.2.11. *If there are states $q_1, \ldots, q_\ell \in Q$ such that*

- $q_0 \xrightarrow{\left[\begin{smallmatrix} z \\ z \end{smallmatrix}\right]} q_1$,
- $q_1 \xrightarrow{\mathsf{3CP}} q_2 \xrightarrow{\mathsf{3CP}} \cdots \xrightarrow{\mathsf{3CP}} q_{\ell-1} \xrightarrow{\mathsf{3CP}}_F q_\ell$, *and*
- $q_\ell = q_k$ *for some* $1 \leq k < \ell$,

then $\overline{E}$ contains unbounded cliques.

Proof. Let $u_j, v_j, w_j, x_j, y_j \in \Sigma^*$ be the words of the 3-cycles pattern from q_j to q_{j+1} for $1 \leq j < \ell$. Define the word $t_j(i,n) := u_j v_j^i w_j x_j^{n-i} y_j$ for all $0 \leq i \leq n$ and $1 \leq j < \ell$. Then

$$\big(z t_1(i,n) \ldots t_{k-1}(i,n) \big(t_k(i,n) \ldots t_\ell(i,n)\big)^\omega\big)_{0 \leq i \leq n}$$

forms a clique in $\overline{E}$ for each $n \in \mathbb{N}$. □

We are now ready to prove the upper bound in Theorem 6.2.2. Since by Proposition 3.1.4, the case $\overline{E} \subseteq (\Sigma^\omega)^{2d}$ for $d > 1$ can be reduced to the case $d = 1$, we can assume that $\overline{E} \subseteq \Sigma^\omega \times \Sigma^\omega$. As above, let $\mathcal{A} = (Q, \Sigma^2, \Delta, q_0, F)$ be an NBA for $\overline{E}$. We already observed that since $\overline{E}$ is a co-equivalence relation, it has an infinite clique if and only if it has unbounded cliques. As shown above, to decide whether $\overline{E}$ has unbounded cliques, it suffices to check whether $\mathcal{A}$ contains the pattern in Lemma 6.2.9 or the pattern in Lemmas 6.2.10 and 6.2.11.

Let us first observe that with the same technique as in Proposition 2.3.12 we can check in nondeterministic logspace whether, given states $p, q \in Q$, there exists a word $z \in \Sigma^*$ such that $p \xrightarrow{\left[\begin{smallmatrix} z \\ z \end{smallmatrix}\right]} q$ and whether, given $q \in Q$, there exists a word $z \in \Sigma^*$ such that $\left[\begin{smallmatrix} z^\omega \\ z^\omega \end{smallmatrix}\right]$ is accepted by $\mathcal{A}$ from q. Furthermore, given two states $q_1, q_5 \in Q$, we can check in nondeterministic logspace whether $q_1 \xrightarrow{\mathsf{3CP}} q_5$ as follows: Construct in logspace an NFA $\mathcal{A}_{q_1,q_5}$ which reads a convolution $u \otimes v \otimes w \otimes x \otimes y$ with $|v| = |w| = |x|$ and $v \neq w$. It initially guesses and stores the states $q_2, q_3, q_4 \in Q$. Then it simulates 7 copies of $\mathcal{A}$ in parallel to check the existence of the runs as in the definition of a 3-cycles pattern. Now, $q_1 \xrightarrow{\mathsf{3CP}} q_5$ if and only if $L(\mathcal{A}_{q_1,q_5})$ is non-empty, which by Proposition 2.3.5 can be checked in nondeterministic logspace. Similarly, we can test $q_1 \xrightarrow{\mathsf{3CP}}_F q_5$. This allows us to detect the pattern in Lemma 6.2.9 in nondeterministic logspace. For the pattern in Lemmas 6.2.10 and 6.2.11 we remark that it suffices to only guess q_k at the beginning

and to check that q_k can be reached via a path of 3-cycles patterns followed by a cycle of 3-cycles patterns such that the last 3-cycles pattern visits a final state.

6.2.4 Monadic Decomposability Over ω-Automatic Structures

In Proposition 3.3.10 we observed that there is a tight connection between monadic decomposability and recognizability. We showed that a formula $\varphi(x_1, \ldots, x_k)$ over some structure $\mathfrak{A}$ that satisfies the assumption of Proposition 3.3.10 is monadically decomposable in the quantifier-free fragment of $\mathfrak{A}$ if and only if the equivalence relation $\approx_j^{[\![\varphi]\!]}$ has finite index for all $j \in [1, k-1]$. Moreover, Proposition 3.3.7 shows that if $\mathfrak{A}$ is ω-automatic with presentation $\mathfrak{p}$, then $[\![\varphi]\!]_{\mathfrak{p}}$ is ω-recognizable if and only if $\approx_j^{[\![\varphi]\!]_{\mathfrak{p}}}$ has finite index for all $j \in [1, k-1]$. Thus, if $\mathfrak{A}$ is ω-automatic and satisfies the condition of Proposition 3.3.10, then φ is monadically decomposable in the quantifier-free fragment of $\mathfrak{A}$ if and only if $[\![\varphi]\!]_{\mathfrak{p}}$ is ω-recognizable. Therefore, we can apply Theorem 6.2.1 to decide monadic decomposability of φ in the quantifier-free fragment of $\mathfrak{A}$ in nondeterministic logspace if a DPA for $[\![\varphi]\!]_{\mathfrak{p}}$ is given. Note that if φ is quantifier-free, then a DPA for $[\![\varphi]\!]_{\mathfrak{p}}$ can be constructed in polynomial space using Proposition 2.3.10 since we assume that $\mathfrak{p}$ is fixed.

Corollary 6.2.12. *Let $\mathfrak{A}$ be a fixed ω-automatic structure that satisfies the condition of Proposition 3.3.10. Then one can decide in polynomial space whether a quantifier-free formula over $\mathfrak{A}$ is monadically decomposable in the quantifier-free fragment of $\mathfrak{A}$.*

In Examples 2.4.9 and 3.3.11 we saw that LRA and LIRA are ω-automatic structures that satisfy the condition of Proposition 3.3.10. Hence, Corollary 6.2.12 implies that monadic decomposability in the quantifier-free fragments of LRA and LIRA is decidable in polynomial space. Recall that we already showed in Corollary 5.6.1 that the precise complexity is coNP. However, Corollary 6.2.12 provides complexity upper bounds for a more general class of structures.

6.3 Recognizability of Deterministic Rational Relations

Let us continue with membership problems over finite words. In Corollary 4.4.9 we already observed the precise complexity of recognizability for regular relations. In this section we consider the problem of recognizability in the strictly more general class of deterministic rational relations. This problem was shown to be decidable by Carton, Choffrut, and Grigorieff [CCG06] for relations of arbitrary arity. In fact, the algorithm from [CCG06] runs in elementary time if the arity is fixed. Carton, Choffrut, and Grigorieff [CCG06] observed that for binary relations the problem can be reduced to checking whether a *deterministic pushdown automaton* (DPDA), i.e. a deterministic automaton equipped with a stack (we refer to [Val75] for a formal definition), accepts a regular language. For a binary relation $R \subseteq \Sigma^* \times \Sigma^*$ let $L_R := \{\mathrm{rev}(u)\#v \mid (u, v) \in R\}$ where $\mathrm{rev}(u)$ denotes the reversal of u.

Proposition 6.3.1. *Let $R \subseteq \Sigma^* \times \Sigma^*$ be a binary rational relation. Then R is recognizable if and only if L_R is regular. Moreover, if R is deterministic rational, then, given a deterministic 2-tape automaton for R, one can construct in logspace a DPDA accepting L_R.*

Here, the DPDA for L_R on input $\mathrm{rev}(u)\#v$ first pushes u onto the stack and then simulates the deterministic 2-tape automaton on the stack content or the input depending on which tape is currently read from. Thus, to check whether a binary deterministic rational relation R is recognizable, we can construct a DPDA for L_R and check whether the DPDA accepts a regular language. Checking whether a DPDA accepts a regular language was shown to be decidable in double exponential time by Valiant [Val75] (improving on a triple exponential-time upper bound by Stearns [Ste67]). Hence, in total this yields a double exponential-time algorithm for recognizability of binary deterministic rational relations. We revisit the proof from [CCG06] to obtain the following improved complexity bounds. In particular, we show that for binary relations the problem is solvable in polynomial time.

Theorem 6.3.2. *Given a deterministic rational relation $R \subseteq (\Sigma^*)^k$, one can decide whether R is recognizable (i) in* P *if $k = 2$, (ii) in* coREXP *if $k > 2$ is fixed, and (iii) in* coNEXP *if k is part of the input.*

Recall that for regular relations (Corollary 4.4.9) and ω-regular relations (Theorem 6.2.1) R the strategy to decide recognizability was to use Proposition 3.3.7 and reduce the problem to the infinite clique problem for the co-equivalence relations $\not\approx_j^R$. However, since the class **DRat** does not enjoy the same closure properties as the classes **Reg** and ω-**Reg**, the relations $\not\approx_j^R$ might not be deterministic rational even if R is deterministic rational. If restricted to binary deterministic rational relations R, it can at least be observed that $\not\approx_1^R$ is a rational relation. However, it is not clear whether the infinite clique problem for rational relations, even when restricted to binary co-equivalence relations, is decidable at all. For this reason, we use a more direct approach by extracting a pattern directly in the deterministic multitape automaton for R whose existence is a sufficient and necessary condition for $\approx_j^R$ having infinite index.

Note that in order to prove Theorem 6.3.2, we can ignore the endmarker $\dashv$ in the definition of **DRat** in Section 2.3.5 since a relation R is recognizable if and only if $\{\boldsymbol{w}(\dashv, \ldots, \dashv) \mid \boldsymbol{w} \in R\}$ is recognizable. Hence, for the rest of this section let $R \subseteq (\Sigma^*)^k$ with $R = R(\mathcal{A})$ for some deterministic k-tape automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ with n states. Furthermore, we assume that all states of $\mathcal{A}$ are reachable from q_0. We also write R_q for the relation accepted from state $q \in Q$, i.e. $R_q := R(\mathcal{A}_q)$ where $\mathcal{A}_q := (Q, \Sigma, \delta, q, F)$.

6.3.1 Characterizing Non-recognizability

As mentioned above, by Proposition 3.3.7, it suffices to decide whether $\approx_j^R$ has finite index for all $j \in [1, k-1]$. To simplify the notation, we will in the following focus on the equivalence relation $\approx_1^R$ and drop the superscript. This is without loss of generality

since by permuting the components of R, we can reduce the test for finite index of $\approx_j$ for each $j \in [1, k-1]$ to the test for finite index of $\approx_1$.

Our first step is to provide several equivalent characterizations of when $\approx_1$ has *infinite* index. Those will be used to identify patterns in $\mathcal{A}$ that witness *non*-recognizability of R. The characterizations will be deduced from the proof of Lemma 3.5 in [CCG06], which states that if $\approx_1$ has finite index, then any word is $\approx_1$-equivalent to a word whose length is exponentially bounded in n.

Let us first introduce some definitions from [CCG06]. A non-empty word $v_1 \in \Sigma^*$ is *null-transparent* if for all $s, t \in Q_1$ we have $s \xrightarrow{(v_1, \boldsymbol{\varepsilon})} t$ implies $t \xrightarrow{(v_1, \boldsymbol{\varepsilon})} t$. In other words, the transition profile $\tau(v_1, \boldsymbol{\varepsilon})$ is idempotent in the transition monoid $\mathrm{TP}(\mathcal{A})$ if restricted to states from Q_1. Recall from Section 6.2 that since $\mathrm{TP}(\mathcal{A})$ is a finite monoid, it has an idempotent exponent $e \geq 1$. Furthermore, observe that if $\tau(v_1, \boldsymbol{\varepsilon})$ is idempotent for a non-empty word $v_1 \in \Sigma^*$, then v_1 is null-transparent. Thus, for every non-empty word $v_1 \in \Sigma^*$ we have that v_1^e is a *null-transparent power*. We call a run $s \xrightarrow{(x, \boldsymbol{z})} t$ an *N-path* if the run switches from Q_1 to $Q \setminus Q_1$ at most N times. A non-empty word $y \in \Sigma^*$ is called *N-invisible in the context of $x \in \Sigma^*$* if any N-path $s \xrightarrow{(x, \boldsymbol{z})} t$ implies $t \xrightarrow{(y, \boldsymbol{\varepsilon})} t$. The bound (i) of the following lemma is proven in [Ste67] and the bound (ii) in [Val75].

Lemma 6.3.3. *Let n be the number of states of $\mathcal{A}$ and let $u_1 \dots u_\ell \in \Sigma^*$ be a product of $\ell \in \mathbb{N}$ non-empty words.*

(i) If $\ell > n!$, then some factor $u_{i+1} \dots u_j$ for $1 \leq i < j \leq \ell$ is null-transparent.

(ii) If $\ell > 2(Nn)^n$ for some $N \geq 0$, then some factor $u_{i+1} \dots u_j$ for $1 \leq i < j \leq \ell$ is N-invisible in the context of $u_1 \dots u_i$.

We say that a set S *separates* two sets X and Y if $X \subseteq S$ and $Y \cap S = \emptyset$ or $Y \subseteq S$ and $X \cap S = \emptyset$. If X is a singleton $\{x\}$, we also say that S separates x and Y (similar if Y is a singleton).

The implication 2) $\Rightarrow$ 1) of the following proposition already appeared in [CCG06] as part of the proof of Lemma 3.5 to show that every $\approx_1$-equivalence class contains a word whose length does not exceed some bound depending on n. However, in our understanding, to prove this implication, the authors used 3) as an intermediate step. Unfortunately, the proof of the implication 3) $\Rightarrow$ 1) contains an argument that we could not follow. For completeness, we reprove the implication 3) $\Rightarrow$ 1) using 4) as an intermediate step.

Proposition 6.3.4. *The following conditions are equivalent:*

1) $\approx_1$ has infinite index.

2) There exist words $x, y, z \in \Sigma^*$ such that y is $nn!$-invisible in the context of x and $xyz \not\approx_1 xz$.

3) There exist $\boldsymbol{v}, \boldsymbol{w} \in (\Sigma^*)^k$ and a state $q \in Q$ such that $q \xrightarrow{\boldsymbol{v}} q$, v_1 is null-transparent, and R_q separates $\boldsymbol{w}$ and $(v_1, \boldsymbol{\varepsilon})\boldsymbol{w}$.

4) There exist $\boldsymbol{v}, \boldsymbol{w} \in (\Sigma^*)^k$ and a state $q \in Q$ such that $q \xrightarrow{\boldsymbol{v}} q$, v_1 is non-empty, and R_q separates $\boldsymbol{w}$ and $(v_1, \boldsymbol{\varepsilon})^+\boldsymbol{w}$.

Proof. Let us first prove the easier directions.

4) $\Rightarrow$ 1): Consider any run $q_0 \xrightarrow{\boldsymbol{u}} q$. Then $u_1 v_1^i w_1 \not\approx_1 u_1 v_1^{i+j} w_1$ for all $i \geq 0$ and $j \geq 1$ since R separates $\boldsymbol{u}\boldsymbol{v}^i\boldsymbol{w}$ and $\boldsymbol{u}\boldsymbol{v}^i(v_1, \boldsymbol{\varepsilon})^j\boldsymbol{w}$. Hence, $\approx_1$ has infinite index.

1) $\Rightarrow$ 2): For the sake of contraposition assume that 2) is false. By (ii) of Lemma 6.3.3, any word of length greater than $f(nn!)$, where $f(N) := 2(Nn)^n$, can be decomposed as uvw such that v is $nn!$-invisible in the context of u and therefore by assumption, $uvw \approx_1 uw$. Repeating this argument yields for any word an $\approx_1$-equivalent word of length at most $f(nn!)$. Therefore, $\approx_1$ has finite index.

2) $\Rightarrow$ 3): Assume that y is $nn!$-invisible in the context of x and $xyz \not\approx_1 xz$. Recall that the length of a tuple of words is the sum of the lengths of its entries. Choose a length-minimal tuple $\boldsymbol{t} \in (\Sigma^*)^{k-1}$ such that

$$(xyz, \boldsymbol{t}) \in R \quad \Longleftrightarrow \quad (xz, \boldsymbol{t}) \notin R. \tag{6.4}$$

In other words, $\boldsymbol{t}$ witnesses that $xyz \not\approx_1 xz$. Let ρ be the unique infix of the run on $(xyz, \boldsymbol{t})$ which is a run from some $p_0 \in Q_1$ to some $p_1 \in Q_1$ reading x on the first tape. Observe that ρ is not an $nn!$-path since y is $nn!$-invisible in the context of x and otherwise one could remove the $(y, \boldsymbol{\varepsilon})$-loop from the run on $(xyz, \boldsymbol{t})$, which would contradict Equation (6.4). In particular, ρ switches from a state in Q_1 to a state in $Q \setminus Q_1$ and then back to a state in Q_1 more than $nn!$ times. Consider the sequence of states in ρ visited directly after switching from $Q \setminus Q_1$ to Q_1. By the pigeonhole principle, there is a state $q \in Q_1$ that occurs in this sequence more than $n!$ times. We can factor $x = \alpha_1 \dots \alpha_{\ell+1}$, where $\alpha_1, \dots, \alpha_\ell$ are non-empty, and a prefix of $\boldsymbol{t}$ into tuples $\boldsymbol{\tau}_1 \dots \boldsymbol{\tau}_{\ell+1}$, where each of the $\boldsymbol{\tau}_1, \dots, \boldsymbol{\tau}_\ell$ has length greater 0, such that

$$p_0 \xrightarrow{(\alpha_1, \boldsymbol{\tau}_1)} q \xrightarrow{(\alpha_2, \boldsymbol{\tau}_2)} q \xrightarrow{(\alpha_3, \boldsymbol{\tau}_3)} \cdots \xrightarrow{(\alpha_\ell, \boldsymbol{\tau}_\ell)} q \xrightarrow{(\alpha_{\ell+1}, \boldsymbol{\tau}_{\ell+1})} p_1$$

for some $\ell > n!$. By (i) of Lemma 6.3.3, there exists a null-transparent factor $\alpha_{i+1} \dots \alpha_j$ for some $1 \leq i < j \leq \ell$. Let us set $x_1 := \alpha_1 \dots \alpha_i$, $x_2 := \alpha_{i+1} \dots \alpha_j$, and $x_3 := \alpha_{j+1} \dots \alpha_{\ell+1}$. Consider the corresponding decomposition $\boldsymbol{t} = \boldsymbol{t}_1\boldsymbol{t}_2\boldsymbol{t}_3$ such that

$$q_0 \xrightarrow{(x_1, \boldsymbol{t}_1)} q \xrightarrow{(x_2, \boldsymbol{t}_2)} q \xrightarrow{(x_3yz, \boldsymbol{t}_3)} r_+ \tag{6.5}$$

and

$$q_0 \xrightarrow{(x_1, \boldsymbol{t}_1)} q \xrightarrow{(x_2, \boldsymbol{t}_2)} q \xrightarrow{(x_3z, \boldsymbol{t}_3)} r_- \tag{6.6}$$

where exactly one of the states $r_+, r_- \in Q$ belongs to F.

