Definability and Model Checking: The Role of Orders and Compositionality

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatiker

Tobias Ganzow

aus

Seesen

Berichter: Univ.-Prof. Dr. Erich Grädel
Dr. habil. Luc Segoufin

Tag der mündlichen Prüfung: 22. Februar 2011

ABSTRACT

Finite model theory and descriptive complexity theory are concerned with assessing the expressive power of logics over finite models and with relating the descriptive resources needed for defining a class of structures to the computational complexity of the corresponding decision problem. In recent years, also the model theory and computational handling (e.g. of the model checking problem) of finitely representable infinite structures have gained much interest.

In the first part of this thesis we study how the expressive power of logics over finite structures is affected by the presence of orders. We will first review the concept of order-invariant definability where we allow formulae to use an additional order-relation that is not present in a given structure in such a way that the truth of the formula in the structure does not depend on the actual interpretation of the order relation, and show that, in the context of monadic second-order logic, order-invariance yields more expressive power than adding modulo-counting quantifiers.

Second, we investigate structures with weaker forms of orderings, namely locally ordered graphs, in which only the neighbourhoods of the vertices are linearly ordered. Using recent results on reachability algorithms by Reingold, we are able to show that the transitive closure of the edge relation in such graphs is definable in deterministic transitive closure logic (DTC) in a two-sorted setting, and this observation is the key to linking the descriptive power of DTC with counting to the computational power of logspace-bounded Turing machines.

The second part of the thesis is concerned with techniques for model checking weak monadic second-order logic (WMSO) on the class of so-

called inductive structures that allow for a finite representation via systems of equations, and which includes structures relevant for verification purposes such as the infinite binary tree, infinite lists, etc. Our new approach presents an algorithmic alternative to automata-theoretic methods, which exhibit certain drawbacks, and is based on a purely logical decomposition technique using the defining equations. Further, the deployed techniques can be extended to obtain a model checking algorithm for the extension of WMSO by an unbounding quantifier, and thus establish the decidability of its model checking problem on the class of inductive structures.

ZUSAMMENFASSUNG

Die Grenzen der Audrucksstärke von Logiken auszuloten und Zusammenhänge zwischen den für die Definition bestimmter Eigenschaften von Strukturen benötigten deskriptiven Mitteln und der Komplexität der entsprechenden Entscheidungsprobleme zu finden, sind zentrale Aspekte der Endlichen Modelltheorie und Deskriptiven Komplexitätstheorie. Vor allem in den letzten Jahren ist zunehmend auch die Modelltheorie und die algorithmische Handhabbarkeit von endlich repräsentierbaren unendlichen Strukturen in den Fokus der Forschung gerückt.

Im ersten Teil der Dissertation wird untersucht, wie sich das Vorhandensein von Ordnungen in einer Struktur auf die Ausdrucksstärke von Logiken auswirkt, und zunächst das Konzept der sogenannten ordnungsinvarianten Definierbarkeit motiviert. Hierbei darf in einer Formel eine Ordnungsrelation verwendet werden, die nicht Bestandteil der Struktur ist, allerdings darf das Auswertungsergebnis der Formel innerhalb einer Struktur nicht von einer bestimmten (linearen) Ordnung der Elemente abhängen. Wir können zeigen, dass ordnungsinvariante monadische Logik zweiter Stufe (MSO) echt ausdrucksstärker ist als die Erweiterung von MSO um Modulo-Zählquantoren.

Anschließend untersuchen wir schwächere Arten von Ordnungen am Beispiel von lokal geordneten ungerichteten Graphen, in denen nur die Nachbarschaften von Knoten linear geordnet sind. Für diese Klasse können wir unter Ausnutzung eines vor wenigen Jahren von Reingold präsentierten Erreichbarkeitsalgorithmus zeigen, dass die transitive Hülle der Kantenrelation durch eine DTC-Formel definierbar ist, wenn die Graphen als zweisortige Struktur vorliegen. Aus diesem Resultat folgt schließlich, dass die Ausdrucks-

stärke von DTC mit Zählquantoren genau mit der Platzkomplexitätsklasse Logspace zusammenfällt.

Der zweite Teil der Arbeit widmet sich dem Auswertungsproblem von WMSO auf einer Klasse von unendlichen Strukturen, den *induktiven Strukturen*, die durch endliche Gleichungssysteme repräsentiert werden können und somit häufig bei Verifikationsproblemen anzutreffende Strukturen wie z.B. den unendlichen Binärbaum oder unendliche Listen umfassen. Im Gegensatz zu automatentheoretischen Methoden, die mit gewissen Nachteilen verbunden sind, wird ein Algorithmus entwickelt, der auf der Dekompositionsmethode beruht und damit direkt auf Formelebene arbeitet. Weiterhin wird gezeigt, wie sich der Ansatz erweitern lässt, um einen Algorithmus für das Auswertungsproblem von WMSO mit dem Unbeschränktheitsquantor auf induktiven Strukturen zu erhalten und somit dessen Entscheidbarkeit zu zeigen.

CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

1 INTRODUCTION

Assessing the potential and limits of the expressive power of logics is a central problem in finite model theory, and a wide range of methods and tools has been developed to conduct this task. The results we obtain in the first part of the thesis rely on Ehrenfeucht-Fraïssé games, which are a standard tool for proving limits of expressive power, and on the relationship between logical expressiveness and computational complexity, which is studied in descriptive complexity theory.

In 1974, Fagin lay the grounds of descriptive complexity theory by proving a connection between the computational resources needed for deciding a property of structures and the logical resources needed for describing this property [Fag74]. In particular, he proved that a property of graphs (e.g. 3-colourability) is decidable in non-deterministic polynomial time if and only if the class of graphs exhibiting this property can be defined by a sentence of existential second-order logic. This lead to the notion of *capturing a complexity class*, i.e., in modern terms, Fagin's result states that existential second-order logic captures NP. Analogously, it was shown that similar correspondences can be obtained for various other logics. For example, least fixed-point logic captures PTIME on ordered structures, known as the Immerman-Vardi Theorem [Imm86, Var82], and transitive closure logic (TC) and deterministic transitive closure logic (DTC) capture NLOGSPACE and LOGSPACE, respectively, on ordered structures [Imm87, Imm88]. But the latter results expose a serious blemish: the characterisation only holds on *ordered* structures.

In fact, characterisations of complexity classes on unordered structures are extremely scarce which is partly due to the particular weakness of many

logics with respect to defining properties that involve counting on unordered structures. As of today, a very prominent open problem in descriptive complexity theory is to find a logic capturing PTIME on *all* finite structures if there is such a logic at all. Solving this problem in either way would have direct implications for the infamous question whether P equals NP, or at least show possible new routes to its answer.

In contrast to Turing machines that always work on the representation of a structure as a word, a formula is evaluated on a given structure as-is. Hence, the machine can possibly make use of a linear order that is necessarily induced by the representation of the structure as an (ordered) word, as long as its output does not depend on a particular representation, whereas the logic does not enjoy this advantage. This motivated the definition of *order-invariance* in the realm of logics: We call a formula order-invariant on a class of structures if, for any structure in this class, the formula either holds for all possible orderings of the universe of the structure or if it does not hold for any possible ordering.

Denoting the set of order-invariant τ-formulae of a logic $\mathcal{L}$ by $\mathcal{L}[\tau]_{<\text{-inv}}$, we obtain that $\mathrm{LFP}_{<\text{-inv}}$ captures PTIME, $\mathrm{DTC}_{<\text{-inv}}$ captures LOGSPACE, and $\mathrm{TC}_{<\text{-inv}}$ captures NLOGSPACE on the class of all finite structures. However, while it is not difficult to design formulae that are order-invariant by construction, already the decision problem whether a first-order formula is order-invariant on the rather restricted class of strings is undecidable as shown by Benedikt and Segoufin [BS05]. Hence, although we will speak of *order-invariant logics* in the following, it is important to keep in mind that they lack an effective syntax which is a very important property of a reasonable logic and usually taken for granted. In particular, this aspect renders the above characterisations of the complexity classes rather unsatisfactory.

With respect to definability, the question arises as to how much expressive power the order-invariant logic gains over the plain logic. If a logic has the interpolation property on a class of structures, it is, in particular, *closed under order-invariant sentences* in the following sense. Assume that a formula $\varphi \in \mathcal{L}[\tau \dot\cup \{<\}]$ is order-invariant. Then

$$\varphi \vDash (\text{“}<' \text{ is a linear order”} \to \varphi[< \,/\, <']) =: \varphi' \in \mathcal{L}[\tau \dot\cup \{<'\}]$$

for any logic $\mathcal{L}$ that is capable of axiomatising linear orders. Hence, there exists an interpolant χ over the common vocabulary τ such that $\varphi \vDash \chi$ and

$\chi \vDash \varphi'$, and if φ order-invariantly defines a class $\mathcal{K}$ of τ-structures, then $\mathcal{K}$ is already defined by the τ-sentence χ (see [EF99]). Since first-order logic has the interpolation property on the class of all (finite *and* infinite) structures, it is closed under order-invariant sentences, i.e. order-invariant FO collapses to plain FO. However, in the finite, this is no longer the case. Gurevich showed that the class of Boolean algebras with an even number of atoms is order-invariantly FO definable, but not definable in plain FO. This implies that, in the finite, first-order logic is neither closed under order-invariant sentences, nor does it have the interpolation property. Otto later presented a further separating example involving a class of special graphs with attached Boolean algebras on which connectedness (of the graph component) is order-invariantly FO definable, but not even definable in infinitary logic with counting quantifiers [Ott00].

In both cases, the considered structures contain a Boolean algebra, and hence the classes have unbounded tree width. In contrast, Benedikt and Segoufin pursued the investigation of order-invariant FO on classes of bounded tree width, and show that order-invariant FO collapses to FO on strings and trees using algebraic characterisation theorems for FO-definable string and tree languages [BS09]. Furthermore, they show that order-invariant FO is at most as expressive as MSO on graphs of bounded degree or tree width. It would be desirable to obtain such an upper bound on the expressiveness in the general case, and it is conceivable that all order-invariantly FO definable queries are actually definable in plain MSO. In fact, all known queries separating order-invariant FO from plain FO are MSO definable. However, since they rely on quite different properties of the structure, it is questionable whether it is possible to devise a direct translation.

In Chapter 3, we focus on order-invariant MSO and counting MSO (CMSO), the extension of MSO by quantifiers that can specify the size of definable sets modulo fixed integers. We introduce an Ehrenfeucht-Fraïssé game characterising the expressive power of CMSO which will be the main tool in proving that order-invariant MSO is strictly more expressive than counting MSO on graphs, which confirms a long-standing open conjecture by Courcelle [Cou96, Conjecture 7.3].

A further question is whether we can capture complexity classes, if not on unordered structures, then at least on classes of structures that feature only weak forms of orderings. In particular this question was studied in the

mid 1990's for Logspace complexity classes and locally ordered graphs. A one-way (or two-way) locally ordered graph is equipped with one (or two) relations that induce, for each vertex, a linear order on its neighbours. In a one-way locally ordered graph, a ternary relation induces a linear order on the successors of each vertex, i.e. on the neighbours reachable from the vertex via outgoing edges, and in a two-way locally ordered graph, a second relation analogously induces linear orders on the predecessors of vertices. Etessami and Immerman proved that transitive closure logic with counting (TC+C) captures NLogspace on two-way locally ordered graphs [EI95b, EI00]. In general, TC+C is still too weak to define a linear order on the entire graph, but it is possible to define a canonical isomorphic copy of the graph on the number sort that comes with the counting extension. That is, a locally ordered graph provides strictly less structure than a linearly ordered graph, but this can be compensated by extending the logic.