Since $\boldsymbol{t}$ is a length-minimal tuple satisfying Equation (6.4) and $\boldsymbol{t}_2$ has length greater 0, we know that

$$(xyz, \boldsymbol{t}_1\boldsymbol{t}_3) \in R \quad \Longleftrightarrow \quad (xz, \boldsymbol{t}_1\boldsymbol{t}_3) \in R$$

and thus

$$(x_2x_3yz, \boldsymbol{t}_3) \in R_q \quad \Longleftrightarrow \quad (x_2x_3z, \boldsymbol{t}_3) \in R_q. \tag{6.7}$$

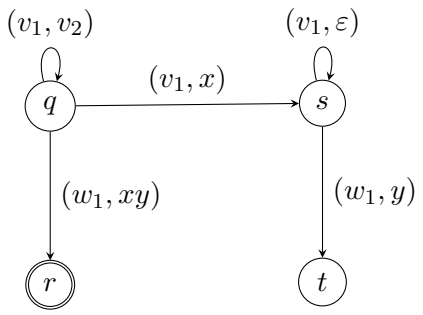


Figure 6.4: The pattern witnessing non-recognizability for deterministic 2-tape automata. Here, exactly one of the states r and t is final.

We claim that (i) R_q separates $(x_2x_3yz, \boldsymbol{t}_3)$ and $(x_3yz, \boldsymbol{t}_3)$ or (ii) R_q separates $(x_2x_3z, \boldsymbol{t}_3)$ and $(x_3z, \boldsymbol{t}_3)$. Otherwise, Equation (6.7) implies

$$(x_3yz, \boldsymbol{t}_3) \in R_q \iff (x_3z, \boldsymbol{t}_3) \in R_q$$

which contradicts Equations (6.5) and (6.6). Now, we set $\boldsymbol{v} := (x_2, \boldsymbol{t}_2)$ and either set $\boldsymbol{w} := (x_3yz, \boldsymbol{t}_3)$ in case (i) or set $\boldsymbol{w} := (x_3z, \boldsymbol{t}_3)$ in case (ii). Then $\boldsymbol{v}$, $\boldsymbol{w}$, and q satisfy 3).

3) $\Rightarrow$ 4): Let $\boldsymbol{v}, \boldsymbol{w} \in (\Sigma^*)^k$ and $q \in Q$ such that $q \xrightarrow{\boldsymbol{v}} q$, v_1 is null-transparent, and R_q separates $\boldsymbol{w}$ and $(v_1, \boldsymbol{\varepsilon})\boldsymbol{w}$. Let $m > |w_2 \dots w_k| + 1$. Let ρ be the run on $(v_1, \boldsymbol{\varepsilon})^m \boldsymbol{w}$ starting in q. It contains a subrun reading $(v_1, \boldsymbol{\varepsilon})$ between two states in Q_1, i.e. we can factor $(w_2, \dots, w_k) = \boldsymbol{xy}$ such that ρ is of the form

$$q \xrightarrow{(v_1^{i-1}, \boldsymbol{x})} r \xrightarrow{(v_1, \boldsymbol{\varepsilon})} s \xrightarrow{(v_1^{m-i} w_1, \boldsymbol{y})} t$$

for some $r, s \in Q_1$ and $i \in [1, m]$. Since v_1 is null-transparent, there is a cycle $s \xrightarrow{(v_1, \boldsymbol{\varepsilon})} s$. Therefore, $V := \{(v_1, \boldsymbol{\varepsilon})^j \boldsymbol{w} \mid j \geq m\}$ is either contained in R_q or disjoint from R_q. Since R_q separates $\boldsymbol{w}$ and $(v_1, \boldsymbol{\varepsilon})\boldsymbol{w}$, it also separates one of them from V. If R_q separates $\boldsymbol{w}$ and V, then $\boldsymbol{v}^m$ and $\boldsymbol{w}$ satisfy the conditions in 4). Otherwise, R_q separates $(v_1, \boldsymbol{\varepsilon})\boldsymbol{w}$ and V, meaning that the tuples $\boldsymbol{v}^m$ and $(v_1, \boldsymbol{\varepsilon})\boldsymbol{w}$ satisfy the conditions in 4). □

6.3.2 Polynomial-Time Algorithm for Binary Relations

From Proposition 6.3.4 we can derive a pattern which is present in $\mathcal{A}$ if and only if R is *not* recognizable. For binary relations the pattern is visualized in Figure 6.4. In this case, the pattern can be detected in polynomial-time by a reduction to the inequivalence problem for deterministic 2-tape automata.

Proposition 6.3.5. *The equivalence relation $\approx_1$ has infinite index if and only if there exist words $v_1, w_1 \in \Sigma^*$ with v_1 non-empty, tuples $\boldsymbol{v}_2, \boldsymbol{x}, \boldsymbol{y} \in (\Sigma^*)^{k-1}$, and states $q, s \in Q$ such that*

- $q \xrightarrow{(v_1, \boldsymbol{v}_2)} q$, $q \xrightarrow{(v_1, \boldsymbol{x})} s$, $s \xrightarrow{(v_1, \boldsymbol{\varepsilon})} s$ *and*

- $(w_1, \boldsymbol{xy}) \in R_q \iff (w_1, \boldsymbol{y}) \notin R_s$.

Proof. For the "if" direction observe that R_q separates $(w_1, \boldsymbol{xy})$ and $(v_1, \boldsymbol{\varepsilon})^+(w_1, \boldsymbol{xy})$. Therefore, $\approx_1$ has infinite index by 4) of Proposition 6.3.4. For the "only if" direction assume that $\approx_1$ has infinite index. Again, by 4) of Proposition 6.3.4, there exist $(v_1, \boldsymbol{v}_2), (w_1, \boldsymbol{w}_2) \in (\Sigma^*)^k$ and a state $q \in Q$ such that $q \xrightarrow{(v_1, \boldsymbol{v}_2)} q$, v_1 is non-empty, and R_q separates $(w_1, \boldsymbol{w}_2)$ and $(v_1, \boldsymbol{\varepsilon})^+(w_1, \boldsymbol{w}_2)$. Let $m > \|\boldsymbol{w}_2\| + 1$ and let v_1^ℓ be a null-transparent power with $\ell \geq 1$. Consider the unique run ρ_q on $(v_1, \boldsymbol{\varepsilon})^{m\ell}(w_1, \boldsymbol{w}_2)$ starting from q. It must contain a subrun of the form $r \xrightarrow{(v_1, \boldsymbol{\varepsilon})^\ell} s$ where $r, s \in Q_1$. Hence, we can factorize $\boldsymbol{w}_2 = \boldsymbol{xy}$ such that ρ_q has the form

$$q \xrightarrow{(v_1^{(i-1)\ell}, \boldsymbol{x})} r \xrightarrow{(v_1, \boldsymbol{\varepsilon})^\ell} s \xrightarrow{(v_1^{(m-i)\ell} w_1, \boldsymbol{y})} t \tag{6.8}$$

for some $i \in [1, m]$. Since v_1^ℓ is null-transparent, there exists a cycle $s \xrightarrow{(v_1, \boldsymbol{\varepsilon})^\ell} s$. Since $\mathcal{A}$ is deterministic, this allows us to choose $i = m$ in Equation (6.8) and write ρ_q as

$$q \xrightarrow{(v_1^{(m-1)\ell}, \boldsymbol{x})} r \xrightarrow{(v_1, \boldsymbol{\varepsilon})^\ell} s \xrightarrow{(w_1, \boldsymbol{y})} t.$$

Since R_q separates $(w_1, \boldsymbol{xy})$ and $(v_1, \boldsymbol{\varepsilon})^{m\ell}(w_1, \boldsymbol{xy})$, we therefore have that $(w_1, \boldsymbol{xy}) \in R_q$ if and only if $(w_1, \boldsymbol{y}) \notin R_s$. Thus, the words $v_1^{m\ell}, w_1$ together with the tuples $\boldsymbol{v}_2^{m\ell}, \boldsymbol{x}, \boldsymbol{y}$ and states q, s are as desired. □

As mentioned above, for binary relations detecting the pattern from Proposition 6.3.5 can be reduced to the problem of checking whether two deterministic 2-tape automata are *equivalent*, i.e. whether they accept the same relation.

Theorem 6.3.6. *The recognizability problem for binary deterministic rational relations is logspace reducible to the equivalence problem for deterministic 2-tape automata.*

Proof. Assume the deterministic rational relation R that we fixed above is binary. By Proposition 3.3.7, R is recognizable if and only if $\approx_1$ has finite index. By Proposition 6.3.5, $\approx_1$ has finite index if and only if for all state pairs $q, s \in Q$ and all words $v_1, w_1, v_2, x, y \in \Sigma^*$ with v_1 non-empty the following two statements are equivalent:

(i) $q \xrightarrow{(v_1, v_2)} q$, $q \xrightarrow{(v_1, x)} s$, $s \xrightarrow{(v_1, \varepsilon)} s$, and $(w_1, xy) \in R_q$

(ii) $q \xrightarrow{(v_1, v_2)} q$, $q \xrightarrow{(v_1, x)} s$, $s \xrightarrow{(v_1, \varepsilon)} s$, and $(w_1, \boldsymbol{y}) \in R_s$

Using an appropriate encoding, we can reduce the equivalence of (i) and (ii) to the equivalence problem of two deterministic 2-tape automata.

Suppose π and ρ are runs that both read the word $v_1 = a_1 \dots a_n$ with $a_i \in \Sigma$ on the first tape, i.e. we can write

$$\pi\colon\ s_0 \xrightarrow{g_0} t_0 \xrightarrow{a_1} s_1 \xrightarrow{g_1} t_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} s_n \xrightarrow{g_n} t_n$$

and

$$\rho\colon\ s'_0 \xrightarrow{h_0} t'_0 \xrightarrow{a_1} s'_1 \xrightarrow{h_1} t'_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} s'_n \xrightarrow{h_n} t'_n$$

where $g_i, h_i \in \Sigma^*$ for all $i \in [1, n]$ and the states t_i, t'_i for $0 \leq i < n$ are precisely the states in π and ρ that are contained in Q_1. We define their *synchronized shuffle* $\pi \sqcup\!\sqcup \rho \in (\Sigma \cup \{\diamond\})^*$ with $\diamond \notin \Sigma$ as

$$\pi \sqcup\!\sqcup \rho := g_0 \diamond h_0 \diamond a_1 \diamond g_1 \diamond h_1 \diamond a_2 \diamond \cdots \diamond a_n \diamond g_n \diamond h_n.$$

We encode (i) as the binary relation

$$S_1 := \{(qsw_1, (\pi \sqcup\!\sqcup \rho)\$y) \mid q, s \in Q,\ \pi\colon q \xrightarrow{(v_1,v_2)} q,\ \rho\colon q \xrightarrow{(v_1,x)} s,\ s \xrightarrow{(v_1,\varepsilon)} s,\ (w_1, xy) \in R_q\}$$

and (ii) as the binary relation

$$S_2 := \{(qsw_1, (\pi \sqcup\!\sqcup \rho)\$y) \mid q, s \in Q,\ \pi\colon q \xrightarrow{(v_1,v_2)} q,\ \rho\colon q \xrightarrow{(v_1,x)} s,\ s \xrightarrow{(v_1,\varepsilon)} s,\ (w_1, y) \in R_s\}$$

where $\$ \notin \Sigma \cup \{\diamond\}$. Observe that $S_1 = S_2$ if and only if (i) and (ii) are equivalent. It remains to verify that S_1 and S_2 are deterministic rational and we can construct deterministic 2-tape automata in logspace. First, for each state pair $q, s \in Q$ we can construct a DFA over $\Sigma \cup \{\diamond\}$ which accepts precisely the synchronized shuffles $\pi \sqcup\!\sqcup \rho$ where $\pi\colon q \xrightarrow{(v_1,v_2)} q$, $\rho\colon q \xrightarrow{(v_1,x)} s$, and $s \xrightarrow{(v_1,\varepsilon)} s$ for some words $v_1, v_2, x \in \Sigma^*$ with v_1 non-empty. Since x can be easily extracted as a subword of $\pi \sqcup\!\sqcup \rho$, a deterministic 2-tape automaton can verify whether the input pair $(qsw_1, (\pi \sqcup\!\sqcup \rho)\$y)$ satisfies $(w_1, xy) \in R_q$ and whether it satisfies $(w_1, y) \in R_s$. □

Since equivalence of two deterministic 2-tape automata can be decided in polynomial time [FG82], Theorem 6.3.6 implies that the recognizability problem for binary deterministic rational relations is decidable in polynomial time, proving the upper bound in Theorem 6.3.2 for binary relations.

6.3.3 Relations of Higher Arity

The approach from Theorem 6.3.6 does not work for arity $k \geq 3$. The issue is that the words v_2, x, y from (i) and (ii) would become $(k-1)$-tuples $\boldsymbol{v}_2, \boldsymbol{x}, \boldsymbol{y}$. It is not clear how to appropriately encode the runs on $(v_1, \boldsymbol{x})$ and $(w_1, \boldsymbol{xy})$ in (i) so that they can be simulated by an automaton. Still, we can express (the negation of) property 3) in Proposition 6.3.4 as the equivalence of two polynomial-space constructible deterministic multitape automata. Since equivalence of deterministic k-tape automata is known to be in coNP [HK91] if k is part of the input and in coRP if k is fixed [Wor13], the complexity bounds from Theorem 6.3.2 follow.

Theorem 6.3.7. *The recognizability problem for k-ary deterministic rational relations is polynomial-space reducible to the equivalence problem for deterministic k-tape automata.*

Proof. Let R be the k-ary deterministic rational relation that we fixed above. By Proposition 3.3.7, R is recognizable if and only if $\approx_j$ has finite index for all $j \in [1, k-1]$. As already mentioned, we can focus on the equivalence relation $\approx_1$. We reduce the test whether $\approx_1$ has finite index to the equivalence problem of polynomial-space constructible deterministic k-tape automata. By 3) of Proposition 6.3.4, $\approx_1$ has finite index if and only if for all states $q \in Q$ and all tuples $\boldsymbol{v}, \boldsymbol{w} \in (\Sigma^*)^k$ the following two statements are equivalent:

(i) $q \xrightarrow{\boldsymbol{v}} q$, v_1 is null-transparent, and $\boldsymbol{w} \in R_q$

(ii) $q \xrightarrow{\boldsymbol{v}} q$, v_1 is null-transparent, and $(v_1, \boldsymbol{\varepsilon})\boldsymbol{w} \in R_q$

We encode (i) and (ii) as deterministic rational relations. First observe that we can construct in polynomial space a DFA for the language of all null-transparent words $v_1 \in \Sigma^*$. It simulates runs on v_1 in parallel from every state $s \in Q_1$ and verifies that $s \xrightarrow{(v_1,\boldsymbol{\varepsilon})} t$ implies $t \xrightarrow{(v_1,\boldsymbol{\varepsilon})} t$. We encode a run π as an alternating sequence $\mathrm{flat}(\pi) \in (Q\Sigma)^*Q$ of states and symbols. Note that a DFA accepting the encoding of valid runs can be constructed in polynomial time. To encode (i) and (ii), we define the k-ary relations

$$S_1 := \{(q\mathrm{flat}(\pi)\$, \boldsymbol{\varepsilon})\boldsymbol{w} \mid q \in Q,\ v_1 \text{ null-transparent},\ \pi\colon q \xrightarrow{\boldsymbol{v}} q,\ \boldsymbol{w} \in R_q\}$$

and

$$S_2 := \{(q\mathrm{flat}(\pi)\$, \boldsymbol{\varepsilon})\boldsymbol{w} \mid q \in Q,\ v_1 \text{ null-transparent},\ \pi\colon q \xrightarrow{\boldsymbol{v}} q,\ (v_1, \boldsymbol{\varepsilon})\boldsymbol{w} \in R_q\}$$

where $\$ \notin \Sigma$. Since v_1 can easily be extracted from $\mathrm{flat}(\pi)$, we can construct in polynomial space deterministic k-tape automata for S_1 and S_2. Furthermore, the statements (i) and (ii) are equivalent if and only if $S_1 = S_2$. □

6.3.4 Reducing Equivalence to Recognizability

Above we solved the recognizability problem for deterministic rational relations by reducing it to the equivalence problem for deterministic multitape automata. In this section we show that there is also a reduction in the reverse direction. In fact, we show that the reverse reduction is even logspace for arbitrary arity. This means that for binary relations the two problems are logspace equivalent.

Theorem 6.3.8. *Let $k \geq 2$. The equivalence problem for deterministic k-tape automata is logspace reducible to the recognizability problem for k-ary deterministic rational relations.*

Proof. Let two deterministic k-tape automata $\mathcal{A}$ and $\mathcal{B}$ be given. First, we ensure that both $R(\mathcal{A})$ and $R(\mathcal{B})$ are finite relations and therefore also recognizable. It was shown by Harju and Karhumäki [HK91] that $\mathcal{A}$ and $\mathcal{B}$ are equivalent if and only if they accept

the same tuples of length at most $n-1$, where n is the total number of states in $\mathcal{A}$ and $\mathcal{B}$. From $\mathcal{A}$ we can compute in logspace a deterministic k-tape automaton $\mathcal{A}'$ such that $R(\mathcal{A}') = R(\mathcal{A}) \cap \{\boldsymbol{u} \in (\Sigma^*)^k \mid \|\boldsymbol{u}\| < n\}$ and analogously $\mathcal{B}'$ from $\mathcal{B}$. The automaton $\mathcal{A}'$ tracks the length of the prefix tuple read so far up to threshold n and rejects all tuples of length at least n.

We claim that $R(\mathcal{A}') = R(\mathcal{B}')$ if and only if

$$\begin{aligned} T := \{(a^i\#, a^i\#, \boldsymbol{\varepsilon}) \mid i \in \mathbb{N}\} R(\mathcal{A}') \cup \\ \{(a^i\#, a^j\#, \boldsymbol{\varepsilon}) \mid i, j \in \mathbb{N}, i \neq j\} R(\mathcal{B}') \end{aligned}$$

is recognizable, where a and $\#$ are fresh distinct symbols. Observe that a deterministic k-tape automaton for T is computable in logspace from $\mathcal{A}'$ and $\mathcal{B}'$. Thus, it remains to prove the claim. If $R(\mathcal{A}') = R(\mathcal{B}')$, then

$$T = \{(a^i\#, a^j\#, \boldsymbol{\varepsilon}) \mid i, j \in \mathbb{N}\} R(\mathcal{A}')$$

is the concatenation of two recognizable relations and hence itself recognizable. Conversely, suppose that $R(\mathcal{A}') \neq R(\mathcal{B}')$ and assume that there exists a tuple $\boldsymbol{v} \in R(\mathcal{A}') \setminus R(\mathcal{B}')$ (the case where $R(\mathcal{B}') \setminus R(\mathcal{A}') \neq \emptyset$ is symmetric). For the sake of contradiction, suppose that T is recognizable. Then $T\boldsymbol{v}^{-1} := \{\boldsymbol{u} \mid \boldsymbol{u}\boldsymbol{v} \in T\}$ is recognizable and since the class of recognizable relations is closed under intersection, also

$$T\boldsymbol{v}^{-1} \cap \{(a^i\#, a^j\#, \boldsymbol{\varepsilon}) \mid i, j \in \mathbb{N}\}$$

is recognizable. This, however, results in a contradiction since

$$T\boldsymbol{v}^{-1} \cap \{(a^i\#, a^j\#, \boldsymbol{\varepsilon}) \mid i, j \in \mathbb{N}\} = \{(a^i\#, a^i\#, \boldsymbol{\varepsilon}) \mid i \in \mathbb{N}\}$$

is clearly not recognizable. □

6.3.5 Constructing an Independent Multitape Automaton

We already observed in Section 2.3.5 that the recognizable relations are precisely the relations accepted by independent multitape automata. Recall that an independent k-tape automaton is a tuple $\mathcal{I} = (\mathcal{A}_1, \ldots, \mathcal{A}_k, F)$ consisting of DFAs $\mathcal{A}_i$ without final states and a set of state tuples $F \subseteq Q_1 \times \cdots \times Q_k$, where Q_i is the state set of $\mathcal{A}_i$. The relation accepted by $\mathcal{I}$ is the set of all word-tuples $(w_1, \ldots, w_k)$ such that for each $i \in [1, k]$ the run of $\mathcal{A}_i$ on w_i ends in a state $q_i \in Q_i$ and it holds that $(q_1, \ldots, q_k) \in F$. In other words, an independent multitape automaton has for each tape a separate automaton and they can only communicate at the end of runs where it is checked whether the combination of states reached is considered final. In Theorem 6.3.2 we showed how to *decide* whether a deterministic rational relation is recognizable. This raises the question of how to *compute* an independent multitape automaton for a recognizable relation given as a deterministic multitape automaton. This problem was also raised in [Bar+19]. The following theorem provides an answer.

Theorem 6.3.9. *Given a deterministic k-tape automaton for a recognizable relation R, one can compute in double exponential time an independent k-tape automaton for R.*

Proof. Let $\mathcal{A}$ be a deterministic k-tape automaton for the recognizable relation $R \subseteq (\Sigma^*)^k$. By Proposition 3.3.7, $\approx_j$ has finite index for all $j \in [1,k]$. For every $j \in [1,k]$ define the relation $\equiv_j \subseteq \Sigma^* \times \Sigma^*$ such that

$$x \equiv_j y \quad :\Longleftrightarrow \quad \text{for all } z \in \Sigma^* \colon xz \approx_j yz$$

which is a right-congruence, i.e. for all $a \in \Sigma$ we have that $x \equiv_j y$ implies $xa \equiv_j ya$. Let n be the number of states of $\mathcal{A}$. Suppose that $x, y, z \in \Sigma^*$ are words such that y is $nn!$-invisible in the context of x. Then $xyz \equiv_j xz$ since otherwise $xyzz' \not\approx_j xzz'$ for some $z' \in \Sigma^*$, which would contradict 2) of Proposition 6.3.4. Hence, for each word $w \in \Sigma^*$ of length $f(nn!) + 1$, where $f(N) := 2(Nn)^n$, there exists an $\equiv_j$-equivalent word $w' \in \Sigma^*$ of length at most $f(nn!)$ by cutting out an $nn!$-invisible factor, whose existence is guaranteed by (ii) of Lemma 6.3.3. Furthermore, w' can be computed from w in double exponential time as remarked in [Val75].

We can now construct the independent k-tape automaton $\mathcal{I} = (\mathcal{A}_1, \ldots, \mathcal{A}_k, F)$ as follows. The states of $\mathcal{A}_j$ are words in Σ^* of length at most $f(nn!)$. The initial state is the empty word ε. If the current state is $w \in \Sigma^*$ and the next input symbol is $a \in \Sigma$, then the next state is the word obtained from wa after removing an $nn!$-invisible factor if possible. In this way, after each step the reached state is a word that is $\equiv_j$-equivalent to the read prefix so far. A tuple of states $\boldsymbol{v} \in (\Sigma^*)^k$ is contained in F if and only if $\boldsymbol{v} \in R$, which can be tested by running $\mathcal{A}$ on $\boldsymbol{v}$. By the above observation, on input tuple $(w_1, \ldots, w_k) \in (\Sigma^*)^k$ each DFA $\mathcal{A}_j$ reaches a state $v_j \in \Sigma^*$ with $v_j \equiv_j w_j$ and therefore $(v_1, \ldots, v_k) \in R$ if and only if $(w_1, \ldots, w_k) \in R$. Thus, I accepts the relation R. □

We remark that the double exponential bound in Theorem 6.3.9 is optimal, which can be derived from the proof by Meyer and Fischer [MF71] for the double exponential succinctness gap between DPDAs and DFAs. We provide a more direct alternative proof.

Proposition 6.3.10. *There exists a family of recognizable relations $R_n \subseteq \{0,1\}^* \times \{0,1\}^*$ with $n \in \mathbb{N}$ such that each R_n is accepted by a deterministic 2-tape automaton with $\mathcal{O}(n^2 \log n)$ states, but any independent 2-tape automaton for R_n has in total at least $2^{2^{n-1}}$ states.*

Proof. Let $R_n \subseteq [1,n]^* \times [1,n]^*$ be the relation containing all pairs (u, v) such that $|v| \leq 2n$ and v is a subword of u. Observe that R_n is accepted by a deterministic 2-tape automaton with $\mathcal{O}(n)$ states (see Example 2.3.29 for the subword relation). Then $\approx_1^{R_n}$ is *Simon's congruence* [Sim75] with parameter $2n$. Its index is finite and bounded from below by $2^{2^{n-1}}$ [KKS15]. If an independent 2-tape automaton $\mathcal{I} = (\mathcal{A}_1, \mathcal{A}_2, F)$ accepts R_n, then the index of $\approx_1^{R_n}$ is a lower bound for the number of states of $\mathcal{A}_1$. Finally, we can replace the alphabet $[1,n]$ by codes from $\{0,1\}^{\lceil \log n \rceil}$, increasing the size of the deterministic 2-tape automaton for R_n by a $(\log n)$-factor. □

The construction from Theorem 6.3.9 will be needed in the next section, where we consider the problem of deciding whether a given deterministic multitape automaton accepts a regular relation.

6.4 Regularity of Deterministic Rational Relations

In this section we show that for deterministic rational relations the regularity problem, i.e. checking whether the relation is regular, can be reduced to the recognizability problem, which can be decided according to Theorem 6.3.2. The reduction allows us to close the remaining gap in the decidability landscape of the considered membership problems over finite-word relations (Figure 6.1), solving an open problem from [CCG06; IT12].

Theorem 6.4.1. *Given a deterministic rational relation $R \subseteq (\Sigma^*)^k$, one can decide whether R is regular (i) in* P *if $k = 2$ and (ii) in $(2k-4)$-*EXP *if $k > 2$ is fixed.*

In the proof we use the well-known *Myhill-Nerode equivalence relation* $\sim_L \subseteq \Sigma^* \times \Sigma^*$ for a language $L \subseteq \Sigma^*$. It is defined such that

$$u \sim_L v \quad :\Longleftrightarrow \quad \forall w \in \Sigma^* \colon uw \in L \Leftrightarrow vw \in L$$

for all $u, v \in \Sigma^*$. It was shown in [NS57] that $\sim_L$ has finite index if and only if L is a regular language.

Before we dive into the proof of Theorem 6.4.1, let us first give an intuition for the binary case. Let R be given by a deterministic 2-tape automaton $\mathcal{A}$ such that every state in its state set Q is reachable. First, suppose that $\mathcal{A}$ has the property that every cycle $p \xrightarrow{(v_1,v_2)} p$ satisfies $|v_1| = |v_2|$. This ensures that $\mathcal{A}$ has *bounded delay*, i.e. the head positions cannot move arbitrarily far apart during the computation of $\mathcal{A}$. In fact, the delay is bounded by the number of states $|Q|$. It is well-known [FS93] that such an automaton recognizes a regular relation since symbols that are "read ahead" on a tape can be stored in a queue of length at most $|Q|$: The NFA $\mathcal{A}'$ that simulates $\mathcal{A}$ stores in its states a state of $\mathcal{A}$ and a word (that is handled like a queue) for both of the two input components. When $\mathcal{A}'$ is in a state in Q_1 with empty queues and reads a pair (a, b), it simulates $\mathcal{A}$ on input a and stores b in the queue for the second component. Later, if a state in Q_2 is reached, it simulates $\mathcal{A}$ on b while taking an ε-transition. Note that by the observation above, $\mathcal{A}'$ only needs to store words of a bounded length, meaning that the state set of $\mathcal{A}'$ is finite. A state of $\mathcal{A}'$ is accepting if the current state of $\mathcal{A}$ is final and both queues are empty.

Let us now consider the case where $\mathcal{A}$ contains an *asynchronous* cycle of the form $p \xrightarrow{(v_1,v_2)} p$ with $|v_1| < |v_2|$ (the case $|v_1| > |v_2|$ is symmetric). We partition $\mathcal{A}$ into an asynchronous part, containing all states which are reachable from an asynchronous cycle, and a synchronous part. While the simulation using a queue works in the synchronous part, the delay can become unbounded in the asynchronous part by traversing asynchronous cycles repeatedly. We claim that R is regular if and only if for every state q in the asynchronous part the relation R_q is recognizable. If each such relation R_q is recognizable and in particular regular, then the computation from state q can be continued synchronously using an NFA for R_q. For the other direction assume that R_q is not recognizable for some state q in the asynchronous part. Then by Proposition 3.3.7, $\approx_1^{R_q}$ has infinite index, i.e. there exist words $(s_i)_{i\geq 1}$ and $(t_{i,j})_{1\leq i<j}$ such that R_q separates $(s_i, t_{i,j})$ and $(s_j, t_{i,j})$ for all $1 \leq i < j$. We claim that the Myhill-Nerode equivalence

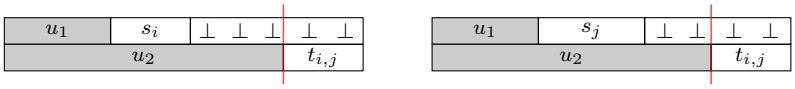


Figure 6.5: An asynchronous cycle can produce runs $q_0 \xrightarrow{(u_1,u_2)} q$ where u_1, u_2 have unbounded length difference. The words s_i, s_j are pairwise $\approx_1^{R_q}$-inequivalent witnessed by the word $t_{i,j}$. Then the convolutions $(u_1 s_i) \otimes u_2$ and $(u_1 s_j) \otimes u_2$ left of the red line are $\sim_{\otimes R}$-inequivalent witnessed by the convolution $\varepsilon \otimes t_{i,j}$ right of the red line.

relation $\sim_{\otimes R}$ of the language $\otimes R$ of convolutions has at least h classes for every $h \geq 1$: Take an asynchronous cycle $p \xrightarrow{(v_1,v_2)} p$ from which q is reachable. We can produce runs $q_0 \xrightarrow{(u_1,u_2)} q$ where the delay $|u_2| - |u_1|$ is arbitrarily large. Pick such a run where $|u_2| - |u_1| \geq \max\{|s_1|, \ldots, |s_h|\}$. Then any two words $(u_1 s_i) \otimes u_2$ and $(u_1 s_j) \otimes u_2$ for $1 \leq i < j \leq h$ are inequivalent with respect to $\sim_{\otimes R}$ since $\otimes R$ separates $(u_1 s_i) \otimes (u_2 t_{i,j})$ and $(u_1 s_j) \otimes (u_2 t_{i,j})$. For an illustration see Figure 6.5. Thus, $\sim_{\otimes R}$ has infinite index, which implies that $\otimes R$ and therefore also R are not regular. Hence, the regularity problem can be reduced to checking recognizability of relations R_q where q is reachable from an asynchronous cycle.

Let us now consider the general case of a k-ary deterministic rational relation R. Again, we can ignore the endmarker $\dashv$ since appending $(\dashv, \ldots, \dashv)$ to R preserves (non-)synchronicity. Hence, for the rest of this section we assume that $R \subseteq (\Sigma^*)^k$ is given by a deterministic k-tape automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ with $R = R(\mathcal{A})$. Moreover, we assume that every state in Q is reachable from q_0. A cycle in $\mathcal{A}$ reading $(v_1, \ldots, v_k) \in (\Sigma^*)^k$ *induces* a partition P on the components $[1, k]$ where two components i and j are in the same block in P if and only if $|v_i| = |v_j|$. For a state $q \in Q$ we define the partition P_q as the coarsest refinement of all partitions induced by a cycle from which q is reachable. As before, let R_q be the relation recognized from state q.

Lemma 6.4.2. *For every $q \in Q$ the partition P_q is computable in time polynomial in the size of $\mathcal{A}$.*

Proof. The algorithm proceeds as follows. As a first step, we compute for each $q \in Q$ the coarsest refinement S_q of all partitions that are induced by *simple* cycles on q. To this end, check for every $1 \leq i < j \leq k$ whether there exists a simple cycle $q \xrightarrow{\boldsymbol{v}} q$ such that $|v_i| \neq |v_j|$ and if so, store it as a constraint that i and j are in different blocks. Then S_q is the coarsest partition of $[1, k]$ that fulfills all stored constraints. Note that the existence of a simple cycle $q \xrightarrow{\boldsymbol{v}} q$ with $|v_i| \neq |v_j|$ can be checked in nondeterministic logspace by storing the current length difference of the words in the i-th and j-th component on the guessed path in a counter whose value is bounded by $|Q|$.