In Chapter 4, we combine this approach with a recent result from complexity theory demonstrating the fruitful connections between the disciplines. The deterministic space complexity of the reachability problem in undirected graphs was a longstanding open problem in complexity theory. For directed graphs, reachability is known to be NLogspace-complete which implies, by Savitch's Theorem, that the problem is in Dspace($\mathcal{O}(\log^2 n)$). For undirected graphs, the problem was known to be complete for the rather artificial class SymLogspace (symmetric Logspace), which is, in fact, defined in terms of the problem. While the problem was believed to be in Logspace, it was not until 2005 that Reingold came up with a proof of this conjecture using a sophisticated argument involving expander graphs [Rei05]. Whereas other algorithms (e.g. Savitch's) need logarithmic memory per iteration or phase, and hence would only meet a logarithmic space requirement if the number of phases could always be reduced to a constant, the novel approach leads to an algorithm that only needs a constant amount of additional memory per phase.

By formalising Reingold's algorithm, we obtain a DTC formula defining the transitive closure of a locally ordered undirected graph with an auxiliary linearly ordered number sort. Combining this formula with the techniques used by Etessami and Immerman showing that TC+C captures NLogspace on two-way locally ordered graphs, we can prove our main result that DTC+C captures Logspace on undirected locally ordered graphs.

In Part II of the thesis, we shift our focus from finite structures to classes of structures that are, in general, infinite but still finitely representable. Those comprise well-known classes such as constraint databases or automatic structures—in this thesis, we study the class of structures representable via finite systems of equations, called inductive structures, and, in particular, we present a compositional method for model checking weak monadic second-order logic (WMSO) on this class.

The investigation of compositional model checking techniques started in the late 1950's with the work of Feferman and Vaught [FV59] on first-order theories of structures that are obtained as generalised products of a possibly infinite, indexed family of structures. In particular, they showed how to reduce the first-order theory of a generalised product to the FO theory of the factors and the MSO theory of the index set. Towards this, they detailed an effective translation of FO formulae speaking about the generalised product to a sequence of formulae speaking about the factor structures and an MSO formula with access to predicates that encode, for each formula of the mentioned sequence, in which factors it is satisfied.

Around the same time, Büchi, Elgot and Trakhtenbrot independently established the connection between finite automata and *weak* monadic second-order logic [Büc60, Elg61, Tra61], where second-order quantifiers range over *finite* subsets of the universe. Decidability of the unrestricted MSO theory of $(\omega, <)$, also known as the *monadic theory of one successor* (S1S), was proved by Büchi [Büc62] shortly afterwards. Later Rabin [Rab69] proved the decidability of the MSO theory of the binary tree, i.e. of the *monadic theory of two successors* (S2S). The latter proof uses a translation of formulae into tree-automata that accept a tree-encoding of a structure if and only if it is a model of the respective formula. Since an interpretation of a structure in the binary tree yields a reduction of its MSO theory to S2S, and hence implies decidability, Rabin's Theorem became one of the most celebrated theorems in theoretical computer science.

It was only later, in 1975, that Shelah proved the decidability of the MSO theory of $(\omega, <)$ and other countable linear orders in his seminal paper [She75] using purely logical methods and, in particular, the composition of monadic theories. However, this method received little attention among computer scientists until it was presented in a more accessible survey paper by Thomas [Tho97]. Furthermore, the original proof is based on a recursive

computation of the theory involving the enumeration of all types of a given quantifier rank, and thus cannot be used as a basis for practical algorithms.

Consequently, the existing implementations of (weak) MSO model checking tools are mainly based on automata-theoretic techniques. Despite its importance for obtaining elegant theoretical results, Rabin's Theorem is challenging for practical purposes. The construction of an automaton that is equivalent to the formula involves a complementation for each quantifier alternation and negation in the formula, and the complementation of Rabin tree automata presents several difficulties: The algorithm is difficult to understand and implement, and the number of states grows exponentially. The situation is better for weak MSO, where complementation of the constructed automata, though still yielding an exponential blow-up, is easier than in the unrestricted case. Admittedly, since the complexity of (even the weak) MSO theory of $(\omega, <)$ is already non-elementary, no technique whatsoever can perform equally well on all input instances.

The focus on weak MSO is based on two arguments. First, WMSO subsumes widely used temporal logics such as LTL and CTL which makes it a reasonably expressive logic. Furthermore, many problems arising in software or hardware verification where the domain of interest is often finite but not a priori bounded in size allow for a natural formalisation in weak MSO. Second, Shelah's composition method could not be generalised to MSO on the binary tree so far. In fact, the question about a composition-based proof of the decidability of S2S is a major open problem, and has a close relationship to the challenge of understanding the algebraic structure of regular tree languages.

The best known tool for model-checking WMSO is MONA [HJJ+95, EKM98] which has been used to verify both hardware [BK95] and pointer manipulating programs [JJKS97]. It is a mature tool, but being based on automata, it only works on linear orders and trees such that more complex structures must first be encoded, which can decrease MONA's performance considerably up to the point where it becomes unusable even for simple formulae.

In contrast, the approach we are presenting in Chapter 5 works directly on the much broader class of inductive structures which comprises classical structures such as $(\omega, <)$ and the infinite binary tree as well as practically relevant structures such as doubly-linked lists or lists of lists. Not having to

interpret these structures in the binary tree makes the approach much more versatile and avoids inflating the formulae.

Our technique for decomposing the formulae yields decompositions that are similar to the reduction sequences described by Courcelle [Cou90] and Makowsky [Mak04], which follow the original idea of Feferman and Vaught. Whereas Courcelle shows how to obtain a reduction sequence for disjoint unions of structures without any connections between the components, and Makowsky exemplifies a construction for ordered sums of disjoint structures, we follow a different approach which matches our more general definition of inductive structures.

We continue to show how to combine the decompositions with the equations defining the inductive structure to obtain a model checking game. Such games have a long tradition and provide a particularly intuitive view on the semantics of a logical formalism. They are played by two opponents, called Verifier and Falsifier, and Verifier tries to prove that the formula is satisfied whereas Falsifier tries to disprove this.

For first-order or weak monadic second-order model checking, reachability games suffice. Positions consist of a formula (taken from the set of subformulae of the formula to be checked) and an interpretation of its free variables. Hence, the size of the game graph is determined by the product of the size of the structure and the size of the formula, i.e. it is finite if and only if the structure is finite. However, regardless of the structure, all possible plays are finite since the length of a play is bounded by the length of the formula.

The model checking game we present is, roughly speaking, obtained as the product graph of the subformulae and decompositions of formulae with initial unravellings of the inductive structure on which the formula is to be evaluated. In contrast to the naïve model checking game for WMSO sketched above, we obtain a game with a finite arena, and the players' objectives are to reach terminal positions that are winning for them. However, since the game graph contains loops now, and running through a loop infinitely often is always bad for one of the players, we actually use a Büchi winning condition. Still, winning regions in Büchi games can be efficiently computed [GTW02]. Further, it is worth noting that, although the decomposition technique is not tied to WMSO but also works for MSO formulae, the game crucially relies on the limitation that set quantifiers only range over finite sets.

Finally, we demonstrate the versatility and potential of this game-based approach by showing how it can be modified to allow for model checking WMSO extended with an *unbounding quantifier* (WMSO+U) introduced by Bojańczyk [Boj04]. This establishes the decidability on inductive structures, i.e. in particular on the binary tree. Previously, decidability of WMSO+U has only been shown on labellings of $(\omega, <)$ [Boj09].

Acknowledgements

First of all, I would like to thank my supervisor Erich Grädel for giving me the opportunity to pursue my research interests, for providing the needed freedom, and his profound advice. Furthermore, I am very grateful to Luc Segoufin for his prompt readiness to commit himself to the task of acting as the co-examiner of this thesis.

I am very indebted to Sasha Rubin and Łukasz Kaiser for many inspiring discussions not only about cliquey grids and decomposing formulae, as well as Diana Fischer, Bernd Puchala, and Roman Rabinovich who were keen enough to agree to proofread a draft of this thesis and helped me improving the presentation. Many thanks also to the rest of my colleagues—Alex, Arnaud, Christof, Daniel, Dietmar, Frank, Henrik, Ingo, Jan, Jörg, Kari, Marcus, Martin, Michael H., Michael U., Michaela, Namit, Nico, Philipp, Stefan R., Stefan W., Vince, and Wladimir—who populated the MGI & i7 floor from early in the morning until late at night at times and who made my time at the institute very enjoyable.

Last but not least, I would like to express my deep gratitude to my family, and especially to my wife Astrid, for their continuing love, support, and encouragement.

2 PRELIMINARIES

We assume some basic familiarity with mathematical logic and algorithmic model theory, and will only briefly introduce the basic notions and concepts mainly for fixing the notation. For further details and introductory texts we refer the reader to [EF99, EFT94, Lib04].

We let $\mathbb{N}$ denote the set of non-negative integers, and let $\mathbb{N}^+ := \mathbb{N} - \{0\}$. The initial subset $\{0, \dots, n-1\}$ of $\mathbb{N}$ will be denoted by $[n]_0$, and we let $[n] := \{1, \dots, n\}$ denote the initial subset of $\mathbb{N}^+$. Given a non-empty finite set $M = \{m_1, \dots, m_k\} \subseteq_{\text{fin}} \mathbb{N}^+$, let $\operatorname{lcm}(M) := \operatorname{lcm}(m_1, \dots, m_k)$ denote the least common multiple of all elements in M; for convenience, we define $\operatorname{lcm}(\varnothing) = 1$. For finite sets X and Y as well as M as before, we abbreviate that $|X| \equiv |Y| \pmod{m}$ for all $m \in M$ using the shorthand $|X| \equiv |Y| \pmod{M}$.

2.1 Structures

Given a finite vocabulary or signature $\tau = \{R_1, \dots, R_m, f_1, \dots, f_n\}$ of relation symbols R_i and function symbols f_i with associated arities $\operatorname{ar}(R_i)$ and $\operatorname{ar}(f_i)$, respectively, a τ-structure $\mathfrak{A} = \left(A, (R_i^{\mathfrak{A}})_{i\in[m]}, (f_i^{\mathfrak{A}})_{i\in[n]}\right)$ is determined by a non-empty set A called the *universe* of the structure and relations $R_i^{\mathfrak{A}} \subseteq A^{\operatorname{ar}(R_i)}$ and functions $f_i^{\mathfrak{A}} : A^{\operatorname{ar}(f_i)} \to A$ interpreting the corresponding symbols in τ. A structure over a vocabulary that does not contain any function symbols is called *relational*. In a relational structure $\mathfrak{A}$, every subset $U \subset A$ of its universe induces the substructure $\mathfrak{A} \cap U := \left(U, (R_i^{\mathfrak{A}} \cap U^{\operatorname{ar}(R_i)})_{i\in[m]}\right)$.

The class of all (finite) structures over a vocabulary τ is denoted by $\operatorname{Str}_{(\text{fin})}[\tau]$. The index may be omitted if it is clear from the context that only

finite structures are considered. If τ contains a binary relation symbol $<$, we denote by $\mathrm{Ord}[\tau]$ the class of linearly ordered τ-structures. For a fixed vocabulary τ, we call $\mathcal{K} \subseteq \mathrm{Str}[\tau]$ a *model class* if it is closed under isomorphism. A class of structures without a common vocabulary is called a *domain*, e.g. the domain of finite structures, the domain of finite graphs, the domain of finite words, and so on. Given a domain $\mathcal{D}$ and a vocabulary τ, we denote by $\mathcal{D}[\tau]$ the class of τ-structures in $\mathcal{D}$. Throughout the whole thesis, we restrict our attention to relational structures and sometimes allow constants (0-ary functions). The first part focuses on finite structures, whereas the structures investigated in the second part are, in general, infinite.

In Chapter 4, we investigate logics in the framework of many-sorted structures, and in particular on special two-sorted structures, so we narrow the definition to this specific case. Given a vocabulary $\tau = \{R_1, \ldots, R_m, f_1, \ldots, f_n\}$, a two-sorted τ-structure $\mathfrak{A}$ consists of two disjoint universes, A_1 and A_2, called the first sort and the second sort, respectively. Each relation symbol R has an associated arity $\mathrm{ar}(R) = (r_1, r_2)$ such that R is interpreted as $R^{\mathfrak{A}} \subseteq A_1^{r_1} \times A_2^{r_2}$. Each function symbol f also has an arity $\mathrm{ar}(f) = (r_1, r_2)$ and a type $\mathrm{tp}(f) \in \{1, 2\}$ depending on whether the result is an element from the second or the first sort, i.e. $f^{\mathfrak{A}} : A_1^{r_1} \times A_2^{r_2} \to A_{\mathrm{tp}(f)}$.