We claim that $P_q = S_{q_1} \sqcap \cdots \sqcap S_{q_n}$ where $q_1, \ldots, q_n$ are the states from which q is reachable. By definition, P_q is finer than $S_{q_1} \sqcap \cdots \sqcap S_{q_n}$. For the other direction let P be a partition induced by a cycle c from which q is reachable. For the sake of contradiction, assume that there exist $1 \leq i < j \leq k$ that are in different blocks in P but in the same

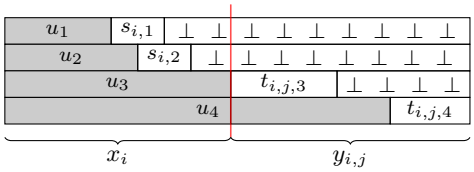


Figure 6.6: If there are h pairwise $\approx^{R_q}_{[1,2]}$-inequivalent tuples $\boldsymbol{s_1}, \ldots, \boldsymbol{s_h}$, then also the Myhill-Nerode equivalence relation $\sim_{\otimes R}$ has at least h classes (witnessed by $x_1, \ldots, x_h$ with $x_i y_{i,j} \in \otimes R$ if and only if $x_j y_{i,j} \notin \otimes R$).

block in $S_{q\ell}$ for all $\ell \in [1, n]$. Since any cycle contains a simple cycle, there exists a simple cycle $p \xrightarrow{\boldsymbol{v}} p$ that is contained in c. By assumption, it holds that $|v_i| = |v_j|$, which means that after removing $p \xrightarrow{\boldsymbol{v}} p$ from c, the cycle c still induces a partition where i and j are in different blocks. Furthermore, q is still reachable from c. We can repeat this argument until c is a simple cycle inducing a partition where i and j are in different blocks, a contradiction. □

Recall that for binary relations we tested recognizability for all states reachable from asynchronous cycles. For relations of higher arity we need to test whether each relation R_q conforms to P_q.

Lemma 6.4.3. *If R_q does not conform to P_q for some state $q \in Q$, then R is not regular.*

Proof. Assume that R_q does not conform to P_q. By Theorem 3.3.5, there exists a partition P induced by a cycle $p \xrightarrow{\boldsymbol{v}} p$ so that q is reachable from p and R_q does not conform to P. By permuting components, we can assume without loss of generality that $|v_1| \leq \cdots \leq |v_k|$. Hence, P is a partition of $[1, k]$ into intervals $B_1, \ldots, B_n$, which are listed in ascending order. Since the intervals $B_1 \cup \cdots \cup B_i$ for $i \in [1, n]$ generate P, there exists an index $r \in [1, n-1]$ such that $\approx^{R_q}_{B_1 \cup \cdots \cup B_r}$ has infinite index by Lemma 3.3.6. Let $1 \leq m < k$ such that $B_1 \cup \cdots \cup B_r = [1, m]$. Observe that for any number $b \in \mathbb{N}$ there exists a run $q_0 \xrightarrow{\boldsymbol{u}} q$ such that $|u_i| + b \leq |u_j|$ for all $i \in [1, m]$ and $j \in [m+1, k]$. Such runs can be constructed by traversing the cycle $p \xrightarrow{\boldsymbol{v}} p$ sufficiently often.

Similarly as in the binary case, we show that for every $h \geq 1$ the Myhill-Nerode equivalence relation $\sim_{\otimes R}$ of the language $\otimes R$ of convolutions has at least h classes. This proves that $\sim_{\otimes R}$ has infinite index, which implies that $\otimes R$ and therefore also R are not regular.

Let $h \geq 1$. Since $\approx^{R_q}_{[1,m]}$ has infinite index, there are tuples $\boldsymbol{s_i} := (s_{i,1}, \ldots, s_{i,m})$ for $i \in [1, h]$ and $\boldsymbol{t_{i,j}} := (t_{i,j,m+1}, \ldots, t_{i,j,k})$ for $1 \leq i < j \leq h$ such that $(\boldsymbol{s_i}, \boldsymbol{t_{i,j}}) \in R_q$ if and only if $(\boldsymbol{s_j}, \boldsymbol{t_{i,j}}) \notin R_q$ for all $1 \leq i < j \leq h$. Let $b := \max\{|s_{i,j}| \mid i \in [1, h], j \in [1, m]\}$. By the above observation, there exists a run $q_0 \xrightarrow{\boldsymbol{u}} q$ such that $|u_i| + b \leq |u_j|$ for all $i \in [1, m]$ and $j \in [m+1, k]$. Therefore, there exists a number $\ell \in \mathbb{N}$ such that all words in the m-tuple $(u_1, \ldots, u_m)\boldsymbol{s_i}$ have length at most ℓ and all words in the $(k-m)$-tuple $(u_{m+1}, \ldots, u_k)$ have length at least ℓ. See Figure 6.6 for an illustration. Since $\mathcal{A}$ is deterministic, we have $\boldsymbol{u}(\boldsymbol{s_i}, \boldsymbol{t_{i,j}}) \in R$ if and only if $\boldsymbol{u}(\boldsymbol{s_j}, \boldsymbol{t_{i,j}}) \notin R$ for all $1 \leq i < j \leq h$.

This can be turned into a proof that $\otimes R$ has at least h Myhill-Nerode equivalence classes as follows. For a word w of length at least ℓ we denote by $\mathrm{pre}_\ell(w)$ the prefix of w of length ℓ and by $\mathrm{suf}_\ell(w)$ the suffix of w after $\mathrm{pre}_\ell(w)$. For all $i \in [1,h]$ define

$$x_i := u_1 s_{i,1} \otimes \cdots \otimes u_m s_{i,m} \otimes \mathrm{pre}_\ell(u_{m+1}) \otimes \cdots \otimes \mathrm{pre}_\ell(u_k)$$

and for all $1 \leq i < j \leq h$ define

$$y_{i,j} := \varepsilon \otimes \cdots \otimes \varepsilon \otimes \mathrm{suf}_\ell(u_{m+1} t_{i,j,m+1}) \otimes \cdots \otimes \mathrm{suf}_\ell(u_k t_{i,j,k}).$$

Observe that $x_i y_{i,j}$ and $x_j y_{i,j}$ are the convolutions of the tuples $\boldsymbol{u}(\boldsymbol{s_i}, \boldsymbol{t_{i,j}})$ and $\boldsymbol{u}(\boldsymbol{s_j}, \boldsymbol{t_{i,j}})$, respectively, and therefore $x_i \not\sim_{\otimes R} x_j$ for all $1 \leq i < j \leq h$. □

Testing whether a relation conforms to a partition is an a priori more difficult problem than recognizability and it is not clear how to decide it for deterministic rational relations. Instead, we will summarize the components inside each partition block into a single component and test recognizability for the summarized relation.

In the following we always assume that the blocks of a partition $P = \{B_1, \ldots, B_n\}$ are ordered so that $\min(B_1) < \cdots < \min(B_n)$ and each block $B_i = \{b_{i,1}, \ldots, b_{i,|B_i|}\}$ is given such that $b_{i,1} < \cdots < b_{i,|B_i|}$. For a relation $S \subseteq (\Sigma^*)^k$ and a partition P of $[1,k]$ as above we define the *summarized relation*

$$S^P := \{(u_{b_{1,1}} \otimes \cdots \otimes u_{b_{1,|B_1|}}, \ldots, u_{b_{n,1}} \otimes \cdots \otimes u_{b_{n,|B_n|}}) \mid (u_1, \ldots, u_k) \in S\}.$$

We write $R_q^\otimes$ for $R_q^{P_q}$. Under the assumption that $R_q^\otimes$ is deterministic rational, we can test whether R_q conforms to P_q.

Lemma 6.4.4. *Let $S \subseteq (\Sigma^*)^k$ and P be a partition of $[1,k]$. If S^P is deterministic rational, then S^P is recognizable if and only if S conforms to P.*

Proof. Let $P = \{B_1, \ldots, B_n\}$. By Proposition 3.3.7, the summarized relation S^P is recognizable if and only if $\approx^{S^P}_{[1,i]}$ has finite index for all $i \in [1, n-1]$. Let $B_{[1,i]} := B_1 \cup \cdots \cup B_i$. By Lemma 3.3.6, we have that S conforms to P if and only if $\approx^S_{B_{[1,i]}}$ has finite index for all $i \in [1, n-1]$. The claim follows since $\approx^{S^P}_{[1,i]}$ has finite index if and only if $\approx^S_{B_{[1,i]}}$ has finite index. □

We are now ready to complete the reduction from regularity to recognizability, proving Theorem 6.4.1. For a partition P of $[1,k]$ we write $Q_P := \{q \in Q \mid P_q = P\}$. We partition Q into layers $L_1, \ldots, L_k$ where $L_t := \{q \in Q \mid |P_q| = t\}$ for all $t \in [1,k]$. Observe that all states reachable from a state $q \in L_t$ are contained in $L_t \cup \cdots \cup L_k$. The algorithm processes the layers $L_k, L_{k-1}, \ldots, L_1$ in descending order. For each state q in layer L_t the algorithm (i) constructs a deterministic t-tape automaton $\mathcal{A}_q^\otimes$ for $R_q^\otimes$, (ii) tests whether $R_q^\otimes$ is recognizable, and (iii) if so, constructs an independent t-tape automaton $\mathcal{I}_q^\otimes$ for $R_q^\otimes$. For layer L_k the automaton $\mathcal{A}_q^\otimes$ is simply the automaton $\mathcal{A}$ with initial state q. For the other layers the automaton $\mathcal{A}_q^\otimes$ will be built from the automata $\mathcal{I}_{q'}^\otimes$ from the previous

layers, which will be explained below (Lemma 6.4.5). The automata $\mathcal{I}_q^\otimes$ are constructed from $\mathcal{A}_q^\otimes$ using Theorem 6.3.9, which is correct under the assumption that $R_q^\otimes$ is indeed recognizable. If one of the recognizability tests is negative, the algorithm terminates and reports that R is not regular. Otherwise, if all recognizability tests succeed, the algorithm reports that R is regular.

Let us argue that the algorithm is correct. If one of the relations $R_q^\otimes$ is deterministic rational but not recognizable, then R is indeed not regular by Lemmas 6.4.3 and 6.4.4. If all recognizability tests succeed, then in particular the relation $R_{q_0}^\otimes$ is recognizable. This easily implies regularity of R since an independent multitape automaton for $R_{q_0}^\otimes$ can be transformed into a DFA for $\otimes R$. In fact, we can make the algorithm slightly more efficient. Observe that the recognizability tests in layer L_1 will always be positive since the relations $R_q^\otimes$ in layer L_1 are regular languages. Therefore, we can skip processing the last layer L_1 and also skip constructing the independent 2-tape automata in layer L_2.

It remains to show how to construct the deterministic multitape automaton $\mathcal{A}_q^\otimes$ for $R_q^\otimes$ from the independent multitape automata from previous layers.

Lemma 6.4.5. *Given $q \in L_t$ for some $1 \leq t < k$ and independent t-tape automata $\mathcal{I}_{q'}^\otimes$ for $R_{q'}^\otimes$ for all $q' \in L_{t+1} \cup \cdots \cup L_k$, one can compute $\mathcal{A}_q^\otimes$ in time polynomial in the sizes of the $\mathcal{I}_{q'}^\otimes$ and exponential in the size of $\mathcal{A}$.*

Proof. Recall that the state set of the deterministic k-tape automaton $\mathcal{A}$ for R is partitioned into $Q = Q_1 \cup \cdots \cup Q_k$. Let $q \in Q_P$ for a partition $P = \{B_1, \ldots, B_t\}$ of $[1, k]$. We show how to construct a deterministic t-tape automaton $\mathcal{A}_q^\otimes = (Q', \Sigma', \delta', q_0', F')$ with $Q' = Q_1' \cup \cdots \cup Q_t'$ and $\Sigma' := \Sigma_\bot^{|B_1|} \cup \cdots \cup \Sigma_\bot^{|B_t|} \cup \{\dashv\}$ where $\Sigma_\bot := \Sigma \cup \{\bot\}$ and $\bot, \dashv \notin \Sigma$ that accepts the relation $\{\boldsymbol{w}(\dashv, \ldots, \dashv) \mid \boldsymbol{w} \in R_q^\otimes\}$ assuming that we already constructed independent multitape automata $\mathcal{I}_{q'}^\otimes$ for all $q' \in L_{t+1} \cup \cdots \cup L_k$. The automaton $\mathcal{A}_q^\otimes$ consists of two parts. The first part simulates $\mathcal{A}$ over the states in Q_P and the second part simulates the independent multitape automata of states contained in $L_{t+1} \cup \cdots \cup L_k$.

First part In the first part it is possible for $\mathcal{A}_q^\otimes$ to read the components within a block of P synchronously since by definition of Q_P the difference of the tape positions between those components is bounded by $|Q|$. Thus, as in the binary case, it suffices for $\mathcal{A}_q^\otimes$ to store a word (read like a queue) of length at most $|Q|$ for each component and simulate $\mathcal{A}$ either on the next symbol in the queue or on the current input symbol if the queue is empty. If it simulates $\mathcal{A}$ on the input symbol of the corresponding tape, we store the symbols read in the other components in the queues of that components. The simulation of $\mathcal{A}$ on the queue is handled like an ε-transition, where nothing is read from the input. More precisely, the first part consists of states of the form $(p, \boldsymbol{w})$ with $p \in Q_P$ and $w_i \in \Sigma_\bot^{\leq |Q|}$ for $1 \leq i \leq k$. Intuitively, p stores the state of $\mathcal{A}$ the simulation is currently at and $\boldsymbol{w}$ stores for every component a queue of symbols that $\mathcal{A}$ still has to be simulated on. The initial state is $q_0' := (q, \boldsymbol{\varepsilon})$. For input $a' = (a_1, \ldots, a_{|B_i|}) \in \Sigma_\bot^{|B_i|}$ for some $i \in [1, t]$ and $a_j \in \Sigma$ for some $j \in [1, |B_i|]$ and state $(p, \boldsymbol{w})$ with $p \in Q_{b_{i,j}}$, $w_{b_{i,j}} = \varepsilon$, and

$p' := \delta(p, a_j) \in Q_P$ we let $(p, \boldsymbol{w}) \in Q'_i$ and $\delta'((p, \boldsymbol{w}), a') := (p', \boldsymbol{w'})$ where

$$w'_{b_{i',j'}} := \begin{cases} w_{b_{i,j'}} a_{j'}, & \text{if } i' = i \text{ and } j' \neq j \\ w_{b_{i',j'}}, & \text{otherwise} \end{cases}$$

for all $i' \in [1, t]$ and $j' \in [1, |B_{i'}|]$. For state $(p, \boldsymbol{w})$ with $p \in Q_i$ for some $i \in [1, k]$, $w_i = au$ for some $a \in \Sigma$ and $u \in \Sigma_\perp^*$, and $p' := \delta(p, a) \in Q_P$ we define

$$\delta'((p, \boldsymbol{w}), \varepsilon) := (p', w_1, \ldots, w_{i-1}, u, w_{i+1}, \ldots, w_k).$$

Note that these ε-transitions can be eliminated without introducing nondeterminism since there is no branching of ε-transitions possible. To each state of the form $(p, \boldsymbol{\varepsilon})$ with $p \in F$ we append a chain of transitions reading the endmarker $\dashv$ in every component and mark the last state of that chain (which is a sink state) as final.

First to second part We now define the transitions from states of the first part to states of the second part. Let $p' \in Q_{P'}$ for some partition $P' = \{B'_1, \ldots, B'_{t'}\}$ that is strictly finer than P and $\mathcal{I}^{\otimes}_{p'} = (\mathcal{A}_1, \ldots, \mathcal{A}_{t'}, F_{p'})$ be an independent t'-tape automaton for $R^{\otimes}_{p'}$ with $\mathcal{A}_i = (Q^i, \Sigma_\perp^{|B'_i|}, \delta^i, q_0^i)$ for all $i \in [1, t']$. The second part consists of states of the form $(q^1, \ldots, q^{t'}, \boldsymbol{w}, \boldsymbol{e}, m)$ with $q^i \in Q^i$ for all $i \in [1, t']$, $\boldsymbol{w}$ as in the first part, $e_i \in \{0, 1\}$ for all $i \in [1, t']$, and $1 \leq m \leq t + 1$. Intuitively, q^i is the state the simulation of $\mathcal{A}_i$ is currently at, e_i stores in a bit whether the simulation of $\mathcal{A}_i$ is finished, and m indicates which component of $\mathcal{A}^{\otimes}_q$ is currently read. For input $a' = (a_1, \ldots, a_{|B_i|}) \in \Sigma_\perp^{|B_i|}$ for some $i \in [1, t]$ and $a_j \in \Sigma$ for some $j \in [1, |B_i|]$ and state $(p, \boldsymbol{w})$ with $p \in Q_{b_{i,j}}$, $w_{b_{i,j}} = \varepsilon$, and $\delta(p, a_j) = p'$ we let $(p, \boldsymbol{w}) \in Q'_i$ and $\delta'((p, \boldsymbol{w}), a') := (q_0^1, \ldots, q_0^{t'}, \boldsymbol{w'}, \boldsymbol{0}, 1)$ where

$$w'_{b_{i',j'}} := \begin{cases} w_{b_{i,j'}} a_{j'}, & \text{if } i' = i \text{ and } j' \neq j \\ w_{b_{i',j'}}, & \text{otherwise} \end{cases}$$

for all $i' \in [1, t]$ and $j' \in [1, |B_{i'}|]$. For state $(p, \boldsymbol{w})$ with $p \in Q_i$ for some $i \in [1, k]$, $w_i = au$ for some $a \in \Sigma$ and $u \in \Sigma_\perp^*$, and $\delta(p, a) = p'$ we define

$$\delta'((p, \boldsymbol{w}), \varepsilon) := (q_0^1, \ldots, q_0^{t'}, w_1, \ldots, w_{i-1}, u, w_{i+1}, \ldots, w_k, \boldsymbol{0}, 1).$$

Note that these ε-transitions can be eliminated again.