2.2 Logics

We provide an overview of the syntax and semantics of those logics used and studied in the following chapters such as first-order logic and its transitive closure extensions, as well as second-order logics and counting extensions.

2.2.1 First-Order Logic

The set of terms over a vocabulary τ contains the constant symbols from τ, all variables from the countable set of variables Var, and if $t_1, \ldots, t_{\mathrm{ar}(f)}$ are τ-terms, then $f t_1 \ldots t_{\mathrm{ar}(f)}$ is a τ-term for all function symbols $f \in \tau$. The set of first-order formulae over τ, $\mathrm{FO}[\tau]$, is obtained by closing the set of atomic formulae built from relation symbols in τ and the equality predicate applied to τ-terms under Boolean connectives such as $\neg, \wedge, \vee$ and existential and universal quantification over individual (first-order) variables. Variables not occurring inside the scope of a quantifier are called *free*, and *bound* otherwise. A formula without free variables is called a *sentence*. A *τ-interpretation* is

given by a τ-structure $\mathfrak{A}$ together with a variable assignment $\beta : \mathrm{Var} \to A$, and yields an interpretation of τ-terms by elements of the structure via

$$\begin{aligned} [\![x]\!]^{\mathfrak{A},\beta} &:= \beta(x) \\ [\![c]\!]^{\mathfrak{A},\beta} &:= c^{\mathfrak{A}} \\ [\![f\, t_1, \ldots, t_{\mathrm{ar}(f)}]\!]^{\mathfrak{A},\beta} &:= f^{\mathfrak{A}}([\![t_1]\!]^{\mathfrak{A},\beta}, \ldots, [\![t_{\mathrm{ar}(f)}]\!]^{\mathfrak{A},\beta}) \quad . \end{aligned}$$

Furthermore, a τ-interpretation $(\mathfrak{A}, \beta)$ is called a *model* of an $\mathrm{FO}[\tau]$ formula φ if $\beta : \mathrm{free}(\varphi) \to A$ is an assignment of all free variables and $\mathfrak{A}, \beta \vDash \varphi$ according to the following semantics.

$$\begin{aligned} &\mathfrak{A}, \beta \vDash t_1 = t_2 && \text{iff } [\![t_1]\!]^{\mathfrak{A},\beta} = [\![t_2]\!]^{\mathfrak{A},\beta} \\ &\mathfrak{A}, \beta \vDash R t_1, \ldots, t_{\mathrm{ar}(R)} && \text{iff } ([\![t_1]\!]^{\mathfrak{A},\beta}, \ldots, [\![t_{\mathrm{ar}(R)}]\!]^{\mathfrak{A},\beta}) \in R^{\mathfrak{A}} \\ &\mathfrak{A}, \beta \vDash \neg\varphi && \text{iff not } \mathfrak{A}, \beta \vDash \varphi \\ &\mathfrak{A}, \beta \vDash \varphi \wedge \psi && \text{iff } \mathfrak{A}, \beta \vDash \varphi \text{ and } \mathfrak{A}, \beta \vDash \psi \\ &\mathfrak{A}, \beta \vDash \varphi \vee \psi && \text{iff } \mathfrak{A}, \beta \vDash \varphi \text{ or } \mathfrak{A}, \beta \vDash \psi \\ &\mathfrak{A}, \beta \vDash \exists x \varphi(x) && \text{iff there ex. } a \in A \text{ s.th. } \mathfrak{A}, \beta[x \mapsto a] \vDash \varphi(x) \\ &\mathfrak{A}, \beta \vDash \forall x \varphi(x) && \text{iff for all } a \in A\text{: } \mathfrak{A}, \beta[x \mapsto a] \vDash \varphi(x) \quad . \end{aligned}$$

By $\beta[x \mapsto a]$ we denote the function obtained from β by either overriding the assignment of the variable x or extending β by mapping x to a. In Chapter 5, we also allow the use of the propositional constants $\top$ denoting *true* and $\bot$ denoting *false* with the obvious semantics that $\mathfrak{A} \vDash \top$ and $\mathfrak{A} \nvDash \bot$ for all structures $\mathfrak{A}$. We write $\varphi(\bar{x})$ to denote that the free variables of the formula are among $x_1, \ldots, x_n$, and instead of $\mathfrak{A}, \beta \vDash \varphi(\bar{x})$, we also write $\mathfrak{A} \vDash \varphi(\bar{a})$ or $\mathfrak{A}, \bar{a} \vDash \varphi(\bar{x})$ if $a_i = \beta(x_i)$ for all i. Furthermore, we denote by $\varphi^{\mathfrak{A}}$ with $\mathrm{free}(\varphi) = \{x_1, \ldots, x_n\}$ the set $\{(a_1, \ldots, a_n) : \mathfrak{A} \vDash \varphi(\bar{a})\}$.

2.2.2 Second-Order Logic

Second-Order logic, SO, is obtained by allowing the same formula building rules as for first-order logic, but also allow the use of second-order variables representing relations of arbitrary arity and quantification over second-order variables. Since SO is a very powerful logic which is algorithmically hardly manageable, there are some common restricted fragments. We focus on *monadic second-order logic*, MSO, where quantification is restricted to unary

relation variables, i.e. to sets. The semantics is obtained from the semantics of FO by adding the following two rules.

$$\begin{aligned}
&\mathfrak{A}, \beta \vDash \exists X \varphi(X) \quad \text{iff there ex. } U \subseteq A \text{ such that } \mathfrak{A}, \beta[X \mapsto U] \vDash \varphi(X) \\
&\mathfrak{A}, \beta \vDash \forall X \varphi(X) \quad \text{iff for all } U \subseteq A, \mathfrak{A}, \beta[X \mapsto U] \vDash \varphi(X) \ .
\end{aligned}$$

Furthermore, we will investigate *weak monadic second-order logic*, WMSO, which has the same syntax as MSO, but the semantics is altered such as to treat quantifiers as ranging over finite sets only, i.e. $\mathfrak{A} \vDash \exists X \varphi(X)$ if and only if $\mathfrak{A} \vDash \varphi(U)$ for some *finite* set $U \subseteq A$, and $\mathfrak{A} \vDash \forall X \varphi(X)$ if and only if $\mathfrak{A} \vDash \varphi(U)$ for all *finite* sets $U \subseteq A$.

2.2.3 Transitive Closure Logics

The transitive closure logic TC extends FO by an operator defining the transitive closure of definable relations, which is, in general, not possible in plain first-order logic. The syntax of FO is extended by the following formation rule

$$\begin{aligned}
&\text{if } \varphi(\bar{x}, \bar{y}, \bar{z}) \text{ is a } (2k + l)\text{-ary formula in TC,} \\
&\quad \text{then } [\mathrm{tc}_{\bar{x}\bar{y}}\, \varphi(\bar{x}, \bar{y}, \bar{z})](\bar{u}, \bar{v}) \text{ is also a formula in TC.}
\end{aligned}$$

The tuple $\bar{z}$ serves as parameters, and the tc-operator binds the variables in $\bar{x}$ and $\bar{y}$ such that the free variables of the formula are those in $\{\bar{u}, \bar{v}, \bar{z}\}$. Towards the definition of the semantics, for a binary relation R, let

$$\begin{aligned}
\mathrm{TC}(R) \coloneqq \{(a, b) \,:\, &\text{for some } n \in \mathbb{N} \text{ there ex. } a_0, a_1, \ldots, a_n \text{ such that} \\
&a = a_0,\ a_n = b, \text{ and } (a_i, a_{i+1}) \in R \text{ for all } 0 \leq i < n\} \ .
\end{aligned}$$

Then $\mathfrak{A} \vDash [\mathrm{tc}_{\bar{x}\bar{y}}\, \varphi(\bar{x}, \bar{y}, \bar{c})](\bar{a}, \bar{b})$ if $(\bar{a}, \bar{b}) \in \mathrm{TC}(\{(\bar{u}, \bar{v}) \,:\, \mathfrak{A} \vDash \varphi(\bar{u}, \bar{v}, \bar{c})\})$. Disregarding the parameter tuple $\bar{z}$, the formula φ defines a $2k$-ary relation, i.e. a binary "edge" relation on k-tuples of elements of $\mathfrak{A}$, and $[\mathrm{tc}_{\bar{x},\bar{y}}\, \varphi(\bar{x}, \bar{y})]$ defines the transitive closure of this relation. Hence, intuitively, $\mathfrak{A} \vDash [\mathrm{tc}_{\bar{x},\bar{y}}\, \varphi(\bar{x}, \bar{y})](\bar{a}, \bar{b})$ if there exists a path of φ-edges between $\bar{a}$ and $\bar{b}$.

In Chapter 4, we investigate a variant called *deterministic transitive closure logic*, DTC, where the formula $[\mathrm{dtc}_{\bar{x},\bar{y}}\, \varphi(\bar{x}, \bar{y}, \bar{z})](\bar{u}, \bar{v})$ defines the

transitive closure over the deterministic part of the relation defined by φ. Given a binary relation R, its *deterministic part* R_{det} is defined as

$$R_{\text{det}} = \{(a,b) \in R : \text{for all } c \text{ such that } (a,c) \in R,\ b = c\} \ ,$$

i.e. we only keep those edges which are unique outgoing edges of a vertex. Hence, $\mathfrak{A} \vDash [\text{dtc}_{\bar{x}\bar{y}}\, \varphi(\bar{x},\bar{y},\bar{c})](\bar{a},\bar{b})$ if $(\bar{a},\bar{b}) \in \text{TC}(R_{\text{det}})$ where $R \coloneqq \{(\bar{u},\bar{v}) : \mathfrak{A} \vDash \varphi(\bar{u},\bar{v},\bar{c})\}$, and since the deterministic part of a relation is FO-definable, the formula $[\text{dtc}_{\bar{x},\bar{y}}\, \varphi(\bar{x},\bar{y},\bar{z})](\bar{u},\bar{v}) \in \text{DTC}$ can be translated into the equivalent TC formula $[\text{tc}_{\bar{x},\bar{y}}\, \varphi(\bar{x},\bar{y},\bar{z}) \wedge \forall \bar{y}'(\varphi(\bar{x},\bar{y}',\bar{z}) \to \bar{y} = \bar{y}')](\bar{u},\bar{v})$.

Restricting the arity of the defining formulae, we obtain the syntactic fragments TC^k and DTC^k for all $k > 0$, in particular, in TC^k it is only possible to define transitive closures of $2k$-ary relations, and indeed Grohe showed that these fragments form an infinite hierarchy in the sense that, for every k, there are queries definable in TC^{k+1} that are not definable in TC^k [Gro96].

On linearly ordered structures, all basic arithmetic operations such as addition, multiplication, exponentiation are DTC-definable. We present definitions for those operations that are used later in Chapter 4.

$$\begin{aligned}
S(x,y) &\equiv x < y \wedge \neg\exists z(x < z \wedge z < y)\\
x + y = z &\equiv [\text{dtc}_{aba'b'}\, Saa' \wedge Sbb'](0,x,y,z)\\
x - y = z &\equiv [\text{dtc}_{aba'b'}\, Sa'a \wedge Sb'b](x,y,z,0)\\
x \cdot y = z &\equiv [\text{dtc}_{aba'b'}\, Saa' \wedge b' = b + x](0,0,y,z)\\
\lfloor x/y \rfloor = z &\equiv \exists x'(y \cdot z = x' \wedge 0 \le x - x' < y)\\
x^y = z &\equiv [\text{dtc}_{aba'b'}\, Saa' \wedge b' = b \cdot x](0,1,y,z)\\
l = \lceil \log x \rceil &\equiv 2^l \ge x \wedge 2^{l-1} < x \ .
\end{aligned}$$

Since all definitions are ultimately based on the successor relation, it is straightforward to extend them to the range of k-tuples of variables representing numbers between 0 and $n^k - 1$, where n is the size of the universe of the structure, by replacing $S(x,y)$ by the following definition saying that the tuple $\bar{\imath}'$ is the lexicographical successor of the tuple $\bar{\imath}$:

$$\mathrm{S}_k(\bar{\iota},\bar{\iota}') = \bigvee_{\ell<k}\left(\bigwedge_{j=1}^{\ell-1}(\iota_j=\iota'_j)\wedge\iota'_\ell=\iota_\ell+1\wedge\bigwedge_{j=\ell+1}^{k}(\iota_j=n-1\wedge\iota'_j=0)\right).$$

2.2.4 Logics and Two-Sorted Structures

In logics expressing properties of two-sorted structures, variables are typed in the sense that individual (first-order) variables either range over the first or the second sort universe, and second-order variables have an associated arity (r_1, r_2), analogously to relation symbols, fixing their range to $A_1^{r_1} \times A_2^{r_2}$. To make the distinction for individual variables explicit, we will consider formulae built over disjoint sets of individual first-sort and second-sort variables, using letters $u, v, x, y, \ldots$ for first-sort variables and Greek letters $\alpha, \beta, \ldots, \mu, \nu, \ldots$ for second-sort variables. A formula $\varphi(\bar{x}, \bar{\mu})$, where $\bar{x}$ is a tuple of r first-sort variables and $\bar{\mu}$ is a tuple of s second-sort variables, consequently defines an (r_1, r_2)-ary relation $R \subseteq A_1^{r_1} \times A_2^{r_2}$.

2.2.5 Counting extensions

The logics we have considered so far lack the ability to count the numbers of elements of arbitrarily large definable sets in general. This can, to different extents, be overcome by adding suitable counting constructs. One such extension which is investigated in Chapter 3 is the extension of monadic second-order logic by new quantifiers that can specify the size of definable sets modulo fixed integers. Another more powerful extension is obtained by introducing counting terms $\#x\,.\,\varphi(x)$ defining the number of elements satisfying φ. To allow for a sensible interpretation of such counting terms, this extension is usually studied on particular two-sorted structures where the first sort is the actual structure of interest and the second sort is a linearly ordered set of at least the same size. Then every element α of the second sort represents the number $k = |\{\beta : \beta < \alpha\}|$, which yields a natural interpretation of counting terms by elements of the second sort.

In particular, we associate with a τ-structure $\mathfrak{A} = (A, R_1, \ldots, R_n)$ the canonical two-sorted $\tau \cup \{<\}$-structure $\mathfrak{A}^* = (A, R_1, \ldots, R_n) \uplus (N_A, <)$ where $N_A := \{0, \ldots, |A|\}$, and $<$ is the usual linear order on the natural numbers restricted to N_A. In these special cases, we will frequently refer to the first sort as the *vertex sort*, and to the second sort as the *number sort*.

2.2.6 Queries and Expressive Power

We have discussed that a model class is an isomorphism-closed class of τ-structures. Given a τ-sentence φ of some logic $\mathcal{L}$, we denote by $\mathrm{Mod}(\varphi)$ the model class of all τ-structures satisfying φ. Furthermore, a τ-formula with k free variables defines a k-ary *global relation* or *query*

$$Q : \mathrm{Str}[\tau] \to \{(a_1, \ldots, a_k) : \mathfrak{A} \vDash \varphi(\bar{a})\} =: \varphi^{\mathfrak{A}} \quad .$$

If φ is a sentence, we obtain a 0-ary or Boolean query mapping each structure to the empty set $\varnothing$ (or *false*) or to the set containing the empty tuple $\{\square\}$ (or *true*) such that $\mathrm{Mod}(\varphi) = \{\mathfrak{A} \in \mathrm{Str}[\tau] : Q(\mathfrak{A}) = \{\square\}\}$. A prominent example occurring frequently in this thesis is the Boolean query EVEN selecting structures over a universe with an even number of elements. The notions of queries or classes of models allows to compare the expressive power of logics by specifying that a logic $\mathcal{L}'$ *is at least as expressive as* $\mathcal{L}$, denoted $\mathcal{L} \subseteq \mathcal{L}'$, if, for every sentence $\varphi \in \mathcal{L}[\tau]$, there exists a sentence $\varphi' \in \mathcal{L}'[\tau]$ such that φ and φ' define the same query, i.e. $\mathrm{Mod}(\varphi) = \mathrm{Mod}(\varphi')$.

Further, a logic $\mathcal{L}$ gives rise to an equivalence relation on τ-structures by virtue of

$$\mathfrak{A} \equiv^{\mathcal{L}} \mathfrak{B} \;:\Longleftrightarrow\; \mathfrak{A} \in \mathrm{Mod}(\varphi) \text{ iff } \mathfrak{B} \in \mathrm{Mod}(\varphi) \text{ for all } \varphi \in \mathcal{L}[\tau] \quad .$$

pointing out that no sentence of the logic can possibly distinguish structure $\mathfrak{A}$ from $\mathfrak{B}$. Especially in Chapter 3, we will study these equivalences and tools for deriving them in more detail.

2.3 Interpretations

Interpretations are a powerful tool for establishing decidability or undecidability of theories but also for obtaining definability results.

Definition 2.1. Let $\mathcal{L}$ be a logic, let σ and τ be relational signatures, and let n be a positive integer. An n-dimensional $\mathcal{L}(\sigma, \tau)$-interpretation $\mathfrak{I}$ consists of the following:

(1) a formula $\delta(x_1, \ldots, x_n) \in \mathcal{L}[\sigma]$, called the *domain formula*,
(2) a formula $\varepsilon(x_1, \ldots, x_n, y_1, \ldots, y_n) \in \mathcal{L}[\sigma]$, defining equality of representing elements

(3) formulae $\varphi_R(\overline{x}_1, \dots, \overline{x}_r) \in \mathcal{L}[\sigma]$, for each $R \in \tau$, where r is the arity of R and each $\overline{x}_i$ is a tuple of n variables, defining the relations of τ.

With every interpretation $\mathfrak{I}$, we can associate FO-definable *admissibility conditions* stating that ε defines a congruence relation on the set of elements satisfying δ with respect to the relations defined by the formulae φ_R. For any σ-structure $\mathfrak{A}$ satisfying the admissibility conditions, the interpretation $\mathfrak{I}$ defines the canonical τ-structure $\mathfrak{I}(\mathfrak{A}) := \left(\delta^{\mathfrak{A}}, (\varphi_R^{\mathfrak{A}})_{R\in\tau}\right)/\varepsilon^{\mathfrak{A}}$, and we say that a τ-structure $\mathfrak{B}$ *is interpreted in* $\mathfrak{A}$ if $\mathfrak{B} \cong \mathfrak{I}(\mathfrak{A})$.

Furthermore, an n-dimensional $\mathcal{L}(\sigma, \tau)$-interpretation $\mathfrak{I}$ defines a translation of $\mathcal{L}[\tau]$-formulae φ into $\mathcal{L}[\sigma]$-formulae $\mathfrak{I}(\varphi)$ obtained by replacing each atomic formula $x = y$ with $\varepsilon(\overline{x}, \overline{y})$, each atom $R(x_1, \dots, x_r)$ with $\varphi_R(\overline{x}_1, \dots, \overline{x}_r)$, and by relativising and expanding the quantifiers, i.e., more precisely, by replacing

- $\forall x \varphi(x)$ with $\forall x_1, \dots, x_n(\delta(\overline{x}) \to \mathfrak{I}(\varphi)(\overline{x}))$, and
- $\exists x \varphi(x)$ with $\exists x_1, \dots, x_n(\delta(\overline{x}) \wedge \mathfrak{I}(\varphi)(\overline{x}))$.

These two translations induced by an interpretation are linked according to the following proposition.

Proposition 2.2. Let $\mathfrak{I}$ be an $\mathcal{L}(\sigma, \tau)$-interpretation. For any σ-structure $\mathfrak{A}$ satisfying the admissibility conditions of $\mathfrak{I}$, and any formula $\varphi \in \mathcal{L}[\tau]$, we have

$$\mathfrak{I}(\mathfrak{A}) \vDash \varphi \quad \text{if and only if} \quad \mathfrak{A} \vDash \mathfrak{I}(\varphi) \ .$$

The concept of interpreting structures directly translates into the two-sorted setting by modifying the definition such as to allow for the specification of two domain formulae, one for each sort, and two formulae defining equality. Thus, an $\mathcal{L}(\sigma, \tau)$-interpretation of a two-sorted structure in another two-sorted structure is specified by two domain formulae $\delta_1(x_1, \dots, x_{n_1}, \alpha_1, \dots, \alpha_{n_2})$ and $\delta_2(x_1, \dots, x_{m_1}, \alpha_1, \dots, \alpha_{m_2})$, i.e. each first-sort element of the interpreted structure is represented by a tuple of n_1 first-sort and n_2 second-sort elements, and each second-sort element is represented by m_1 first-sort and m_2 second-sort elements, two formulae defining equality $\varepsilon_1(\overline{x}, \overline{\alpha}, \overline{y}, \overline{\beta})$ and $\varepsilon_2(\overline{x}, \overline{\alpha}, \overline{y}, \overline{\beta})$, and, as before, formulae defining the relations of τ with the appropriate numbers and sorts of variables. Such interpretations will be frequently used in Chapter 4.

2.4 Descriptive Complexity Theory

The central aspect of descriptive complexity theory is the study of the relationship of the descriptive resources that are necessary for describing a property of structures in a logic to the computational resources needed for deciding that property. Since we only rely on high-level results, we do not give an introduction to details such as representing structures as input to Turing machines or representing classes of structures as decision problems. Further details can be found in [EF99, GKL$^+$07, Lib04].

Definition 2.3. A logic $\mathcal{L}$ *captures* a complexity class $\mathcal{C}$ on a domain $\mathcal{D}$ of finite structures if for all vocabularies τ and all $\mathcal{K} \subseteq \mathcal{D}[\tau]$,

$$\mathcal{K}_< \in \mathcal{C} \text{ if and only if there exists a sentence } \varphi \in \mathcal{L}[\tau] \text{ such that } \mathcal{K} = \{\mathfrak{A} \in \mathcal{D}[\tau] : \mathfrak{A} \models \varphi\} \ ,$$

where $\mathcal{K}_< = \{(\mathfrak{A}, <) : \mathfrak{A} \in \mathcal{K}, \text{ and } < \text{ is a linear order on } \mathfrak{A}\}$.

Note that the structures constituting the classes $\mathcal{K}$ in the definition are neither necessarily ordered nor unordered. On the other hand, containment in a complexity class is only well defined for a class of ordered structures since a Turing machine only decides properties of an ordered representative of an unordered structure. However, it must accept or reject an input structure $\mathfrak{A}$ for all possible orderings of its universe, i.e. the result of the computation must be order-invariant.

2.5 Games

We consider two-player games played on directed graphs. The underlying graph, called *arena*, is given by a triple $\mathcal{A} = (V, V_0, E)$ such that (V, E) is a directed graph and $V_0 \subseteq V$ is a subset of the vertices belonging to player 0. Since every position belongs either to player 0 or to player 1, we do not need to specify the set of vertices of player 1 explicitly, but it is understood that $V_1 := V \setminus V_0$. A *game* $\mathcal{G}(\mathcal{A}, \mathrm{Win}, v_0)$ is given by an arena $\mathcal{A}$, a winning condition Win, and an initial position v_o. It is played by moving a token starting from the initial position, and, depending on whether the current position belongs to player 0 or player 1, the respective player moves the token along an edge of the arena. This continues until one of the players gets stuck,

or, if there are no terminal vertices, it continues forever, and the generated finite or infinite path π is called a *play*. The winning condition specifies the subset of plays that are winning for player 0.

A *strategy* for player σ is a function $f_\sigma : V^* V_\sigma \to V$ that tells the player for each history of the play and at each of her positions where to move. The strategy f_σ is called a *winning strategy for player σ from position v* if player σ wins the game starting at v against her opponent by following the strategy f_σ no matter how her opponent plays. A strategy is called *positional* if it can be represented as a function $f_\sigma : V_\sigma \to V$, i.e. if the next move only depends on the current position but not on the history of the play. A *strategy with finite memory* is a function $f_\sigma : V_\sigma \times S \to V$ where S is a finite set of memory states. After every move, the memory is updated depending on the current position and the previous content according to a function $u : V \times S \to S$, and the next move of player σ playing according to f_σ is determined by the current position in the game and the current memory content.