Second part Let p', P', and $\mathcal{I}^{\otimes}_{p'}$ as above, where we denote the elements of a block of P' by $B'_i = \{b'_{i,1}, \ldots, b'_{i,|B'_i|}\}$ for all $i \in [1, t']$ in ascending order as usual. In the second part, $\mathcal{A}^{\otimes}_q$ simulates the independent t'-tape automaton $\mathcal{I}^{\otimes}_{p'}$. For each block B_i with $i \in [1, t]$ the DFAs $\mathcal{A}_j$ that are responsible for components contained in B_i are simulated in parallel either on the queue or the current input symbol. After the whole input was read, $\mathcal{A}^{\otimes}_q$ checks with final states whether the remaining content of the queues leads in each $\mathcal{A}_i$ for $i \in [1, t']$ to some state f^i such that $(f^1, \ldots, f^{t'}) \in F_{p'}$. Let us make this precise.

Let $f\colon \{1,\dots,t'\} \to \{1,\dots,t\}$ with $i' \mapsto i$ such that $B'_{i'} \subseteq B_i$. Note that f is well-defined since P' is finer than P. For input $a' = (a_{b_{i,1}},\dots,a_{b_{i,|B_i|}}) \in \Sigma_\perp^{|B_i|} \setminus \{(\perp,\dots,\perp)\}$ for some $i \in [1,t]$ and state $(q^1,\dots,q^{t'},\boldsymbol{w},\boldsymbol{e},i)$ we let $(q^1,\dots,q^{t'},\boldsymbol{w},\boldsymbol{e},i) \in Q'_i$ and $\delta'((q^1,\dots,q^{t'},\boldsymbol{w},\boldsymbol{e},i),a') := (p^1,\dots,p^{t'},\boldsymbol{w'},\boldsymbol{e'},i)$ where for all $i' \in [1,t']$ we set

$$p^{i'} := \delta^{i'}(q^{i'},(c_{b'_{i',1}},\dots,c_{b'_{i',|B'_{i'}|}}))$$

if $f(i') = i$, $c_{b'_{i',j'}} \neq \perp$ for some $j' \in [1,|B'_{i'}|]$, and $e_{i'} = 0$ and we set $p^{i'} := q^{i'}$ if either $f(i') \neq i$ or $f(i') = i$ and $c_{b'_{i',j'}} = \perp$ for all $j' \in [1,|B'_{i'}|]$. Here, we define

$$c_{b'_{i',j'}} := \begin{cases} a_{b'_{i',j'}}, & \text{if } w_{b'_{i',j'}} = \varepsilon \\ c, & \text{if } w_{b'_{i',j'}} = cu \text{ for some } c \in \Sigma_\perp, u \in \Sigma_\perp^* \end{cases}$$

for all $i' \in [1,t']$ with $f(i') = i$ and $j' \in [1,|B'_{i'}|]$. Furthermore, we let

$$w'_{b'_{i',j'}} := \begin{cases} ua_{b'_{i',j'}}, & \text{if } f(i') = i \text{ and } w_{b'_{i',j'}} = cu \text{ for some } c \in \Sigma_\perp, u \in \Sigma_\perp^* \\ w_{b'_{i',j'}}, & \text{otherwise} \end{cases}$$

and

$$e'_{i'} := \begin{cases} 1, & \text{if } f(i') = i \text{ and } c_{b'_{i',j'}} = \perp \text{ for all } j' \in [1,|B'_{i'}|] \\ e_{i'}, & \text{otherwise} \end{cases}$$

for all $i' \in [1,t']$ and $j' \in [1,|B'_{i'}|]$. For $i \in [1,t]$ and state $(q^1,\dots,q^{t'},\boldsymbol{w},\boldsymbol{e},i)$ we let $(q^1,\dots,q^{t'},\boldsymbol{w},\boldsymbol{e},i) \in Q_i$ and $\delta'((q^1,\dots,q^{t'},\boldsymbol{w},\boldsymbol{e},i),\dashv) := (q^1,\dots,q^{t'},\boldsymbol{w},\boldsymbol{e},i+1)$. A state of the form $(q^1,\dots,q^{t'},\boldsymbol{w},\boldsymbol{e},n+1)$ with $w_i = u_i v_i$ for some $u_i \in \Sigma^*$ and $v_i \in \{\perp\}^*$ is final if for all $i' \in [1,t']$ we have that $\mathcal{A}_{i'}$ reaches state $f^{i'}$ from $q^{i'}$ on reading $u_{b'_{i',1}} \otimes \cdots \otimes u_{b'_{i',|B'_{i'}|}}$ and $(f^1,\dots,f^{t'}) \in F_{p'}$. □

Finally, we argue that the running time of the algorithm is $2(k-2)$-fold exponential in the automaton size $|\mathcal{A}|$. Inductively, we prove that in layer L_t we can construct the automata $\mathcal{A}_q^\otimes$ with $q \in L_t$ in $2(k-t)$-fold exponential time and the automata $\mathcal{I}_q^\otimes$ in $2(k-t+1)$-fold exponential time. In particular, the automata sizes are bounded by their respective construction times. In layer L_k the automata $\mathcal{A}_q^\otimes$ are constructed in polynomial time. In the other layers L_t with $3 \leq t < k$ the automata $\mathcal{A}_q^\otimes$ are constructed by Lemma 6.4.5 in time polynomial in the sizes of the independent multitape automata from previous layers, i.e. in $2(k-t)$-fold exponential time since $2(k-(t+1)-1) = 2(k-t)$. Each automaton $\mathcal{I}_q^\otimes$ is constructed in double exponential time in the size of $\mathcal{A}_q^\otimes$ using Theorem 6.3.9, i.e. $2(k-t+1)$-fold exponential in $|\mathcal{A}|$. Furthermore, each recognizability test on $\mathcal{A}_q^\otimes$ in layer L_t where $t \in [3,k]$ takes double exponential time in $|\mathcal{A}_q^\otimes|$ by Theorem 6.3.2, i.e. $2(k-t+1)$-fold exponential in $|\mathcal{A}|$. The relations $R_q^\otimes$ in layer L_2 are binary and therefore recognizability can be checked in polynomial time in

$|\mathcal{A}_q^{\otimes}|$, i.e. $2(k-2)$-fold exponential in $|\mathcal{A}|$. Note that as argued above, we do not have to check recognizability in layer L_1 and we neither have to construct the $\mathcal{I}_q^{\otimes}$ in layer L_2.

To conclude this section, let us remark that there is also a reduction in the reverse direction, i.e. from recognizability to regularity.

Proposition 6.4.6. *Given a k-tape automaton $\mathcal{A}$ for a relation R, one can compute in logspace a k-tape automaton $\mathcal{B}$ for a relation S such that R is recognizable if and only if S is regular. Moreover, if $\mathcal{A}$ is deterministic, then so is $\mathcal{B}$.*

Proof. Let $R \subseteq (\Sigma^*)^k$ be a rational relation and $\Sigma' := \Sigma \cup \{\#, \vdash\}$ with $\#, \vdash \notin \Sigma$. We claim that R is recognizable if and only if

$$R' := \{(\#^{n_1} \vdash w_1, \ldots, \#^{n_k} \vdash w_k) \mid n_1, \ldots, n_k \geq 0 \text{ and } (w_1, \ldots, w_k) \in R\}$$

is regular. Since from a k-tape automaton for R we can easily construct a k-tape automaton for R' in logspace while preserving determinism, this implies that recognizability is logspace reducible to regularity for both rational relations and deterministic rational relations.

For the "only if" direction assume that $R = \bigcup_{i=1}^{n} L_{i,1} \times \cdots \times L_{i,k}$ for regular languages $L_{i,j} \subseteq \Sigma^*$. Then $R' = \bigcup_{i=1}^{n} \{\#\}^*\{\vdash\}L_{i,1} \times \cdots \times \{\#\}^*\{\vdash\}L_{i,k}$, where the $\{\#\}^*\{\vdash\}L_{i,j} \subseteq (\Sigma')^*$ are clearly regular languages. Thus, R' is recognizable and therefore also regular.

For the "if" direction we assume that R is not recognizable, which by Proposition 3.3.7 means that $\approx_r$ has infinite index for some $r \in [1, k-1]$. Thus, there are words $v_1, v_2, \ldots \in \Sigma^*$ and tuples $\boldsymbol{w_{i,j}} = (w_{i,j,1}, \ldots, w_{i,j,k-1}) \in (\Sigma^*)^{k-1}$ such that $v_i \odot_r \boldsymbol{w_{i,j}} \in R$ if and only if $v_j \odot_r \boldsymbol{w_{i,j}} \notin R$ for all $1 \leq i < j$. We show that the Myhill-Nerode equivalence relation $\sim_{\otimes R'}$ of the language of convolutions $\otimes R'$ has infinite index. This implies that $\otimes R'$ and therefore also R' are not regular. Let

$$x_i := \#^{n_i} \vdash \otimes \cdots \otimes \#^{n_i} \vdash \otimes \vdash v_i \otimes \#^{n_i} \vdash \otimes \cdots \otimes \#^{n_i} \vdash$$

with $\vdash v_i$ in the r-th component and $n_i := |v_i|$ for all $i \geq 1$ and let

$$y_{i,j} := w_{i,j,1} \otimes \cdots \otimes w_{i,j,r-1} \otimes \varepsilon \otimes w_{i,j,r} \otimes \cdots \otimes w_{i,j,k-1}$$

for all $1 \leq i < j$. Then we have that $x_i y_{i,j} \in \otimes R'$ if and only if $x_j y_{i,j} \notin \otimes R'$, which means that $x_i \not\sim_{\otimes R'} x_j$ for all $1 \leq i < j$. □

7 Conclusion

We conclude this thesis with a brief summary of the main results and some future work.

Summary We considered Ramsey quantifiers, which state the existence of infinite cliques, in various first-order theories. For automatic structures, i.e. where the domain and the relations can be captured by automata, we showed that the Ramsey quantifier can be evaluated in logspace when an automaton representing the formula is given. For tree-automatic structures we proved that the evaluation can be performed in polynomial time if a deterministic bottom-up tree automaton is provided or if the relation is transitive. With this we showed that the infinite clique problem and recurrent reachability is NL-complete for regular relations, EXP-complete for tree-regular relations given by nondeterministic or deterministic top-down tree automata, and P-complete for transitive or co-transitive tree-regular relations or relations given by deterministic bottom-up tree automata. As applications we identified liveness in form of recurrent reachability with generalized Büchi condition and the recognizability problem. More precisely, we were able to show that recurrent reachability with generalized Büchi condition is PSPACE-complete (resp. EXP-complete) for (tree-)regular relations. Moreover, we showed that tree-recognizability of tree-regular relations is P-complete for deterministic tree automata and EXP-complete for nondeterministic tree automata.

For linear integer/real arithmetic we showed that the Ramsey quantifier can be eliminated from an existential formula in polynomial time, yielding an existential formula of linear size. This has several applications to linear liveness in systems with counters and/or clocks. In particular, we proved linear liveness to be NP-complete for succinct one-counter automata, reversal-bounded counter machines with one unrestricted counter, continuous vector addition systems with states, and Parikh automata. In the context of well-structured transition systems we proved that checking whether a quantifier-free Presburger formula defines a well-quasi-ordering is coNP-complete. As a further application we could show that monadic decomposability is coNP-complete for quantifier-free formulas in LIA, LRA, and LIRA.

Finally, we considered membership problems in hierarchies of classes of relations over finite and infinite words. Over infinite words we showed that the infinite clique problem for ω-regular co-equivalence relations is NL-complete. This implies that ω-recognizability is NL-complete or PSPACE-complete depending on whether the ω-regular relation is given by a deterministic parity or nondeterministic Büchi automaton, respectively. Thus, the complexity is the same as in the finite-word case. Over finite words we furthermore considered recognizability in a more general class than the class of regular relations. We proved that recognizability of deterministic rational relations is in P for binary relations, in coREXP for relations of fixed arity greater than two, and in coNEXP if the arity is

part of the input. Using a reduction, we showed that this result implies that regularity of deterministic rational relations is in P for binary relations and in $(2k-4)$-EXP for relations of arity $k > 2$.

Future work In Chapter 4 we showed that the Ramsey quantifier can be evaluated in automatic structures. This, however, does not imply that the Ramsey quantifier can be eliminated in this structure. In Chapter 5 we saw that Presburger arithmetic, which is an automatic structure, admits elimination of the Ramsey quantifier. Another example where the Ramsey quantifier can be eliminated is *Büchi arithmetic*, which is a universal automatic structure, i.e. every automatic structure is first-order interpretable in it. Because of that, the Ramsey quantifier applied to a formula in Büchi arithmetic can be eliminated by first computing the automaton for the relation defined by the formula, then applying Theorem 4.2.1, and finally computing again a formula in Büchi arithmetic for the resulting automaton. Note that without further insight on the structure of the automata this procedure is nonelementary.

Question. *Is there a more efficient elimination procedure for Büchi arithmetic similar to the one in Theorem 5.3.1 for Presburger arithmetic?*

Another question is whether the Ramsey quantifier can be eliminated in the existential fragment of Büchi arithmetic, which as shown in [HR21] is not expressively complete.

In Chapter 5 we observed that the Ramsey quantifier can be efficiently eliminated from existential formulas in common SMT theories (LIA, LRA, and LIRA). We leave it as future work to study Ramsey quantifiers in other SMT theories (like EUF or non-linear real arithmetic). In the case of non-linear real arithmetic, it is known that the Ramsey quantifier can be eliminated [Cow79], but the precise complexity is open.

We saw that one major application of the Ramsey quantifier is liveness verification. In the introduction we motivated that the Ramsey quantifier can be used to express a proof rule for termination (see Equation (1.2)). To formulate the proof rule in Equation (1.1) using *constrained horn clauses* (CHC), Grebenshchikov et al. [Gre+12] extend CHC with an ad-hoc condition for checking whether a relation is (disjunctively) well-founded. A more general extension, that would also allow to express more liveness properties, would be to allow the use of Ramsey quantifiers in CHC. An indicator that this might be worth studying is that the Ramsey quantifier can be eliminated from formulas in common SMT theories, as observed in Chapter 5.

Question. *Can the framework of CHC be extended with Ramsey quantifiers such that solutions can still be synthesized?*

In Chapter 6 we showed that the infinite clique problem for ω-regular co-equivalence relations is decidable in nondeterministic logspace. For arbitrary ω-regular relations decidability remains a major open problem.

Question. *Is the infinite clique problem decidable for general ω-regular relations?*

Even if restricted to transitive ω-regular relations, decidability is not known. This could, for example, be applied to liveness verification of infinite-state systems that have ω-regular reachability relations [BLW04a; BLW04b].

In Theorem 6.4.1 we proved that it is decidable to check whether a given deterministic rational relation is regular. This answers a question by Ibarra and Trân [IT12], who further asked whether the same problem for a relation given by an *unambiguous* automaton is decidable. Here, an unambiguous automaton is a generalization of a deterministic automaton where on every input there is at most one accepting run but possibly multiple non-accepting ones.

Question. *Is it decidable whether a relation given by an unambiguous automaton is regular?*

To the best of our knowledge, it is not even known whether the recognizability problem for this class of relations is decidable.

In Section 5.6.1 we considered the *decision* problem of monadic decomposability for quantifier-free formulas in linear integer/real arithmetic and proved coNP-completeness. It was shown by Hague et al. [Hag+20] that in case of quantifier-free Presburger arithmetic one can *compute* the monadic decomposition in exponential time if it exists. It was shown that the exponential blow-up is unavoidable if the output formulas are restricted to DNF/CNF. Such a result is not known if the restriction to DNF/CNF is lifted.

Question. *What is the complexity of computing a monadic decomoposition of formulas in LRA and LIRA if one exists? How large are the decompositions when restricted to DNF/CNF or not restricted at all?*

Similarly, for relations on words one can consider the problem of computing an independent multitape automaton if the relation turns out to be recognizable. In Theorem 6.3.9 we proved that for deterministic rational relations this is possible in double exponential time, which we showed to be optimal. If the relation is regular and given by an automaton, then we cannot just first convert the automaton to a deterministic multitape automaton since this would incur an exponential blow-up.

Question. *What is the precise complexity of computing an independent multitape automaton for a regular relation given by an automaton if the relation is recognizable?*

Finally, we note that the implementation [Ber+23] of the algorithms developed in Chapter 5 is only a prototype. For example, it does not yet incorporate the SMT-LIB format [BFT16], which could be extended to allow the use of Ramsey quantifiers. Furthermore, there is no implementation yet of the algorithms in Chapter 4 for the evaluation of Ramsey quantifiers over (tree-)regular relations.

Bibliography

[AAS12] P. A. Abdulla, M. F. Atig, and J. Stenman. "Dense-Timed Pushdown Automata". In: *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*. IEEE Computer Society, 2012, pp. 35–44. DOI: 10.1109/LICS.2012.15.

[AB09] S. Arora and B. Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. ISBN: 978-0-521-42426-4. DOI: 10.1017/CBO9780511804090.

[Abd+02] P. A. Abdulla, B. Jonsson, P. Mahata, and J. d'Orso. "Regular Tree Model Checking". In: *Computer Aided Verification, 14th International Conference, CAV 2002,Copenhagen, Denmark, July 27-31, 2002, Proceedings*. Ed. by E. Brinksma and K. G. Larsen. Vol. 2404. Lecture Notes in Computer Science. Springer, 2002, pp. 555–568. DOI: 10.1007/3-540-45657-0_47.

[Abd+04] P. A. Abdulla, B. Jonsson, M. Nilsson, and M. Saksena. "A Survey of Regular Model Checking". In: *CONCUR 2004 - Concurrency Theory, 15th International Conference, London, UK, August 31 - September 3, 2004, Proceedings*. Ed. by P. Gardner and N. Yoshida. Vol. 3170. Lecture Notes in Computer Science. Springer, 2004, pp. 35–48. DOI: 10.1007/978-3-540-28644-8_3.