In the case that the arena contains terminal positions, we usually consider a reachability winning condition such that the player who gets stuck loses. If the arena admits infinite plays, these are either considered as draws or declared as winning for one of the players.

A crucial property of the games we consider is *determinacy* which means that, from any position v, exactly one of the players has a winning strategy. Already in 1913, Zermelo proved, using the example of chess, the determinacy of finite reachability games

Theorem 2.4 (Zermelo). In every finite reachability game, from each position exactly one of the players has a strategy to win or both players have a strategy to achieve a draw.

The games we consider in Chapter 5 belong to the class of infinite games. In these games, all plays are infinite which is achieved by prohibiting terminal positions in the arena. In general, the set of possible plays of an infinite game is uncountable, so the specification of a winning condition can already be very complex. However, rather simple winning conditions suffice for our purposes; we will consider Büchi games, parity games, and Muller games. Their respective winning conditions are specified in terms of an arena that is extended by a function $\Omega : V \to C = \{0, \dots, d\}$ assigning priorities (also referred to as colours) from a finite set C to the positions and

the set $\inf(\Omega(\pi))$ denoting the set of priorities that are seen infinitely often during the play π as follows.

- The Büchi winning condition is defined by a set $F \subseteq C$ such that $\text{Win} = \{\pi \in V^\omega : \inf(\Omega(\pi)) \cap F \neq \varnothing\}$.
- The parity winning condition is specified by $\text{Win} = \{\pi \in V^\omega : \min\{\inf(\Omega(\pi))\} \text{ is even}\}$.
- The Muller winning condition is given by a family $\mathcal{F} \subseteq \mathcal{P}(C)$ such that $\text{Win} = \{\pi \in V^\omega : \inf(\Omega(\pi)) \in \mathcal{F}\}$.

These winning conditions are very simple from a topological point of view, in fact all these games are Borel games, and hence they are determined as shown by Martin [Mar75].

For parity and Muller games, there are even stronger determinacy results regarding the complexity of the winning strategies.

Theorem 2.5 (Emerson & Jutla [EJ91]). Every parity game is determined in positional strategies.

Theorem 2.6 (Gurevich & Harrington [GH82], and McNaughton [McN93]). Every Muller game is determined in finite memory strategies.

Büchi games also enjoy positional determinacy since they can easily be translated into parity games with only two priorities. For a direct proof see [GTW02].

Part I

Order Invariance and Local Orderings

3 ORDER-INVARIANCE VS. COUNTING IN MSO

Certain properties are easily definable on structures that are equipped with particular relations. For example, on structures with a successor relation and on linearly ordered structures, we can easily express in MSO that the cardinality of the universe is even by stating that the set containing every other element of the order starting with the smallest one does not contain the largest element. We will see later that MSO fails to define this query (EVEN) on the class of plain sets. On the other hand, the property itself is not tied to ordered structures, and it is especially not a property of the order. Rather, since we are not interested in the actual elements contained in the set described above, the truth of the formula does not depend on a particular linear order at all.

Intuitively, whenever a property is definable on a class of ordered structures such that the formula does not make a statement about the order but rather uses the order without depending on a particular interpretation, we say that the property is *order-invariantly definable*. This concept is made more precise in the following definition.

Definition 3.1. Let τ be a relational vocabulary, and let $\varphi \in \mathcal{L}[\tau \dot\cup \{<\}]$, i.e. φ may contain an additional relation symbol $<$. Then φ is called *order-invariant on a class $\mathcal{K}$ of τ-structures* if and only if $(\mathfrak{A}, <_1) \vDash \varphi \iff (\mathfrak{A}, <_2) \vDash \varphi$ for all $\mathfrak{A} \in \mathcal{K}$ and all linear orders $<_1$ and $<_2$ on A.

Considering order-invariant MSO, Courcelle investigated the relationship between order-invariant MSO and *counting* MSO (short CMSO), an

extension of MSO that allows to reason about cardinalities of sets modulo fixed integers. He showed that order-invariant MSO collapses to CMSO on the class of trees [Cou96], conjecturing that order-invariant MSO is strictly stronger than CMSO for general graphs [Cou96, Conjecture 7.3]. Further, results by Lapoire [Lap98] are claimed to imply the collapse of $\mathrm{MSO}_{<\text{-inv}}$ to CMSO on all classes of bounded tree width.

This chapter is devoted to the proof of Courcelle's conjecture showing that this collapse does not occur in general. Towards this goal, we first introduce an Ehrenfeucht-Fraïssé game capturing equivalence of structures with respect to CMSO formulae and present a class of grid-like structures that is definable in order-invariant MSO but not in CMSO.

3.1 Counting MSO

The notion of (modulo-)counting monadic second-order logic (CMSO) can be introduced in two different, but nonetheless equivalent, ways. The first is to regard CMSO as an extension of MSO by modulo-counting first-order quantifiers.

Definition 3.2. Let τ be a signature and $M \subseteq \mathbb{N}^+$ a set of moduli, then

- every formula $\varphi \in \mathrm{MSO}[\tau]$ is also a formula in $\mathrm{CMSO}^{(M)}[\tau]$, and
- if $\varphi(x) \in \mathrm{CMSO}^{(M)}[\tau]$ and $m \in M$, then $\exists^{(m)} x.\varphi(x) \in \mathrm{CMSO}^{(M)}[\tau]$.

If we do not restrict the set of modulo-counting quantifiers being used, we obtain the full language $\mathrm{CMSO}[\tau] = \mathrm{CMSO}^{(\mathbb{N}^+)}[\tau]$. The semantics of plain MSO formulae is as expected, and we have $\mathfrak{A} \vDash \exists^{(m)} x.\varphi(x)$ if and only if $|\{a \in A : \mathfrak{A} \vDash \varphi(a)\}| \equiv 0 \pmod{m}$. The quantifier rank $\mathrm{qr}(\psi)$ of a $\mathrm{CMSO}[\tau]$ formula ψ is defined as for MSO-formulae with the additional rule that $\mathrm{qr}\left(\exists^{(m)} x.\varphi(x)\right) = 1 + \mathrm{qr}(\varphi)$, i.e. we do not distinguish between different kinds of quantifiers.

Here we use an alternative but equivalent definition of CMSO, namely the extension of the MSO language by monadic second-order predicates $C^{(m)}$ which hold true of a set X if and only if $|X| \equiv 0 \pmod{m}$. As in the definition above, formulae of the fragment $\mathrm{CMSO}^{(M)}[\tau]$ may only use predicates $C^{(m)}$ where $m \in M$. The back-and-forth translation can be carried out along the following equivalences which increase the quantifier rank by

at most one in each step:

$$\exists^{(m)} x.\varphi(x) \equiv \exists X(C^{(m)}(X) \wedge \forall x(Xx \leftrightarrow \varphi(x))) \quad \text{and}$$
$$C^{(m)}(X) \equiv \exists^{(m)} x.Xx \ .$$

Hence, if we obtain ψ' as the translation of a formula ψ, then $\mathrm{qr}(\psi') \leq 2 \cdot \mathrm{qr}(\psi)$.

Furthermore, the introduction of additional predicates $C^{(m,r)}$ (or, equivalently, additional modulo-counting quantifiers $\exists^{(m,r)}$) stating for a set X that $|X| \equiv r \pmod m$ does not increase the expressive power since they can be simulated as follows (increasing the quantifier rank by $r+1$):

$$C^{(m,r)}(X) \equiv \exists X_0(\text{“}X_0 \subseteq X\text{”} \wedge \text{“}|X_0| = r\text{”} \wedge \text{“}C^{(m)}(X \setminus X_0)\text{”}),$$

where all subformulae are easily expressible in MSO.

It is an easy observation that, generalising the idea of how to express EVEN on ordered structures, every CMSO formula is equivalent over the class of finite structures to an order-invariant MSO formula.

Proposition 3.3. $\mathrm{CMSO} \subseteq \mathrm{MSO}_{<\text{-inv}}$.

Proof. A modulo-counting quantifier $\exists^{(q)}$ can be translated in the following way, stating that the set of elements satisfying φ can be partitioned into q subsets $X_0, \ldots, X_{q-1}$, each containing every q-th element satisfying φ, such that X_0 contains the minimal and X_{q-1} contains the maximal element satisfying φ:

$$\exists^{(q)} x.\varphi(x) := \exists X \exists X_0 \ldots \exists X_{q-1} \left(\begin{array}{l} \forall x\,(Xx \leftrightarrow \varphi(x)) \wedge \text{“}\{X_0, \ldots, X_{q-1}\} \text{ is a partition of } X\text{”} \\ \wedge \exists x\big(X_0 x \wedge \forall y(Xy \rightarrow x \leq y)\big) \\ \wedge \exists x\big(X_{q-1} x \wedge \forall y(Xy \rightarrow x \geq y)\big) \\ \wedge \forall x \forall y \left(S_{\varphi,<}(x,y) \rightarrow \left(\bigwedge_{i=0}^{q-1} X_i x \leftrightarrow X_{i+1 \pmod q} y \right) \right) \end{array} \right)$$

where $S_{\varphi,<}$ defines the successor relation on the subset of elements defined by φ that is induced by an arbitrary order $<$ on the universe of the structure.

Note however, that the quantifier rank of the translated formula grows depending on the parameter in the modulo-counting quantifier. Q.E.D.

3.2 The Ehrenfeucht-Fraïssé method for CMSO

In this section, we develop the tools to prove inexpressibility results on counting MSO by means of the Ehrenfeucht-Fraïssé method, which provides a game-theoretic characterisation of the indistinguishability of structures with respect to CMSO sentences.

Definition 3.4. Two τ-structures $\mathfrak{A}$ and $\mathfrak{B}$ are called *rank r/mod M equivalent*, $\mathfrak{A} \equiv_r^M \mathfrak{B}$, if $\mathfrak{A} \models \varphi$ if and only if $\mathfrak{B} \models \varphi$ for all $\varphi \in \mathrm{CMSO}^{(M)}[\tau]$ with $\mathrm{qr}(\varphi) \leq r$.

The Ehrenfeucht-Fraïssé game capturing the expressiveness of MSO parametrised by the quantifier-rank (cf. [EF99, Lib04]) can be naturally extended to a game capturing the expressiveness of CMSO parametrised by the quantifier rank and the set of moduli being used in the cardinality predicates or counting quantifiers.

Viewing CMSO as MSO with additional quantifiers $\exists^{(m)} x.\varphi(x)$ for all m in a fixed set M leads to a new type of move which is also described by Nurmonen in the context of extending FO by modulo-counting quantifiers [Nur00]. Since a modulo-counting quantifier actually combines notions of a first-order and a monadic second-order quantifier in the sense that it makes a statement about the cardinality of a certain *set* of elements, but, on the other hand, behaves like a first-order quantifier binding an *element* variable and making a statement about that particular element, the move capturing modulo-counting quantification consists of two phases. First, Spoiler chooses one of the structures $\mathfrak{A}$ or $\mathfrak{B}$, say $\mathfrak{A}$, where he selects a set of elements $S \subseteq A$. Duplicator answers with a set $D \subseteq B$ such that $|S| \equiv |D| \pmod{M}$. In the second phase of the move, Spoiler and Duplicator select elements $b \in B$ and $a \in A$, respectively, such that $a \in S$ if and only if $b \in D$. After the move, reflecting the first-order nature of the quantifier, only the two selected elements a and b are remembered and contribute to the next position in the game, whereas the information about the chosen sets is discarded.

Due to the huge effort an additional move demands from descriptions of winning strategies for the two players, we prefer the approach to CMSO via second-order cardinality predicates, which yields an Ehrenfeucht-Fraïssé game that allows a much clearer description of winning strategies. Since we do not use additional quantifiers, we get along with exactly the same types of

moves as in the Ehrenfeucht-Fraïssé game for MSO, and instead we modify the winning condition to take the new predicates into account.

Towards this end, we first introduce a suitable concept of partial isomorphisms between structures.