[Abd+96] P. A. Abdulla, K. Cerans, B. Jonsson, and Y. Tsay. "General Decidability Theorems for Infinite-State Systems". In: *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*. IEEE Computer Society, 1996, pp. 313–321. DOI: 10.1109/LICS.1996.561359.

[AD94] R. Alur and D. L. Dill. "A Theory of Timed Automata". In: *Theor. Comput. Sci.* 126.2 (1994), pp. 183–235. DOI: 10.1016/0304-3975(94)90010-8.

[AK86] K. R. Apt and D. Kozen. "Limits for Automatic Verification of Finite-State Concurrent Systems". In: *Inf. Process. Lett.* 22.6 (1986), pp. 307–309. DOI: 10.1016/0020-0190(86)90071-2.

[AS87] B. Alpern and F. B. Schneider. "Recognizing Safety and Liveness". In: *Distributed Comput.* 2.3 (1987), pp. 117–126. DOI: 10.1007/BF01782772.

[ASU86] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley series in computer science / World student series edition. Addison-Wesley, 1986. ISBN: 0-201-10088-6. URL: https://www.worldcat.org/oclc/12285707.

[Bar+05] S. Bardin, A. Finkel, J. Leroux, and P. Schnoebelen. "Flat Acceleration in Symbolic Model Checking". In: *Automated Technology for Verification and Analysis, Third International Symposium, ATVA 2005, Taipei, Taiwan, October 4-7, 2005, Proceedings*. Ed. by D. A. Peled and Y. Tsay. Vol. 3707. Lecture Notes in Computer Science. Springer, 2005, pp. 474–488. DOI: 10.1007/11562948_35.

[Bar+19] P. Barceló, C. Hong, X. B. Le, A. W. Lin, and R. Niskanen. "Monadic Decomposability of Regular Relations". In: *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*. Ed. by C. Baier, I. Chatzigiannakis, P. Flocchini, and S. Leonardi. Vol. 132. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 103:1–103:14. DOI: 10.4230/LIPIcs.ICALP.2019.103.

[Bar+22] H. Barbosa, C. W. Barrett, M. Brain, G. Kremer, H. Lachnitt, M. Mann, A. Mohamed, M. Mohamed, A. Niemetz, A. Nötzli, A. Ozdemir, M. Preiner, A. Reynolds, Y. Sheng, C. Tinelli, and Y. Zohar. "cvc5: A Versatile and Industrial-Strength SMT Solver". In: *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I*. Ed. by D. Fisman and G. Rosu. Vol. 13243. Lecture Notes in Computer Science. Springer, 2022, pp. 415–442. DOI: 10.1007/978-3-030-99524-9_24.

[Bau77] A. Baudisch. "Decidability of the theory of abelian groups with Ramsey quantifiers". In: *Bulletin de l'Académie Polonaise des Sciences. Série des Sciences Mathématiques, Astronomiques et Physiques* 25 (1977), pp. 733–739.

[Bau84] A. Baudisch. "Magidor-Malitz Quantifiers in Modules". In: *J. Symb. Log.* 49.1 (1984), pp. 1–8. DOI: 10.2307/2274085.

[BEM97] A. Bouajjani, J. Esparza, and O. Maler. "Reachability Analysis of Pushdown Automata: Application to Model-Checking". In: *CONCUR '97: Concurrency Theory, 8th International Conference, Warsaw, Poland, July 1-4, 1997, Proceedings*. Ed. by A. W. Mazurkiewicz and J. Winkowski. Vol. 1243. Lecture Notes in Computer Science. Springer, 1997, pp. 135–150. DOI: 10.1007/3-540-63141-0_10.

[Ben12] M. Ben-Ari. *Mathematical Logic for Computer Science, 3rd Edition*. Springer, 2012. ISBN: 978-1-4471-4128-0. DOI: 10.1007/978-1-4471-4129-7.

[Ber+22] P. Bergsträßer, M. Ganardi, A. W. Lin, and G. Zetzsche. "Ramsey Quantifiers over Automatic Structures: Complexity and Applications to Verification". In: *37th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2022, Haifa, Israel, August 2-5, 2022*. Ed. by C. Baier and D. Fisman. ACM, 2022, 28:1–28:14. DOI: 10.1145/3531130.3533346.

[Ber+23] P. Bergsträßer, M. Ganardi, A. W. Lin, and G. Zetzsche. *Ramsey Quantifiers in Linear Arithmetics - Artifact*. Oct. 2023. DOI: 10.5281/zenodo.8422415.

[Ber+24] P. Bergsträßer, M. Ganardi, A. W. Lin, and G. Zetzsche. "Ramsey Quantifiers in Linear Arithmetics". In: *Proceedings of the ACM on Programming Languages* 8.POPL (2024), pp. 1–32. DOI: 10.1145/3632843.

[BER94] A. Bouajjani, R. Echahed, and R. Robbana. "On the Automatic Verification of Systems with Continuous Variables and Unbounded Discrete Data Structures". In: *Hybrid Systems II, Proceedings of the Third International Workshop on Hybrid Systems, Ithaca, NY, USA, October 1994*. Ed. by P. J. Antsaklis, W. Kohn, A. Nerode, and S. Sastry. Vol. 999. Lecture Notes in Computer Science. Springer, 1994, pp. 64–85. DOI: 10.1007/3-540-60472-3_4.

[BFT16] C. Barrett, P. Fontaine, and C. Tinelli. *The Satisfiability Modulo Theories Library (SMT-LIB)*. https://SMT-LIB.org. 2016.

[BG00] A. Blumensath and E. Grädel. "Automatic Structures". In: *15th Annual IEEE Symposium on Logic in Computer Science, Santa Barbara, California, USA, June 26-29, 2000*. IEEE Computer Society, 2000, pp. 51–62. DOI: 10.1109/LICS.2000.855755.

[BG04] A. Blumensath and E. Grädel. "Finite Presentations of Infinite Structures: Automata and Interpretations". In: *Theory Comput. Syst.* 37.6 (2004), pp. 641–674. DOI: 10.1007/s00224-004-1133-y.

[BG23] P. Bergsträßer and M. Ganardi. "Revisiting Membership Problems in Subclasses of Rational Relations". In: *38th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2023, Boston, MA, USA, June 26-29, 2023*. IEEE, 2023, pp. 1–14. DOI: 10.1109/LICS56636.2023.10175722.

[BH06] B. Boigelot and F. Herbreteau. "The Power of Hybrid Acceleration". In: *Computer Aided Verification, 18th International Conference, CAV 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*. Ed. by T. Ball and R. B. Jones. Vol. 4144. Lecture Notes in Computer Science. Springer, 2006, pp. 438–451. DOI: 10.1007/11817963_40.

[BH17] M. Blondin and C. Haase. "Logics for continuous reachability in Petri nets and vector addition systems with states". In: *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*. IEEE Computer Society, 2017, pp. 1–12. DOI: 10.1109/LICS.2017.8005068.

[BHV04] A. Bouajjani, P. Habermehl, and T. Vojnar. "Abstract Regular Model Checking". In: *Computer Aided Verification, 16th International Conference, CAV 2004, Boston, MA, USA, July 13-17, 2004, Proceedings*. Ed. by R. Alur and D. A. Peled. Vol. 3114. Lecture Notes in Computer Science. Springer, 2004, pp. 372–386. DOI: 10.1007/978-3-540-27813-9_29.

[BK80] J. Baldwin and D. Kueker. "Ramsey quantifiers and the finite cover property". In: *Pacific Journal of Mathematics* 90.1 (1980), pp. 11–19. DOI: 10.2140/pjm.1980.90.11.

[Blo+16] M. Blondin, A. Finkel, C. Haase, and S. Haddad. "Approaching the Coverability Problem Continuously". In: *Tools and Algorithms for the Construction and Analysis of Systems - 22nd International Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*. Ed. by M. Chechik and J. Raskin. Vol. 9636. Lecture Notes in Computer Science. Springer, 2016, pp. 480–496. DOI: 10.1007/978-3-662-49674-9_28.

[BLW03] B. Boigelot, A. Legay, and P. Wolper. "Iterating Transducers in the Large (Extended Abstract)". In: *Computer Aided Verification, 15th International Conference, CAV 2003, Boulder, CO, USA, July 8-12, 2003, Proceedings*. Ed. by W. A. H. Jr. and F. Somenzi. Vol. 2725. Lecture Notes in Computer Science. Springer, 2003, pp. 223–235. DOI: 10.1007/978-3-540-45069-6_24.

[BLW04a] B. Boigelot, A. Legay, and P. Wolper. "Omega-Regular Model Checking". In: *Tools and Algorithms for the Construction and Analysis of Systems, 10th International Conference, TACAS 2004, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2004, Barcelona, Spain, March 29 - April 2, 2004, Proceedings*. Ed. by K. Jensen and A. Podelski. Vol. 2988. Lecture Notes in Computer Science. Springer, 2004, pp. 561–575. DOI: 10.1007/978-3-540-24730-2_41.

[BLW04b] A. Bouajjani, A. Legay, and P. Wolper. "Handling Liveness Properties in (*omega*-)Regular Model Checking". In: *Proceedings of the 6th International Workshop on Verification of Infinite-State Systems, INFINITY 2004, London, UK, September 4, 2004*. Ed. by J. C. Bradfield and F. Moller. Vol. 138. Electronic Notes in Theoretical Computer Science 3. Elsevier, 2004, pp. 101–115. DOI: 10.1016/J.ENTCS.2005.02.061.

[Bou+00] A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. "Regular Model Checking". In: *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings*. Ed. by E. A. Emerson and A. P. Sistla. Vol. 1855. Lecture Notes in Computer Science. Springer, 2000, pp. 403–418. DOI: 10.1007/10722167_31.

[Bou+11] A. Bouajjani, M. Bozga, P. Habermehl, R. Iosif, P. Moro, and T. Vojnar. "Programs with lists are counter automata". In: *Formal Methods Syst. Des.* 38.2 (2011), pp. 158–192. DOI: 10.1007/s10703-011-0111-7.

[Bou+12] A. Bouajjani, P. Habermehl, A. Rogalewicz, and T. Vojnar. "Abstract regular (tree) model checking". In: *Int. J. Softw. Tools Technol. Transf.* 14.2 (2012), pp. 167–191. DOI: 10.1007/S10009-011-0205-Y.

[BPR13] T. A. Beyene, C. Popeea, and A. Rybalchenko. "Solving Existentially Quantified Horn Clauses". In: *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*. Ed. by N. Sharygina and H. Veith. Vol. 8044. Lecture Notes in Computer Science. Springer, 2013, pp. 869–882. DOI: 10.1007/978-3-642-39799-8_61.

[BT12] A. Bouajjani and T. Touili. "Widening techniques for regular tree model checking". In: *Int. J. Softw. Tools Technol. Transf.* 14.2 (2012), pp. 145–165. DOI: 10.1007/S10009-011-0208-8.

[BT76] I. Borosh and L. B. Treybig. "Bounds on positive integral solutions of linear Diophantine equations". In: *Proceedings of the American Mathematical Society* 55.2 (1976), pp. 299–304. DOI: 10.2307/2041711.

[Bur+90] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. "Symbolic Model Checking: 10^{20} States and Beyond". In: *Proceedings of the Fifth Annual Symposium on Logic in Computer Science (LICS '90), Philadelphia, Pennsylvania, USA, June 4-7, 1990*. IEEE Computer Society, 1990, pp. 428–439. DOI: 10.1109/LICS.1990.113767.

[Cau92] D. Caucal. "On the Regular Structure of Prefix Rewriting". In: *Theor. Comput. Sci.* 106.1 (1992), pp. 61–86. DOI: 10.1016/0304-3975(92)90278-N.

[CCG06] O. Carton, C. Choffrut, and S. Grigorieff. "Decision problems among the main subfamilies of rational relations". In: *RAIRO Theor. Informatics Appl.* 40.2 (2006), pp. 255–275. DOI: 10.1051/ita:2006005.

[CFG03] Y. Chen, J. Flum, and M. Grohe. "Bounded Nondeterminism and Alternation in Parameterized Complexity Theory". In: *18th Annual IEEE Conference on Computational Complexity (Complexity 2003), 7-10 July 2003, Aarhus, Denmark*. IEEE Computer Society, 2003, pp. 13–29. DOI: 10.1109/CCC.2003.1214407.

[Cho+03] J. Chomicki, D. Q. Goldin, G. M. Kuper, and D. Toman. "Variable Independence in Constraint Databases". In: *IEEE Trans. Knowl. Data Eng.* 15.6 (2003), pp. 1422–1436. DOI: 10.1109/TKDE.2003.1245282.

[CJ99] H. Comon and Y. Jurski. "Timed Automata and the Theory of Real Numbers". In: *CONCUR '99: Concurrency Theory, 10th International Conference, Eindhoven, The Netherlands, August 24-27, 1999, Proceedings*. Ed. by J. C. M. Baeten and S. Mauw. Vol. 1664. Lecture Notes in Computer Science. Springer, 1999, pp. 242–257. DOI: 10.1007/3-540-48320-9_18.

[CK90] C. C. Chang and H. J. Keisler. *Model Theory*. Elsevier, 1990. ISBN: 9780444558312. URL: https://shop.elsevier.com/books/model-theory/chang/978-0-444-88054-3.

[CKL01] S. S. Cosmadakis, G. M. Kuper, and L. Libkin. "On the orthographic dimension of definable sets". In: *Inf. Process. Lett.* 79.3 (2001), pp. 141–145. DOI: 10.1016/S0020-0190(00)00184-8.

[CL07] T. Colcombet and C. Löding. "Transforming structures by set interpretations". In: *Log. Methods Comput. Sci.* 3.2 (2007). DOI: 10.2168/LMCS-3(2:4)2007.

[CL15] L. Clemente and S. Lasota. "Timed Pushdown Automata Revisited". In: *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*. IEEE Computer Society, 2015, pp. 738–749. DOI: 10.1109/LICS.2015.73.

[CL18] L. Clemente and S. Lasota. "Binary Reachability of Timed Pushdown Automata via Quantifier Elimination and Cyclic Order Atoms". In: *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*. Ed. by I. Chatzigiannakis, C. Kaklamanis, D. Marx, and D. Sannella. Vol. 107. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 118:1–118:14. DOI: 10.4230/LIPICS.ICALP.2018.118.

[CMS24] D. Chistikov, A. Mansutti, and M. R. Starchak. "Integer Linear-Exponential Programming in NP by Quantifier Elimination". In: *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*. Ed. by K. Bringmann, M. Grohe, G. Puppis, and O. Svensson. Vol. 297. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 132:1–132:20. DOI: 10.4230/LIPICS.ICALP.2024.132.

[CO21] W. Czerwinski and L. Orlikowski. "Reachability in Vector Addition Systems is Ackermann-complete". In: *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*. IEEE, 2021, pp. 1229–1240. DOI: 10.1109/FOCS52979.2021.00120.

[Com+08] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, C. Löding, S. Tison, and M. Tommasi. *Tree Automata Techniques and Applications*. 2008, p. 262. URL: https://inria.hal.science/hal-03367725.

[Coo72] S. A. Cook. "A Hierarchy for Nondeterministic Time Complexity". In: *Proceedings of the 4th Annual ACM Symposium on Theory of Computing, May 1-3, 1972, Denver, Colorado, USA*. Ed. by P. C. Fischer, H. P. Zeiger, J. D. Ullman, and A. L. Rosenberg. ACM, 1972, pp. 187–192. DOI: 10.1145/800152.804913.

[Cow79] J. R. Cowles. "The theory of Archimedean real closed fields in logics with Ramsey quantifiers". In: *Fundamenta Mathematicae* 103.1 (1979), pp. 65–76. URL: http://eudml.org/doc/211045.

[CPR11] B. Cook, A. Podelski, and A. Rybalchenko. "Proving program termination". In: *Commun. ACM* 54.5 (2011), pp. 88–98. DOI: 10.1145/1941487.1941509.

[Dan+00] Z. Dang, O. H. Ibarra, T. Bultan, R. A. Kemmerer, and J. Su. "Binary Reachability Analysis of Discrete Pushdown Timed Automata". In: *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings*. Ed. by E. A. Emerson and A. P. Sistla. Vol. 1855. Lecture Notes in Computer Science. Springer, 2000, pp. 69–84. DOI: 10.1007/10722167_9.

[Dan01] Z. Dang. "Binary Reachability Analysis of Pushdown Timed Automata with Dense Clocks". In: *Computer Aided Verification, 13th International Conference, CAV 2001, Paris, France, July 18-22, 2001, Proceedings*. Ed. by G. Berry, H. Comon, and A. Finkel. Vol. 2102. Lecture Notes in Computer Science. Springer, 2001, pp. 506–518. DOI: 10.1007/3-540-44585-4_48.

[DF99] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999. ISBN: 978-1-4612-6798-0. DOI: 10.1007/978-1-4612-0515-9.

[DI02] Z. Dang and O. H. Ibarra. "The Existence of ω-Chains for Transitive Mixed Linear Relations and Its Applications". In: *Int. J. Found. Comput. Sci.* 13.6 (2002), pp. 911–936. DOI: 10.1142/S0129054102001539.

[DLS02] D. Dams, Y. Lakhnech, and M. Steffen. "Iterating transducers". In: *J. Log. Algebraic Methods Program.* 52-53 (2002), pp. 109–127. DOI: 10.1016/S1567-8326(02)00025-5.