Definition 3.5. With any structure $\mathfrak{A}$ and any set $M \subseteq_{\text{fin}} \mathbb{N}^+$ we associate the (first-order) power set structure $\mathfrak{A}^M := \big(\mathcal{P}(A), (C^{(m,r)})_{\substack{m\in M \\ 0\leq r<m}}\big)$, where the predicates $C^{(m,r)}$ are interpreted by the set of those elements $x \in \mathcal{P}(A)$ such that $|x| \equiv r \pmod m$. (Note that first-order predicates in the power set structure $\mathfrak{A}^M$ naturally correspond to second-order predicates in $\mathfrak{A}$.)

Let $\mathfrak{A}$ and $\mathfrak{B}$ be τ-structures, and let $M \subseteq_{\text{fin}} \mathbb{N}^+$ be a fixed set of moduli. Then the mapping $(A_1, \dots, A_s, a_1, \dots, a_t) \mapsto (B_1, \dots, B_s, b_1, \dots, b_t)$ is called a *twofold partial isomorphism between $\mathfrak{A}$ and $\mathfrak{B}$ with respect to M* if

(1) $(a_1, \dots, a_t) \mapsto (b_1, \dots, b_t)$ is a partial isomorphism between the expansions $(\mathfrak{A}, A_1, \dots, A_s)$ and $(\mathfrak{B}, B_1, \dots, B_s)$ and
(2) $(A_1, \dots, A_s) \mapsto (B_1, \dots, B_s)$ is a partial isomorphism between $\mathfrak{A}^M$ and $\mathfrak{B}^M$.

3.2.1 The Ehrenfeucht-Fraïssé game

We propose the following Ehrenfeucht-Fraïssé game to capture the expressiveness of CMSO where the use of moduli is restricted to a (finite) set M and formulae of quantifier rank at most r.

Definition 3.6 (Ehrenfeucht-Fraïssé game for CMSO)**.** Let $M \subseteq_{\text{fin}} \mathbb{N}^+$ and $r \in \mathbb{N}$. The r-round (mod M) Ehrenfeucht-Fraïssé game $\mathcal{G}_r^M(\mathfrak{A}, \mathfrak{B})$ is played by Spoiler and Duplicator on τ-structures $\mathfrak{A}$ and $\mathfrak{B}$. In each turn, Spoiler can choose between the following types of moves.

- *Point move*: Spoiler selects an element in one of the structures, and Duplicator answers by selecting an element in the other structure.
- *Set move*: Spoiler selects a set of elements X in one of the structures, and Duplicator responds by choosing a set of elements Y in the other structure.

After $r = s + t$ rounds, when the players have chosen pairs of sets $(A_1, B_1), \dots (A_s, B_s)$ as well as pairs of elements $(a_1, b_1), \dots, (a_t, b_t)$ in the two structures $\mathfrak{A}$ and $\mathfrak{B}$, Duplicator wins the game if and only if

$(A_1, \dots, A_s, a_1, \dots, a_t) \mapsto (B_1, \dots, B_s, b_1, \dots, b_t)$ is a twofold partial isomorphism between $\mathfrak{A}$ and $\mathfrak{B}$ with respect to M.

First note that, although Duplicator is required to answer a set move X by a set Y such that $|X| \equiv |Y| \pmod M$ in order to win, we do not have to make this explicit in the rules of the moves since these cardinality constraints are already imposed by the winning condition (X and Y would not define a twofold partial isomorphism if they did not satisfy the same cardinality predicates). Furthermore, for $M = \varnothing$ or $M = \{1\}$, the resulting game $\mathcal{G}_r^M(\mathfrak{A}, \mathfrak{B})$ corresponds exactly to the usual Ehrenfeucht-Fraïssé game for MSO.

3.2.2 Rank r/mod M-Types

A *rank r/mod M-CMSO type T* in s free second-order variables and t free first-order variables is a maximally consistent set of $\mathrm{CMSO}^{(M)}$ formulae $\varphi(\overline{X}, \overline{x})$ of quantifier rank at most r. Accordingly, the *rank r/mod M-type* of the tuples $\overline{U}, \overline{a}$ in a structure $\mathfrak{A}$ is defined as

$$\mathrm{tp}^{M,r}_{\mathfrak{A}}(\overline{U}, \overline{a}) = \{\varphi(\overline{X}, \overline{x}) : \varphi \in \mathrm{CMSO}^{(M)}[\tau],\ \mathrm{qr}(\varphi) \le r,\ (\mathfrak{A}, \overline{U}, \overline{a}) \vDash \varphi\} \ .$$

By definition, $(\mathfrak{A}, \overline{U}, \overline{a}) \equiv^M_r (\mathfrak{B}, \overline{V}, \overline{b})$ if and only if $\mathrm{tp}^{M,r}_{\mathfrak{A}}(\overline{U}, \overline{a}) = \mathrm{tp}^{M,r}_{\mathfrak{B}}(\overline{V}, \overline{b})$.

Definition 3.7. Let $\overline{X} = (X_1, \dots, X_s)$ and $\overline{x} = (x_1, \dots, x_t)$ be sequences of free second- and first-order variables, respectively. The rank r/mod M-isomorphism type induced by $\overline{U}, \overline{a}$ in $\mathfrak{A}$ is defined as

$$\begin{aligned}
\varphi^{M,0}_{\mathfrak{A},\overline{U},\overline{a}}(\overline{X}, \overline{x}) &= \bigwedge \{\varphi : \varphi \in \mathrm{tp}^{M,0}_{\mathfrak{A}}(\overline{U}, \overline{a})\} \\
\varphi^{M,r+1}_{\mathfrak{A},\overline{U},\overline{a}}(\overline{X}, \overline{x}) &= \bigwedge_{a \in A} \exists x_{t+1} \varphi^{M,r}_{\mathfrak{A},\overline{U},\overline{a}a}(\overline{X}, \overline{x} x_{t+1}) \\
&\quad \wedge \forall x_{t+1} \bigvee_{a \in A} \varphi^{M,r}_{\mathfrak{A},\overline{U},\overline{a}a}(\overline{X}, \overline{x} x_{t+1}) \\
&\quad \wedge \bigwedge_{U \subseteq A} \exists X_{s+1} \varphi^{M,r}_{\mathfrak{A},\overline{U}U,\overline{a}}(\overline{X} X_{s+1}, \overline{x}) \\
&\quad \wedge \forall X_{s+1} \bigvee_{U \subseteq A} \varphi^{M,r}_{\mathfrak{A},\overline{U}S,\overline{a}}(\overline{X} X_{s+1}, \overline{x}) \ .
\end{aligned}$$

Note that (up to equivalence) $\mathrm{tp}^{M,0}_{\mathfrak{A}}(\overline{U}, \overline{a})$ contains only finitely many atomic and negated atomic formulae such that the conjunction in the definition above is finite. As we will prove in the following, the formula $\varphi^{M,r}_{\mathfrak{A},\overline{U},\overline{a}}(\overline{X}, \overline{x})$ intuitively describe those pairs of tuples $(\overline{V}, \overline{b})$ that are sufficiently similar to $(\overline{U}, \overline{a})$ such that Duplicator would win an r-round Ehrenfeucht-Fraïssé game starting from the position described by the tuples. We begin by proving the rather trivial fact that $\varphi^{M,r}_{\mathfrak{A},\overline{U},\overline{a}}$ indeed says that $\overline{U}, \overline{a}$ looks similar enough to itself.

Proposition 3.8. For all structures $\mathfrak{A}$, $M \subseteq_{\mathrm{fin}} \mathbb{N}^+$, $r \in \mathbb{N}$, and tuples $\overline{U}$ and $\overline{a}$ in $\mathfrak{A}$

(1) $\mathrm{qr}(\varphi^{M,r}_{\mathfrak{A},\overline{U},\overline{a}}(\overline{X}, \overline{x})) = r$, and
(2) $(\mathfrak{A}, \overline{U}, \overline{a}) \models \varphi^{M,r}_{\mathfrak{A},\overline{U},\overline{a}}(\overline{X}, \overline{x})$.

Proof. (1) Obviously follows from the definition.

(2) For $r = 0$, $\varphi^{M,0}_{\mathfrak{A},\overline{U},\overline{a}}(\overline{X}, \overline{x})$ is the conjunction of all $\varphi \in \mathrm{tp}^{M,0}_{\mathfrak{A}}(\overline{U}, \overline{a})$, and hence, by definition of the rank r/mod M-type, $\mathfrak{A} \models \varphi^{M,0}_{\mathfrak{A},\overline{U},\overline{a}}(\overline{U}, \overline{a})$. For $r > 0$, $\mathfrak{A} \models \varphi^{M,r}_{\mathfrak{A},\overline{U},\overline{a}}(\overline{U}, \overline{a})$ holds if

- $\mathfrak{A} \models \bigwedge_{a \in A} \exists x_{t+1} \varphi^{M,r-1}_{\mathfrak{A},\overline{U},\overline{a}a}(\overline{U}, \overline{a}x_{t+1})$,
- $\mathfrak{A} \models \forall x_{t+1} \bigvee_{a \in A} \varphi^{M,r-1}_{\mathfrak{A},\overline{U},\overline{a}a}(\overline{U}, \overline{a}x_{t+1})$,
- $\mathfrak{A} \models \bigwedge_{U \subseteq A} \exists X_{s+1} \varphi^{M,r-1}_{\mathfrak{A},\overline{U}U,\overline{a}}(\overline{U}X_{s+1}, \overline{a})$, and
- $\mathfrak{A} \models \forall X_{s+1} \bigvee_{U \subseteq A} \varphi^{M,r-1}_{\mathfrak{A},\overline{U}U,\overline{a}}(\overline{U}X_{s+1}, \overline{a})$.

Assuming that the claim is true for $r - 1$, the first formula holds if we choose, in each conjunct, the element a for x_{t+1}, and for each a chosen for x_{t+1} in the second formula the disjunct for this particular a is true. The same reasoning applies for the remaining two formulae. Hence, by induction, $\mathfrak{A} \models \varphi^{M,r}_{\mathfrak{A},\overline{U},\overline{a}}(\overline{U}, \overline{a})$ for all $r \in \mathbb{N}$. Q.E.D.

Theorem 3.9. The following are equivalent:

(1) $(\mathfrak{A}, \overline{U}, \overline{a}) \equiv^M_r (\mathfrak{B}, \overline{V}, \overline{b})$
(2) $(\mathfrak{B}, \overline{V}, \overline{b}) \models \varphi^{M,r}_{\mathfrak{A},\overline{U},\overline{a}}(\overline{X}, \overline{x})$
(3) Duplicator wins $\mathcal{G}^M_r(\mathfrak{A}, \overline{U}, \overline{a}, \mathfrak{B}, \overline{V}, \overline{b})$.

Proof. (1) $\Rightarrow$ (2) follows from Proposition 3.8.

(2) $\Leftrightarrow$ (3): Let $r = 0$. $(\mathfrak{B}, \overline{V}, \overline{b}) \models \varphi^{M,0}_{\mathfrak{A},\overline{U},\overline{a}}(\overline{X}, \overline{x})$ if and only if $(\mathfrak{B}, \overline{V}, \overline{b}) \models \varphi$ for all $\varphi \in \mathrm{tp}^{M,0}_{\mathfrak{A}}(\overline{U}, \overline{a})$, if and only if $(\overline{U}, \overline{a}) \mapsto (\overline{V}, \overline{b})$ is a twofold partial isomorphism between $\mathfrak{A}$ and $\mathfrak{B}$, if and only if Duplicator wins $\mathcal{G}^M_0(\mathfrak{A}, \overline{U}, \overline{a}, \mathfrak{B}, \overline{V}, \overline{b})$.

$r > 0$: $(\mathfrak{B}, \overline{V}, \overline{b}) \models \varphi^{M,r}_{\mathfrak{A},\overline{U},\overline{a}}(\overline{X}, \overline{x})$

iff $\mathfrak{B} \models \bigwedge_{a\in A} \exists x_{t+1} \varphi^{M,r-1}_{\mathfrak{A},\overline{U},\overline{a}a}(\overline{V}, \overline{b}x_{t+1})$,
$\mathfrak{B} \models \forall x_{t+1} \bigvee_{a\in A} \varphi^{M,r-1}_{\mathfrak{A},\overline{U},\overline{a}a}(\overline{V}, \overline{b}x_{t+1})$,
$\mathfrak{B} \models \bigwedge_{U\subseteq A} \exists X_{s+1} \varphi^{M,r-1}_{\mathfrak{A},\overline{U}U,\overline{a}}(\overline{V}X_{s+1}, \overline{b})$, and
$\mathfrak{B} \models \forall X_{s+1} \bigvee_{U\subseteq A} \varphi^{M,r-1}_{\mathfrak{A},\overline{U}U,\overline{a}}(\overline{V}X_{s+1}, \overline{b})$

iff for all moves $a \in A$ of Spoiler, Duplicator finds a $b \in B$, and
for all moves $b \in B$ of Spoiler, Duplicator finds an $a \in A$
such that she wins the game $\mathcal{G}^M_{r-1}(\mathfrak{A}, \overline{U}, \overline{a}a, \mathfrak{B}, \overline{V}, \overline{b}b)$, and
for all moves $U \in A$ of Spoiler, Duplicator finds a $V \in B$, and
for all moves $V \in B$ of Spoiler, Duplicator finds a $U \in A$
such that she wins the game $\mathcal{G}^M_{r-1}(\mathfrak{A}, \overline{U}U, \overline{a}, \mathfrak{B}, \overline{V}V, \overline{b})$.

(3) $\Rightarrow$ (1): For $r = 0$, Duplicator wins the game $\mathcal{G}^M_0(\mathfrak{A}, \overline{U}, \overline{a}, \mathfrak{B}, \overline{V}, \overline{b})$ if $(\overline{U}, \overline{a}) \mapsto (\overline{V}, \overline{b})$ is a twofold partial isomorphism between $\mathfrak{A}$ and $\mathfrak{B}$. Hence, $(\mathfrak{A}, \overline{U}, \overline{a})$ and $(\mathfrak{B}, \overline{V}, \overline{b})$ agree on all quantifier free formulae, and $(\mathfrak{A}, \overline{U}, \overline{a}) \equiv^M_0 (\mathfrak{B}, \overline{V}, \overline{b})$.

Let $r > 0$, and assume that Duplicator has a winning strategy in the game $\mathcal{G}^M_r(\mathfrak{A}, \overline{U}, \overline{a}, \mathfrak{B}, \overline{V}, \overline{b})$. Then for all $a \in A$ there is a $b \in B$ (and vice versa) such that she wins $\mathcal{G}^M_{r-1}(\mathfrak{A}, \overline{U}, \overline{a}a, \mathfrak{B}, \overline{V}, \overline{b}b)$. By induction hypothesis, $(\mathfrak{A}, \overline{U}, \overline{a}a) \equiv^M_{r-1} (\mathfrak{B}, \overline{V}, \overline{b}b)$. Consider a formula φ with $\mathrm{qr}(\varphi) = r$. Then φ is a Boolean combination of formulae that have a lower quantifier rank, or that are of the form $\exists x\psi(x)$, $\forall x\psi(x)$, $\exists X\psi(X)$, $\forall X\psi(X)$, where $\mathrm{qr}(\psi) = r - 1$. Hence, for any such $\psi(x)$, we have $(\mathfrak{A}, \overline{U}, \overline{a}a) \models \psi(x)$ if and only if $(\mathfrak{B}, \overline{V}, \overline{b}b) \models \psi(x)$ (and analogously for $\psi(X)$ and set moves). Thus, also $(\mathfrak{A}, \overline{U}, \overline{a}) \models \varphi$ if and only if $(\mathfrak{B}, \overline{V}, \overline{b}) \models \varphi$. Q.E.D.

Corollary 3.10. Let $\mathfrak{A}$ and $\mathfrak{B}$ be τ-structures, $r \in \mathbb{N}$, and $M \subseteq_{\mathrm{fin}} \mathbb{N}$. Then the following are equivalent:

(1) $\mathfrak{A} \equiv^M_r \mathfrak{B}$.
(2) Duplicator has a winning strategy in the r-round (mod M) Ehrenfeucht-Fraïssé game $\mathcal{G}^M_r(\mathfrak{A}, \mathfrak{B})$.

Lemma 3.11. Every formula $\varphi(\overline{X}, \overline{x}) \in \mathrm{CMSO}^{(M)}$ of quantifier rank r is equivalent to a disjunction of r-mod-M-isomorphism types, in particular

$$\vDash \varphi(\overline{X}, \overline{x}) \leftrightarrow \bigvee \{ \varphi^{M,r}_{\mathfrak{A},\overline{U},\overline{a}} : \mathfrak{A} \in \mathrm{Str}[\tau],\ \mathfrak{A} \vDash \varphi(\overline{U}, \overline{a}) \} \ .$$

Proof. First of all, the disjunction is finite since there are only finitely many non-equivalent formulae $\varphi^{M,0}_{\mathfrak{A},\overline{U},\overline{a}}$ as there are only finitely many non-equivalent (negated) atomic τ-formulae, and assuming that there are only finitely many non-equivalent formulae $\varphi^{M,r}_{\mathfrak{A},\overline{U},\overline{a}}$, there are, by definition, also only finitely many non-equivalent formulae $\varphi^{M,r}_{\mathfrak{A},\overline{U},\overline{a}}$.

Assume that $\mathfrak{B} \vDash \varphi(\overline{V}, \overline{b})$, then $\varphi^{M,r}_{\mathfrak{B},\overline{V},\overline{b}}$ appears in the disjunction, and hence, $\mathfrak{B} \vDash \bigvee \ldots$. Conversely, assume $\mathfrak{B} \vDash \bigvee \ldots$. Then $\mathfrak{B} \vDash \varphi^{M,r}_{\mathfrak{A},\overline{U},\overline{a}}$ for some $\mathfrak{A}$, $\overline{U} \subseteq A$, and $\overline{a} \in A$ with $\mathfrak{A} \vDash \varphi(\overline{U}, \overline{a})$. By Theorem 3.9, $(\mathfrak{B}, \overline{V}, \overline{b}) \equiv^M_r (\mathfrak{A}, \overline{U}, \overline{a})$, and hence, also $\mathfrak{B} \vDash \varphi(\overline{V}, \overline{b})$. Q.E.D.

Then we obtain the following standard argument to prove non-definability results.

Theorem 3.12. A class $\mathcal{K}$ of τ-structures is not definable in CMSO if and only if, for every $r \in \mathbb{N}$ and every $M \subseteq_{\mathrm{fin}} \mathbb{N}^+$, there are τ-structures $\mathfrak{A}_{M,r} \in \mathcal{K}$ and $\mathfrak{B}_{M,r} \notin \mathcal{K}$ such that $\mathfrak{A}_{M,r} \equiv^M_r \mathfrak{B}_{M,r}$.

Proof. The "if" direction is straightforward. Considering the "only if" direction assume that there exist $r \in \mathbb{N}$ and $M \subseteq_{\mathrm{fin}} \mathbb{N}^+$ such that all $\mathfrak{A}, \mathfrak{B} \in \mathrm{Str}[\tau]$ that are neither both in $\mathcal{K}$ nor both not in $\mathcal{K}$ are not r-mod-M-equivalent. Then $\varphi := \bigvee \{ \varphi^{M,r}_{\mathfrak{A}} : \mathfrak{A} \in \mathcal{K} \}$ defines $\mathcal{K}$. Q.E.D.

3.3 CMSO on disjoint unions of structures

It is possible for many logics such as FO and MSO to deduce the theory of the disjoint union of two structures from the theories of the separate structures. This result, which is also one of the central tools exploited and investigated in Chapter 5, does also hold for counting MSO. Depending on the intended use there are two approaches towards proving it. The approach we are using later and which is also carried out by Courcelle [Cou90, Lemma 4.5] for CMSO is to specify a translation of sentences talking about the disjoint union of two structures into a Boolean combination of sentences each talking about

one of the individual structures. At this point we are rather interested in definability issues, so we use an Ehrenfeucht-Fraïssé game-based technique showing that winning strategies for Duplicator in separate games on two pairs of structures can be combined into a winning strategy on the pair of disjoint unions of the structures.

Lemma 3.13. Let $\mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{B}_1$, and $\mathfrak{B}_2$ be τ-structures such that $\mathfrak{A}_1 \equiv_r^M \mathfrak{B}_1$ and $\mathfrak{A}_2 \equiv_r^M \mathfrak{B}_2$. Then $\mathfrak{A}_1 \dot\cup \mathfrak{A}_2 \equiv_r^M \mathfrak{B}_1 \dot\cup \mathfrak{B}_2$.

Proof. Consider the game on $\mathfrak{A} := \mathfrak{A}_1 \dot\cup \mathfrak{A}_2$ and $\mathfrak{B} := \mathfrak{B}_1 \dot\cup \mathfrak{B}_2$. A Spoiler's point move in $\mathfrak{A}$ (resp., in $\mathfrak{B}$) is answered by Duplicator according to her winning strategy in either $\mathcal{G}_r^M(\mathfrak{A}_1, \mathfrak{B}_1)$ or $\mathcal{G}_r^M(\mathfrak{A}_2, \mathfrak{B}_2)$ depending on whether Spoiler chose and elements from A_1 or A_2 (resp, B_1 or B_2). A set move $S \subseteq A$ (analogous for $S \subseteq B$) is decomposed into two subsets $S_1 := S \cap A_1$ and $S_2 := S \cap A_2$, and is answered by Duplicator by the set $D := D_1 \cup D_2$ consisting of the sets D_1 and D_2 chosen according to her winning strategies as responses to S_1 and S_2 in the respective games $\mathcal{G}_r^M(\mathfrak{A}_1, \mathfrak{B}_1)$ and $\mathcal{G}_r^M(\mathfrak{A}_2, \mathfrak{B}_2)$.

Since A_1 and A_2 as well as B_1 and B_2 are disjoint, we have $|S| = |S_1| + |S_2|$ and $|D| = |D_1| + |D_2|$. Furthermore, as the sets D_1 and D_2 are chosen according to Duplicator's winning strategies in the games on $\mathfrak{A}_1$ and $\mathfrak{B}_1$, and $\mathfrak{A}_2$ and $\mathfrak{B}_2$, respectively, $|S_1| \equiv |D_1| \pmod M$ and $|S_2| \equiv |D_2| \pmod M$. Since $\equiv \pmod M$ is a congruence relation with respect to addition, we have that $|S| \equiv |D| \pmod M$. It is easily verified that the sets and elements chosen according to this strategy indeed define a twofold partial isomorphism between $\mathfrak{A}$ and $\mathfrak{B}$. Q.E.D.

As a direct corollary we obtain the following result that will be used in the inductive step in the forthcoming proofs and intuitively states that remembering the origin of elements does not render disjoint unions of indistinguishable components distinguishable.

Corollary 3.14. Let $\mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{B}_1$, and $\mathfrak{B}_2$ be τ-structures, such that $\mathfrak{A}_1 \equiv_r^M \mathfrak{B}_1$ and $\mathfrak{A}_2 \equiv_r^M \mathfrak{B}_2$. Then $(\mathfrak{A}_1 \dot\cup \mathfrak{A}_2, A_1) \equiv_r^M (\mathfrak{B}_1 \dot\cup \mathfrak{B}_2, B_1)$, where A_1 and B_1 denote the universes of $\mathfrak{A}_1$ and $\mathfrak{B}_1$, respectively.

Proof. We consider the following $\tau \dot\cup \{P\}$-expansions of the given structures: $\mathfrak{A}_1' := (\mathfrak{A}_1, A_1)$, $\mathfrak{B}_1' := (\mathfrak{B}_1, B_1)$, $\mathfrak{A}_2' := (\mathfrak{A}_2, \varnothing)$, and $\mathfrak{B}_2' := (\mathfrak{B}_2, \varnothing)$. It is immediate that

(1) $\mathfrak{A}_1 \equiv_r^M \mathfrak{B}_1$ implies $(\mathfrak{A}_1, A_1) \equiv_r^M (\mathfrak{B}_1, B_1)$, and

(2) $\mathfrak{A}_2 \equiv_r^M \mathfrak{B}_2$ implies $(\mathfrak{A}_2, \varnothing) \equiv_r^M (\mathfrak{B}_2, \varnothing)$

since Duplicator can obviously win the respective Ehrenfeucht-Fraïssé games on the expanded structures using the same strategies as in the games witnessing the equivalences on the left-hand side. The claim follows by applying the previous lemma to the $\tau \dot\cup \{P\}$-expansions. Q.E.D.