[DSK01] Z. Dang, P. San Pietro, and R. A. Kemmerer. "On Presburger Liveness of Discrete Timed Automata". In: *STACS 2001, 18th Annual Symposium on Theoretical Aspects of Computer Science, Dresden, Germany, February 15-17, 2001, Proceedings*. Ed. by A. Ferreira and H. Reichel. Vol. 2010. Lecture Notes in Computer Science. Springer, 2001, pp. 132–143. DOI: 10.1007/3-540-44693-1_12.

[EM65] C. C. Elgot and J. E. Mezei. "On Relations Defined by Generalized Finite Automata". In: *IBM J. Res. Dev.* 9.1 (1965), pp. 47–68. DOI: 10.1147/rd.91.0047.

[End01] H. B. Enderton. *A mathematical introduction to logic*. Second. Academic Press, 2001. ISBN: 978-0-12-238452-3. DOI: 10.1016/C2009-0-22107-6.

[Eng99] J. Engelfriet. "Derivation Trees of Ground Term Rewriting Systems". In: *Inf. Comput.* 152.1 (1999), pp. 1–15. DOI: 10.1006/INCO.1998.2786.

[ES01] J. Esparza and S. Schwoon. "A BDD-Based Model Checker for Recursive Programs". In: *Computer Aided Verification, 13th International Conference, CAV 2001, Paris, France, July 18-22, 2001, Proceedings*. Ed. by G. Berry, H. Comon, and A. Finkel. Vol. 2102. Lecture Notes in Computer Science. Springer, 2001, pp. 324–336. DOI: 10.1007/3-540-44585-4_30.

[Esp+00] J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon. "Efficient Algorithms for Model Checking Pushdown Systems". In: *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings*. Ed. by E. A. Emerson and A. P. Sistla. Vol. 1855. Lecture Notes in Computer Science. Springer, 2000, pp. 232–247. DOI: 10.1007/10722167_20.

[FG06] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. ISBN: 978-3-540-29952-3. DOI: 10.1007/3-540-29953-X.

[FG19a] A. Finkel and E. Gupta. "The Well Structured Problem for Presburger Counter Machines". In: *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2019, December 11-13, 2019, Bombay, India*. Ed. by A. Chattopadhyay and P. Gastin. Vol. 150. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 41:1–41:15. DOI: 10.4230/LIPICS.FSTTCS.2019.41.

[FG19b] A. Finkel and E. Gupta. "The Well Structured Problem for Presburger Counter Machines". In: *CoRR* abs/1910.02736 (2019). DOI: 10.48550/arXiv.1910.02736.

[FG78] E. P. Friedman and S. A. Greibach. "On Equivalence and Subclass Containment Problems for Deterministic Context-Free Languages". In: *Inf. Process. Lett.* 7.6 (1978), pp. 287–290. DOI: 10.1016/0020-0190(78)90019-4.

[FG82] E. P. Friedman and S. A. Greibach. "A Polynomial Time Algorithm for Deciding the Equivalence Problem for 2-Tape Deterministic Finite State Acceptors". In: *SIAM J. Comput.* 11.1 (1982), pp. 166–183. DOI: 10.1137/0211013.

[Fou26] J. B. J. Fourier. "Solution d'une question particuliere du calcul des inégalités". In: *Nouveau Bulletin des Sciences par la Société philomatique de Paris* 99 (1826).

[FR68] P. C. Fischer and A. L. Rosenberg. "Multitape One-Way Nonwriting Automata". In: *J. Comput. Syst. Sci.* 2.1 (1968), pp. 88–101. DOI: 10.1016/S0022-0000(68)80006-6.

[FR75] J. Ferrante and C. Rackoff. "A Decision Procedure for the First Order Theory of Real Addition with Order". In: *SIAM J. Comput.* 4.1 (1975), pp. 69–76. DOI: 10.1137/0204006.

[Frü+91] T. W. Frühwirth, E. Shapiro, M. Y. Vardi, and E. Yardeni. "Logic Programs as Types for Logic Programs". In: *Proceedings of the Sixth Annual Symposium on Logic in Computer Science (LICS '91), Amsterdam, The Netherlands, July 15-18, 1991*. IEEE Computer Society, 1991, pp. 300–309. DOI: 10.1109/LICS.1991.151654.

[FS01] A. Finkel and P. Schnoebelen. "Well-structured transition systems everywhere!" In: *Theor. Comput. Sci.* 256.1-2 (2001), pp. 63–92. DOI: 10.1016/S0304-3975(00)00102-X.

[FS93] C. Frougny and J. Sakarovitch. "Synchronized Rational Relations of Finite and Infinite Words". In: *Theor. Comput. Sci.* 108.1 (1993), pp. 45–82. DOI: 10.1016/0304-3975(93)90230-Q.

[FWW97] A. Finkel, B. Willems, and P. Wolper. "A direct symbolic approach to model checking pushdown systems". In: *Second International Workshop on Verification of Infinite State Systems, Infinity 1997, Bologna, Italy, July 11-12, 1997*. Ed. by F. Moller. Vol. 9. Electronic Notes in Theoretical Computer Science. Elsevier, 1997, pp. 27–37. DOI: 10.1016/S1571-0661(05)80426-8.

[Got+05] G. Gottlob, C. Koch, R. Pichler, and L. Segoufin. "The complexity of XPath query evaluation and XML typing". In: *J. ACM* 52.2 (2005), pp. 284–335. DOI: 10.1145/1059513.1059520.

[Grä20] E. Grädel. "Automatic Structures: Twenty Years Later". In: *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*. Ed. by H. Hermanns, L. Zhang, N. Kobayashi, and D. Miller. ACM, 2020, pp. 21–34. DOI: 10.1145/3373718.3394734.

[Gre+12] S. Grebenshchikov, N. P. Lopes, C. Popeea, and A. Rybalchenko. "Synthesizing software verifiers from proof rules". In: *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '12, Beijing, China - June 11 - 16, 2012*. Ed. by J. Vitek, H. Lin, and F. Tip. ACM, 2012, pp. 405–416. DOI: 10.1145/2254064.2254112.

[GRS99] S. Grumbach, P. Rigaux, and L. Segoufin. "On the Orthographic Dimension of Constraint Databases". In: *Database Theory - ICDT '99, 7th International Conference, Jerusalem, Israel, January 10-12, 1999, Proceedings*. Ed. by C. Beeri and P. Buneman. Vol. 1540. Lecture Notes in Computer Science. Springer, 1999, pp. 199–216. DOI: 10.1007/3-540-49257-7_14.

[GS66a] S. Ginsburg and E. Spanier. "Semigroups, Presburger formulas, and languages". In: *Pacific journal of Mathematics* 16.2 (1966), pp. 285–296. DOI: 10.2140/pjm.1966.16.285.

[GS66b] S. Ginsburg and E. H. Spanier. "Bounded regular sets". In: *Proceedings of the American Mathematical Society* 17.5 (1966), pp. 1043–1049. DOI: 10.1090/S0002-9939-1966-0201310-3.

[GS78] J. von zur Gathen and M. Sieveking. "A bound on solutions of linear integer equalities and inequalities". In: *Proceedings of the American Mathematical Society* 72.1 (1978), pp. 155–158. DOI: 10.2307/2042554.

[GTW02] E. Grädel, W. Thomas, and T. Wilke, eds. *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*. Vol. 2500. Lecture Notes in Computer Science. Springer, 2002. ISBN: 3-540-00388-6. DOI: 10.1007/3-540-36387-4.

[Guh+22] S. Guha, I. Jecker, K. Lehtinen, and M. Zimmermann. "Parikh Automata over Infinite Words". In: *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2022, December 18-20, 2022, IIT Madras, Chennai, India*. Ed. by A. Dawar and V. Guruswami. Vol. 250. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 40:1–40:20. DOI: 10.4230/LIPICS.FSTTCS.2022.40.

[Haa+09] C. Haase, S. Kreutzer, J. Ouaknine, and J. Worrell. "Reachability in Succinct and Parametric One-Counter Automata". In: *CONCUR 2009 - Concurrency Theory, 20th International Conference, CONCUR 2009, Bologna, Italy, September 1-4, 2009. Proceedings*. Ed. by M. Bravetti and G. Zavattaro. Vol. 5710. Lecture Notes in Computer Science. Springer, 2009, pp. 369–383. DOI: 10.1007/978-3-642-04081-8_25.

[Haa+24] C. Haase, S. N. Krishna, K. Madnani, O. S. Mishra, and G. Zetzsche. "An Efficient Quantifier Elimination Procedure for Presburger Arithmetic". In: *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*. Ed. by K. Bringmann, M. Grohe, G. Puppis, and O. Svensson. Vol. 297. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 142:1–142:17. DOI: 10.4230/LIPICS.ICALP.2024.142.

[Hag+20] M. Hague, A. W. Lin, P. Rümmer, and Z. Wu. "Monadic Decomposition in Integer Linear Arithmetic". In: *Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Paris, France, July 1-4, 2020, Proceedings, Part I*. Ed. by N. Peltier and V. Sofronie-Stokkermans. Vol. 12166. Lecture Notes in Computer Science. Springer, 2020, pp. 122–140. DOI: 10.1007/978-3-030-51074-9_8.

[Hjo+08] G. Hjorth, B. Khoussainov, A. Montalbán, and A. Nies. "From Automatic Structures to Borel Structures". In: *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*. IEEE Computer Society, 2008, pp. 431–441. DOI: 10.1109/LICS.2008.28.

[HK91] T. Harju and J. Karhumäki. "The Equivalence Problem of Multitape Finite Automata". In: *Theor. Comput. Sci.* 78.2 (1991), pp. 347–355. DOI: 10.1016/0304-3975(91)90356-7.

[HL11] M. Hague and A. W. Lin. "Model Checking Recursive Programs with Numeric Data Types". In: *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*. Ed. by G. Gopalakrishnan and S. Qadeer. Vol. 6806. Lecture Notes in

Computer Science. Springer, 2011, pp. 743–759. DOI: 10.1007/978-3-642-22110-1_60.

[Hod76] B. R. Hodgson. "Théories décidables par automate fini". PhD thesis. Université de Montréal, 1976.

[HR21] C. Haase and J. Rózycki. "On the Expressiveness of Büchi Arithmetic". In: *Foundations of Software Science and Computation Structures - 24th International Conference, FOSSACS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings*. Ed. by S. Kiefer and C. Tasson. Vol. 12650. Lecture Notes in Computer Science. Springer, 2021, pp. 310–323. DOI: 10.1007/978-3-030-71995-1_16.

[HS65] J. Hartmanis and R. E. Stearns. "On the computational complexity of algorithms". In: *Transactions of the American Mathematical Society* 117 (1965), pp. 285–306. DOI: 10.1090/S0002-9947-1965-0170805-7.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979. ISBN: 0-201-02988-X.

[Iba+00] O. H. Ibarra, J. Su, Z. Dang, T. Bultan, and R. A. Kemmerer. "Counter Machines: Decidable Properties and Applications to Verification Problems". In: *Mathematical Foundations of Computer Science 2000, 25th International Symposium, MFCS 2000, Bratislava, Slovakia, August 28 - September 1, 2000, Proceedings*. Ed. by M. Nielsen and B. Rovan. Vol. 1893. Lecture Notes in Computer Science. Springer, 2000, pp. 426–435. DOI: 10.1007/3-540-44612-5_38.

[Iba78] O. H. Ibarra. "Reversal-Bounded Multicounter Machines and Their Decision Problems". In: *J. ACM* 25.1 (1978), pp. 116–133. DOI: 10.1145/322047.322058.

[Imm88] N. Immerman. "Nondeterministic Space is Closed Under Complementation". In: *SIAM J. Comput.* 17.5 (1988), pp. 935–938. DOI: 10.1137/0217058.

[IT12] O. H. Ibarra and N. Q. Trân. "On synchronized multi-tape and multi-head automata". In: *Theor. Comput. Sci.* 449 (2012), pp. 74–84. DOI: 10.1016/J.TCS.2012.04.006.

[JN00] B. Jonsson and M. Nilsson. "Transitive Closures of Regular Relations for Verifying Infinite-State Systems". In: *Tools and Algorithms for Construction and Analysis of Systems, 6th International Conference, TACAS 2000, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS 2000, Berlin, Germany, March 25 - April 2, 2000, Proceedings*. Ed. by S. Graf and M. I. Schwartzbach. Vol. 1785. Lecture Notes in Computer Science. Springer, 2000, pp. 220–234. DOI: 10.1007/3-540-46419-0_16.

[Kar11] A. Kartzow. "First-Order Model Checking on Generalisations of Pushdown Graphs". PhD thesis. Technische Universität Darmstadt, 2011. URL: http://tuprints.ulb.tu-darmstadt.de/2681/.

[KKS15] P. Karandikar, M. Kufleitner, and P. Schnoebelen. "On the index of Simon's congruence for piecewise testability". In: *Inf. Process. Lett.* 115.4 (2015), pp. 515–519. DOI: 10.1016/j.ipl.2014.11.008.

[KKV01] V. King, O. Kupferman, and M. Y. Vardi. "On the Complexity of Parity Word Automata". In: *Foundations of Software Science and Computation Structures, 4th International Conference, FOSSACS 2001 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2001 Genova, Italy, April 2-6, 2001, Proceedings*. Ed. by F. Honsell and M. Miculan. Vol. 2030. Lecture Notes in Computer Science. Springer, 2001, pp. 276–286. DOI: 10.1007/3-540-45315-6_18.

[KL08] D. Kuske and M. Lohrey. "First-order and counting theories of omega-automatic structures". In: *J. Symb. Log.* 73.1 (2008), pp. 129–150. DOI: 10.2178/JSL/1208358745.

[KL10] D. Kuske and M. Lohrey. "Some natural decision problems in automatic graphs". In: *J. Symb. Log.* 75.2 (2010), pp. 678–710. DOI: 10.2178/jsl/1268917499.

[KN94] B. Khoussainov and A. Nerode. "Automatic Presentations of Structures". In: *Logical and Computational Complexity. Selected Papers. Logic and Computational Complexity, International Workshop LCC '94, Indianapolis, Indiana, USA, 13-16 October 1994*. Ed. by D. Leivant. Vol. 960. Lecture Notes in Computer Science. Springer, 1994, pp. 367–392. DOI: 10.1007/3-540-60178-3_93.

[Kőn27] D. Kőnig. "Über eine Schlussweise aus dem Endlichen ins Unendliche". In: *Acta Sci. Math.(Szeged)* 3.2-3 (1927), pp. 121–130.

[Koz77] D. Kozen. "Lower Bounds for Natural Proof Systems". In: *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*. IEEE Computer Society, 1977, pp. 254–266. DOI: 10.1109/SFCS.1977.16.

[Koz97] D. Kozen. *Automata and computability*. Undergraduate texts in computer science. Springer, 1997. ISBN: 978-0-387-94907-9. DOI: 10.1007/978-1-4612-1844-9.

[KR03] F. Klaedtke and H. Rueß. "Monadic Second-Order Logics with Cardinalities". In: *Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003. Proceedings*. Ed. by J. C. M. Baeten, J. K. Lenstra, J. Parrow, and G. J. Woeginger. Vol. 2719. Lecture Notes in Computer Science. Springer, 2003, pp. 681–696. DOI: 10.1007/3-540-45061-0_54.

[KR83] H. A. Kierstead and J. B. Remmel. "Indiscernibles and Decidable Models". In: *J. Symb. Log.* 48.1 (1983), pp. 21–32. DOI: 10.2307/2273316.

[Kus10] D. Kuske. "Is Ramsey's Theorem omega-automatic?" In: *27th International Symposium on Theoretical Aspects of Computer Science, STACS 2010, March 4-6, 2010, Nancy, France*. Ed. by J. Marion and T. Schwentick. Vol. 5. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010, pp. 537–548. DOI: 10.4230/LIPIcs.STACS.2010.2483.

[KV14] K. Korovin and M. Veanes. "Skolemization Modulo Theories". In: *Mathematical Software - ICMS 2014 - 4th International Congress, Seoul, South Korea, August 5-9, 2014. Proceedings*. Ed. by H. Hong and C. Yap. Vol. 8592. Lecture Notes in Computer Science. Springer, 2014, pp. 303–306. DOI: 10.1007/978-3-662-44199-2_47.

[Leg12] A. Legay. "Extrapolating (omega-)regular model checking". In: *Int. J. Softw. Tools Technol. Transf.* 14.2 (2012), pp. 119–143. DOI: 10.1007/S10009-011-0209-7.

[Ler21] J. Leroux. "The Reachability Problem for Petri Nets is Not Primitive Recursive". In: *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*. IEEE, 2021, pp. 1241–1252. DOI: 10.1109/FOCS52979.2021.00121.

[Li+20] X. Li, T. Chen, Z. Wu, and M. Xia. "Computing Linear Arithmetic Representation of Reachability Relation of One-Counter Automata". In: *Dependable Software Engineering. Theories, Tools, and Applications - 6th International Symposium, SETTA 2020, Guangzhou, China, November 24-27, 2020, Proceedings*. Ed. by J. Pang and L. Zhang. Vol. 12153. Lecture Notes in Computer Science. Springer, 2020, pp. 89–107. DOI: 10.1007/978-3-030-62822-2_6.

[Lib03] L. Libkin. "Variable independence for first-order definable constraints". In: *ACM Trans. Comput. Log.* 4.4 (2003), pp. 431–451. DOI: 10.1145/937555.937557.

[Lib05] L. Libkin. "Logics for Unranked Trees: An Overview". In: *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*. Ed. by L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung. Vol. 3580. Lecture Notes in Computer Science. Springer, 2005, pp. 35–50. DOI: 10.1007/11523468_4.

[Lin12] A. W. Lin. "Accelerating tree-automatic relations". In: *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India*. 2012, pp. 313–324. DOI: 10.4230/LIPIcs.FSTTCS.2012.313.

[Lip76] R. Lipton. "The Reachability Problem Requires Exponential Space". In: *Yale University, Department of Computer Science, Report* 63 (1976). URL: https://www.cs.yale.edu/publications/techreports/tr63.pdf.

[Lis79] L. P. Lisovik. "The identity problem for regular events over the direct product of free and cyclic semigroups". In: *Dok. Akad. Nauk USSR* 6 (1979), pp. 410–413.

[Löd06] C. Löding. "Reachability Problems on Regular Ground Tree Rewriting Graphs". In: *Theory Comput. Syst.* 39.2 (2006), pp. 347–383. DOI: 10.1007/s00224-004-1170-6.

[LS05] D. Lugiez and P. Schnoebelen. "Decidable first-order transition logics for PA-processes". In: *Inf. Comput.* 203.1 (2005), pp. 75–113. DOI: 10.1016/J.IC.2005.02.003.

[LS07] C. Löding and A. Spelten. "Transition Graphs of Rewriting Systems over Unranked Trees". In: *Mathematical Foundations of Computer Science 2007, 32nd International Symposium, MFCS 2007, Ceský Krumlov, Czech Republic, August 26-31, 2007, Proceedings*. Ed. by L. Kucera and A. Kucera. Vol. 4708. Lecture Notes in Computer Science. Springer, 2007, pp. 67–77. DOI: 10.1007/978-3-540-74456-6_8.

[LS19a] J. Leroux and S. Schmitz. "Reachability in Vector Addition Systems is Primitive-Recursive in Fixed Dimension". In: *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*. IEEE, 2019, pp. 1–13. DOI: 10.1109/LICS.2019.8785796.

[LS19b] C. Löding and C. Spinrath. "Decision Problems for Subclasses of Rational Relations over Finite and Infinite Words". In: *Discret. Math. Theor. Comput. Sci.* 21.3 (2019). DOI: 10.23638/DMTCS-21-3-4.

[MB08] L. M. de Moura and N. S. Bjørner. "Z3: An Efficient SMT Solver". In: *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*. Ed. by C. R. Ramakrishnan and J. Rehof. Vol. 4963. Lecture Notes in Computer Science. Springer, 2008, pp. 337–340. DOI: 10.1007/978-3-540-78800-3_24.

[McM93] K. L. McMillan. *Symbolic model checking*. Kluwer, 1993. ISBN: 978-0-7923-9380-1. DOI: 10.1007/978-1-4615-3190-6.

[MF71] A. R. Meyer and M. J. Fischer. "Economy of Description by Automata, Grammars, and Formal Systems". In: *12th Annual Symposium on Switching and Automata Theory, East Lansing, Michigan, USA, October 13-15, 1971*. IEEE Computer Society, 1971, pp. 188–191. DOI: 10.1109/SWAT.1971.11.

[Min67] M. L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Inc., 1967. ISBN: 978-0-13-165563-8.

[MM77] M. Magidor and J. Malitz. "Compact extensions of L(Q) (part 1a)". In: *Annals of Mathematical Logic* 11.2 (1977), pp. 217–261. ISSN: 0003-4843. DOI: https://doi.org/10.1016/0003-4843(77)90019-5.

[Mot36] T. S. Motzkin. "Beiträge zur Theorie der linearen Ungleichungen". PhD thesis. University of Basel, 1936.

[MS72] A. R. Meyer and L. J. Stockmeyer. "The Equivalence Problem for Regular Expressions with Squaring Requires Exponential Space". In: *13th Annual Symposium on Switching and Automata Theory, College Park, Maryland, USA, October 25-27, 1972*. IEEE Computer Society, 1972, pp. 125–129. DOI: 10.1109/SWAT.1972.29.

[MSL21] O. Markgraf, D. Stan, and A. W. Lin. "Learning Union of Integer Hypercubes with Queries - (with Applications to Monadic Decomposition)". In: *Computer Aided Verification - 33rd International Conference, CAV 2021, Virtual Event, July 20-23, 2021, Proceedings, Part II*. Ed. by A. Silva and K. R. M. Leino. Vol. 12760. Lecture Notes in Computer Science. Springer, 2021, pp. 243–265. DOI: 10.1007/978-3-030-81688-9_12.

[Nev02] F. Neven. "Automata, Logic, and XML". In: *Computer Science Logic, 16th International Workshop, CSL 2002, 11th Annual Conference of the EACSL, Edinburgh, Scotland, UK, September 22-25, 2002, Proceedings*. Ed. by J. C. Bradfield. Vol. 2471. Lecture Notes in Computer Science. Springer, 2002, pp. 2–26. DOI: 10.1007/3-540-45793-3_2.

[NS57] A. Nerode and B. P. Sauer. *Fundamental Concepts in the Theory of Systems (WADC Technical Report)*. ASTIA Document No. AD 155741. Wright Air Development Center, Air Research and Development Command, United States Air Force, 1957.

[Pin22] J.-É. Pin. *Mathematical Foundations of Automata Theory*. https://www.irif.fr/~jep/PDF/MPRI/MPRI.pdf. 2022.

[PR04] A. Podelski and A. Rybalchenko. "Transition Invariants". In: *19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings*. IEEE Computer Society, 2004, pp. 32–41. DOI: 10.1109/LICS.2004.1319598.

[Pre29] M. Presburger. "Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt". In: *Comptes Rendus du I congres de Mathematiciens de Pays Slaves* (1929), pp. 92–101.

[QSW17] K. Quaas, M. Shirmohammadi, and J. Worrell. "Revisiting reachability in timed automata". In: *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*. IEEE Computer Society, 2017, pp. 1–12. DOI: 10.1109/LICS.2017.8005098.

[Rab70] M. O. Rabin. "Weakly definable relations and special automata". In: *Mathematical Logic and Foundations of Set Theory*. Ed. by Y. Bar-Hillel. North-Holland Publishing Company, 1970, pp. 1–23. DOI: 10.1016/S0049-237X(08)71929-3.

[Rac78] C. Rackoff. "The Covering and Boundedness Problems for Vector Addition Systems". In: *Theor. Comput. Sci.* 6 (1978), pp. 223–231. DOI: 10.1016/0304-3975(78)90036-1.

[Rad08] F. G. Radmacher. "An Automata Theoretic Approach to Rational Tree Relations". In: *SOFSEM 2008: Theory and Practice of Computer Science, 34th Conference on Current Trends in Theory and Practice of Computer Science, Nový Smokovec, Slovakia, January 19-25, 2008, Proceedings*. Ed. by V. Geffert, J. Karhumäki, A. Bertoni, B. Preneel, P. Návrat, and M. Bieliková. Vol. 4910. Lecture Notes in Computer Science. Springer, 2008, pp. 424–435. DOI: 10.1007/978-3-540-77566-9_37.

[Ram30] F. P. Ramsey. "On a Problem of Formal Logic". In: *Proceedings of the London Mathematical Society* s2-30.1 (Jan. 1930), pp. 264–286. ISSN: 0024-6115. DOI: 10.1112/plms/s2-30.1.264.

[Rao97] J.-C. Raoult. "Rational tree relations." eng. In: *Bulletin of the Belgian Mathematical Society - Simon Stevin* 4.1 (1997), pp. 149–176. DOI: 10.36045/bbms/1105730627.

[RS59] M. O. Rabin and D. S. Scott. "Finite Automata and Their Decision Problems". In: *IBM J. Res. Dev.* 3.2 (1959), pp. 114–125. DOI: 10.1147/RD.32.0114.

[Rub08] S. Rubin. "Automata Presenting Structures: A Survey of the Finite String Case". In: *Bull. Symb. Log.* 14.2 (2008), pp. 169–209. DOI: 10.2178/bsl/1208442827.

[Saf88] S. Safra. "On the Complexity of omega-Automata". In: *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*. IEEE Computer Society, 1988, pp. 319–327. DOI: 10.1109/SFCS.1988.21948.

[Sak09] J. Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009. ISBN: 978-0-521-84425-3. DOI: 10.1017/CBO9781139195218.

[Sav70] W. J. Savitch. "Relationships Between Nondeterministic and Deterministic Tape Complexities". In: *J. Comput. Syst. Sci.* 4.2 (1970), pp. 177–192. DOI: 10.1016/S0022-0000(70)80006-X.

[Sch08] U. Schöning. *Logic for computer scientists*. Birkhäuser, 2008. ISBN: 978-0-8176-4762-9. DOI: 10.1007/978-0-8176-4763-6.

[Sei+04] H. Seidl, T. Schwentick, A. Muscholl, and P. Habermehl. "Counting in Trees for Free". In: *Automata, Languages and Programming: 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings*. Ed. by J. Díaz, J. Karhumäki, A. Lepistö, and D. Sannella. Vol. 3142. Lecture Notes in Computer Science. Springer, 2004, pp. 1136–1149. DOI: 10.1007/978-3-540-27836-8_94.

[Sei94] H. Seidl. "Haskell Overloading is DEXPTIME-Complete". In: *Inf. Process. Lett.* 52.2 (1994), pp. 57–60. DOI: 10.1016/0020-0190(94)00130-8.

[Sim75] I. Simon. "Piecewise testable events". In: *Automata Theory and Formal Languages, 2nd GI Conference, Kaiserslautern, May 20-23, 1975*. Ed. by H. Barkhage. Vol. 33. Lecture Notes in Computer Science. Springer, 1975, pp. 214–222. DOI: 10.1007/3-540-07407-4_23.

[Sip97] M. Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997. ISBN: 978-0-534-94728-6.

[Son85] E. D. Sontag. "Real Addition and the Polynomial Hierarchy". In: *Inf. Process. Lett.* 20.3 (1985), pp. 115–120. DOI: 10.1016/0020-0190(85)90076-6.

[SS82] J. H. Schmerl and S. G. Simpson. "On the role of Ramsey quantifiers in first order arithmetic". In: *The Journal of Symbolic Logic* 47.2 (1982), pp. 423–435. DOI: 10.2307/2273152.

[Ste67] R. E. Stearns. "A Regularity Test for Pushdown Machines". In: *Inf. Control.* 11.3 (1967), pp. 323–340. DOI: 10.1016/S0019-9958(67)90591-8.

[SVW87] A. P. Sistla, M. Y. Vardi, and P. Wolper. "The Complementation Problem for Büchi Automata with Appplications to Temporal Logic". In: *Theor. Comput. Sci.* 49 (1987), pp. 217–237. DOI: 10.1016/0304-3975(87)90008-9.

[SW15] J. Swernofsky and M. Wehar. "On the Complexity of Intersecting Regular, Context-Free, and Tree Languages". In: *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*. Ed. by M. M. Halldórsson, K. Iwama, N. Kobayashi, and B. Speckmann. Vol. 9135. Lecture Notes in Computer Science. Springer, 2015, pp. 414–426. DOI: 10.1007/978-3-662-47666-6_33.

[Sze88] R. Szelepcsényi. "The Method of Forced Enumeration for Nondeterministic Automata". In: *Acta Informatica* 26.3 (1988), pp. 279–284. DOI: 10.1007/BF00299636.

[Sze94] A. Szepietowski. *Turing Machines with Sublogarithmic Space*. Vol. 843. Lecture Notes in Computer Science. Springer, 1994. ISBN: 3-540-58355-6. DOI: 10.1007/3-540-58355-6.

[Tho90] W. Thomas. "Automata on Infinite Objects". In: *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*. Ed. by J. van Leeuwen. Elsevier and MIT Press, 1990, pp. 133–191. DOI: 10.1016/B978-0-444-88074-1.50009-3.

[TL08] A. W. To and L. Libkin. "Recurrent Reachability Analysis in Regular Model Checking". In: *Logic for Programming, Artificial Intelligence, and Reasoning, 15th International Conference, LPAR 2008, Doha, Qatar, November 22-27, 2008. Proceedings*. 2008, pp. 198–213. DOI: 10.1007/978-3-540-89439-1_15.

[TL10] A. W. To and L. Libkin. "Algorithmic Metatheorems for Decidable LTL Model Checking over Infinite Systems". In: *Foundations of Software Science and Computational Structures, 13th International Conference, FOSSACS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*. 2010, pp. 221–236. DOI: 10.1007/978-3-642-12032-9_16.

[To10] A. W. To. "Model Checking Infinite-State Systems: Generic and Specific Approaches". PhD thesis. University of Edinburgh, 2010. URL: https://era.ed.ac.uk/handle/1842/4671.

[Val75] L. G. Valiant. "Regularity and Related Problems for Deterministic Pushdown Automata". In: *J. ACM* 22.1 (1975), pp. 1–10. DOI: 10.1145/321864.321865.

[Vea+17] M. Veanes, N. Bjørner, L. Nachmanson, and S. Bereg. "Monadic Decomposition". In: *J. ACM* 64.2 (2017), 14:1–14:28. DOI: 10.1145/3040488.

[VSS05] K. N. Verma, H. Seidl, and T. Schwentick. "On the Complexity of Equational Horn Clauses". In: *Automated Deduction - CADE-20, 20th International Conference on Automated Deduction, Tallinn, Estonia, July 22-27, 2005, Proceedings*. Ed. by R. Nieuwenhuis. Vol. 3632. Lecture Notes in Computer Science. Springer, 2005, pp. 337–352. DOI: 10.1007/11532231_25.

[VW86] M. Y. Vardi and P. Wolper. "An Automata-Theoretic Approach to Automatic Program Verification". In: *Proceedings of the Symposium on Logic in Computer Science (LICS '86), Cambridge, Massachusetts, USA, June 16-18, 1986*. IEEE Computer Society, 1986, pp. 332–344.

[Weh14] M. Wehar. "Hardness Results for Intersection Non-Emptiness". In: *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*. Ed. by J. Esparza, P. Fraigniaud, T. Husfeldt, and E. Koutsoupias. Vol. 8573. Lecture Notes in Computer Science. Springer, 2014, pp. 354–362. DOI: 10.1007/978-3-662-43951-7_30.

[Weh16] M. Wehar. "On the Complexity of Intersection Non-Emptiness Problems". PhD thesis. University of Buffalo, 2016. URL: http://www.michaelwehar.com/documents/mwehar_dissertation.pdf.

[Wei97] V. Weispfenning. "Complexity and Uniformity of Elimination in Presburger Arithmetic". In: *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation, ISSAC 1997, Maui, Hawaii, USA, July 21-23, 1997*. Ed. by B. W. Char, P. S. Wang, and W. Küchlin. ACM, 1997, pp. 48–53. DOI: 10.1145/258726.258746.

[Wei99] V. Weispfenning. "Mixed Real-Integer Linear Quantifier Elimination". In: *Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation, ISSAC '99, Vancouver, B.C., Canada, July 29-31, 1999*. Ed. by K. O. Geddes, B. Salvy, and S. S. Dooley. ACM, 1999, pp. 129–136. DOI: 10.1145/309831.309888.

[Wor13] J. Worrell. "Revisiting the Equivalence Problem for Finite Multitape Automata". In: *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*. Ed. by F. V. Fomin, R. Freivalds, M. Z. Kwiatkowska, and D. Peleg. Vol. 7966. Lecture Notes in Computer Science. Springer, 2013, pp. 422–433. DOI: 10.1007/978-3-642-39212-2_38.

Curriculum Vitae: Pascal Bergsträßer

Employment

- Since 2020: **Scientific Employee** at the University of Kaiserslautern-Landau

Education

- 2020 - 2025: **PhD Student** at the University of Kaiserslautern-Landau
- 2018 - 2020: **Master in Computer Science** at the University of Kaiserslautern
- 2015 - 2018: **Bachelor in Computer Science** at the University of Kaiserslautern

Publications

[Ber+24] P. Bergsträßer, M. Ganardi, A. W. Lin, and G. Zetzsche. "Ramsey Quantifiers in Linear Arithmetics". In: *Proceedings of the ACM on Programming Languages* 8.POPL (2024), pp. 1–32. DOI: 10.1145/3632843.

[BKLZ24] P. Bergsträßer, C. Köcher, A. W. Lin, and G. Zetzsche. "The Power of Hard Attention Transformers on Data Sequences: A formal language theoretic perspective". In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. Ed. by A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang. 2024. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/af58a33861ac45472ea1cc5860d2b13e-Paper-Conference.pdf.

[BG23] P. Bergsträßer and M. Ganardi. "Revisiting Membership Problems in Subclasses of Rational Relations". In: *38th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2023, Boston, MA, USA, June 26-29, 2023*. IEEE, 2023, pp. 1–14. DOI: 10.1109/LICS56636.2023.10175722.

[Ber+22] P. Bergsträßer, M. Ganardi, A. W. Lin, and G. Zetzsche. "Ramsey Quantifiers over Automatic Structures: Complexity and Applications to Verification". In: *37th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2022, Haifa, Israel, August 2-5, 2022*. Ed. by C. Baier and D. Fisman. ACM, 2022, 28:1–28:14. DOI: 10.1145/3531130.3533346.

[BGZ21] P. Bergsträßer, M. Ganardi, and G. Zetzsche. "A Characterization of Wreath Products Where Knapsack Is Decidable". In: *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference)*. Ed. by M. Bläser and B. Monmege. Vol. 187. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 11:1–11:17. DOI: 10.4230/LIPICS.STACS.2021.11.