It is well known that MSO exhibits a certain weakness regarding the ability to specify cardinality constraints on sets, i.e. structures over an empty vocabulary.

Proposition 3.15. Let $\mathfrak{A}$ and $\mathfrak{B}$ be $\varnothing$-structures, and let $r \in \mathbb{N}$. Then $\mathfrak{A} \equiv_r \mathfrak{B}$ if $|A|, |B| \geq 2^r$.

A proof using Ehrenfeucht-Fraïssé games is presented, for example, by Libkin [Libo4, Proposition 7.12]. By adapting this proof, we show that basically the same holds for CMSO.

Lemma 3.16. Let $\mathfrak{A}$ and $\mathfrak{B}$ be $\varnothing$-structures, $M \subseteq_{\text{fin}} \mathbb{N}^+$, and $r \in \mathbb{N}$. Then $\mathfrak{A} \equiv_r^M \mathfrak{B}$ if $|A|, |B| \geq (2^{r+1} - 4)\operatorname{lcm}(M)$ and $|A| \equiv |B| \pmod M$.

Proof. We prove by induction on the number of rounds that Duplicator wins the (mod M) r-round Ehrenfeucht-Fraïssé game $\mathcal{G}_r^M(\mathfrak{A}, \mathfrak{B})$. For $r = 0$ and $r = 1$ the claim is obviously true. Let $r > 1$, assume that the claim holds for $r-1$, and consider the first move of the r-round game. We assume that Spoiler makes his move in $\mathfrak{A}$ since the reasoning in the other case is completely symmetric.

If Spoiler makes a set move $S \subseteq A$, we consider the following cases:

(1) $|S| < (2^r - 4)\cdot\operatorname{lcm}(M)$ (or $|A - S| < (2^r - 4)\cdot\operatorname{lcm}(M)$). Then Duplicator selects a set $D \subseteq B$ such that $|D| = |S|$ (or $|B - D| = |A - S|$), and hence $S \cong D$ and $A - S \equiv_{r-1}^M B - D$ (or $A - S \cong B - D$ and $S \equiv_{r-1}^M D$).

(2) $|S|, |A - S| \geq (2^r - 4)\cdot\operatorname{lcm}(M)$. Then Duplicator selects a set $D \subseteq B$ such that $|D| \equiv |S| \pmod M$ and $|D|, |B - D| \geq (2^r - 2)\cdot\operatorname{lcm}(M)$. In fact, she chooses for D half of the elements plus $\ell < \operatorname{lcm}(M)$ additional elements to fulfil the cardinality constraints $|D| \equiv |S| \pmod M$. Then, for the set $B - D$ of non-selected elements, we have

$$\begin{aligned} |B - D| &\geq \frac{1}{2}\big((2^{r+1} - 4)\operatorname{lcm}(M)\big) - \ell \\ &\geq (2^r - 2)\operatorname{lcm}(M) - \operatorname{lcm}(M) \geq (2^r - 4)\operatorname{lcm}(M) \end{aligned}$$

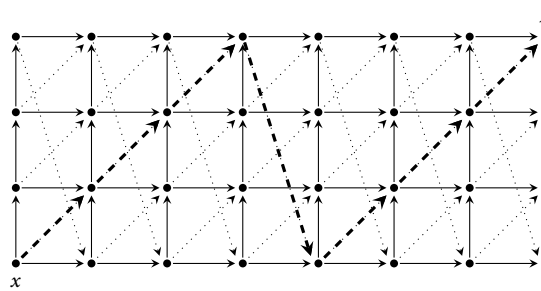

Figure 3.1. Grid with distinguished horizontal and vertical edges, a definable diagonal-edge relation

for all ℓ satisfying $0 \leq \ell < \mathrm{lcm}(M)$. Since $|D| = |B - D| + 2\ell$, obviously $|D| \geq (2^r - 4)\,\mathrm{lcm}(M)$ as well.

Thus, in both cases, by the induction hypothesis we get $S \equiv^M_{r-1} D$ and $A - S \equiv^M_{r-1} B - D$. Hence, by Corollary 3.14 $(A, S) \equiv^M_{r-1} (B, D)$, i.e. Duplicator has a winning strategy in the remaining $(r-1)$-round game from position (S, D).

If Spoiler makes a point move $s \in A$, Duplicator answers by choosing an arbitrary element $d \in B$. Similar to Case 1 above, we observe that $(\{s\}, s) \cong (\{d\}, d)$ and $A - \{s\} \equiv^M_{r-1} B - \{d\}$ by the induction hypothesis. Thus, by Lemma 3.13, $(A, s) \equiv^M_{r-1} (B, d)$ implying that Duplicator has a winning strategy for the remaining $r - 1$ rounds from position (s, d). Q.E.D.

3.4 The Separating Example

We will first give a brief description of our example showing that $\mathrm{MSO}_{<\text{-inv}}$ is strictly more expressive than CMSO. We consider a property of two-dimensional grids, namely that the number of rows divides the number of columns. This property is easily definable in MSO for grids that are given as directed graphs with two edge relations, one for the horizontal edges pointing rightwards, and one for the vertical edges pointing upwards, by defining a new relation of diagonal edges combining one step rightwards and one step upwards wrapping around from the top border to the bottom border but not from the right to the left border. Note that there is a path following those diagonal edges starting from the bottom-left corner of the

grid and ending in the top-right corner if and only if the vertical dimension divides the horizontal dimension of the grid (see Figure 3.1). Thus, for our purposes, we have to weaken the structure in the sense that we hide some information that remains accessible to $\mathrm{MSO}_{<\text{-inv}}$-formulae but not to CMSO formulae.

An appropriate loss of information is achieved by replacing the two edge relations with their reflexive symmetric transitive closure, i.e. to consider grids given as structures with two equivalence relations which provide a notion of *rows* and *columns*. Obviously, notions like corner and border vertices as well as the notion of an order on the rows and columns that were important for the MSO-definition of the divisibility property are lost, but clearly, all these notions can be regained in presence of an arbitrary linear order. First, the order allows us to uniquely define an element (e.g. the $<$-least element) to be the bottom-left corner of the grid, and second, the order induces successor relations on the set of columns as well as on the set of rows, from which both horizontal and vertical successors of any vertex can be deduced. Since the divisibility property is obviously invariant with respect to the ordering of the rows or columns, this allows for expressing it in $\mathrm{MSO}_{<\text{-inv}}$. The remainder of this section is devoted to developing the arguments showing that CMSO fails to express this property on the following class of grid-like structures.

Definition 3.17. A *cliquey* (k, ℓ)*-grid* is a $\{\sim_h, \sim_v\}$-structure that is isomorphic to $\mathfrak{G}_{k\ell} := (\{0, \dots, k-1\} \times \{0, \dots, \ell-1\}, \sim_h, \sim_v)$, where

$$\sim_h := \{((x, y), (x', y')) : x = x'\} \text{ and}$$
$$\sim_v := \{((x, y), (x', y')) : y = y'\},$$

i.e. $\sim_h$ consists of exactly k equivalence classes (called *rows*), each containing ℓ elements, and $\sim_v$ consists of exactly ℓ equivalence classes (called *columns*), each containing k elements, such that every equivalence class of $\sim_h$ intersects every equivalence class of $\sim_v$ in exactly one element and vice versa.

A *horizontally coloured cliquey* (k, ℓ)*-grid* is a $\{\sim_v, P_0, \dots, P_{k-1}\}$-structure that is isomorphic to

$$\mathfrak{G}_{k\ell} := (\{0, \dots, k-1\} \times \{0, \dots, \ell-1\}, \sim_v, P_0, \dots, P_{k-1})$$

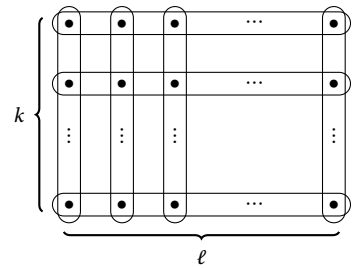

Figure 3.2. A cliquey (k, ℓ)-grid

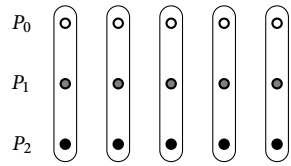

Figure 3.3. A horizontally coloured cliquey $(3, 5)$-grid

where

$$\sim_v := \{((x, y), (x', y')) : y = y'\} \text{ and}$$
$$P_i := \{i\} \times \{0, \ldots, \ell - 1\}, \text{for all } i \in \{0, \ldots, k - 1\} \ .$$

That is, a horizontally coloured cliquey (k, ℓ)-grid is obtained from the $\{\sim_v\}$-reduct of a cliquey (k, ℓ)-grid by introducing new unary predicates $\{P_0, \ldots, P_{k-1}\}$ (in the following referred to as *colours*), such that each set P_i corresponds to exactly one of the former $\sim_h$ equivalence classes.

The name originates from the initial intuition of replacing the paths defined by the vertical edges and the horizontal edges in the grid in Figure 3.1 by cliques. However, the formalisation via equivalence relations (the classes of which correspond to the mentioned cliques) allows for a more elegant exposition.

The class is first-order definable by a sentence ψ_{grid} stating that

- $\sim_v$ and $\sim_h$ are equivalence relations, and
- every pair consisting of one equivalence class of $\sim_h$ and $\sim_v$ each has exactly one element in common

as these properties are sufficient to enforce the desired grid-like structure.

Note that even the second property is first-order definable since every equivalence class is uniquely determined by any of its elements.

The following two lemmas justify the introduction of the notion of horizontally coloured cliquey grids for use in the forthcoming proofs.

Lemma 3.18. Let $\mathfrak{G}^{\mathrm{col}}_{k\ell_1}$, $\mathfrak{G}^{\mathrm{col}}_{k\ell_2}$, $\mathfrak{G}^{\mathrm{col}}_{k\ell'_1}$, and $\mathfrak{G}^{\mathrm{col}}_{k\ell'_2}$ be horizontally coloured cliquey grids such that $\mathfrak{G}^{\mathrm{col}}_{k\ell_1} \equiv^M_r \mathfrak{G}^{\mathrm{col}}_{k\ell'_1}$ and $\mathfrak{G}^{\mathrm{col}}_{k\ell_2} \equiv^M_r \mathfrak{G}^{\mathrm{col}}_{k\ell'_2}$. Then $\mathfrak{G}^{\mathrm{col}}_{k,\ell_1+\ell_2} \equiv^M_r \mathfrak{G}^{\mathrm{col}}_{k,\ell'_1+\ell'_2}$.

Proof. Note that, since there are no horizontal edges in horizontally coloured cliquey grids and the vertical dimension of all grids is k, $\mathfrak{G}^{\mathrm{col}}_{k,\ell_1+\ell_2}$ is the disjoint union of the two smaller horizontally coloured cliquey grids $\mathfrak{G}^{\mathrm{col}}_{k\ell_1}$ and $\mathfrak{G}^{\mathrm{col}}_{k\ell_2}$, and of course, the same holds for $\mathfrak{G}^{\mathrm{col}}_{k,\ell'_1+\ell'_2}$. Thus, the claim follows by Lemma 3.13. Q.E.D.

Lemma 3.19. Let $\mathfrak{G}^{\mathrm{col}}_{k\ell} \equiv^M_r \mathfrak{G}^{\mathrm{col}}_{k\ell'}$. Then $\mathfrak{G}_{k\ell} \equiv^M_r \mathfrak{G}_{k\ell'}$.

Proof. For each fixed horizontal dimension k, $\mathfrak{G}_{k,\ell}$ is interpretable in $\mathfrak{G}^{\mathrm{col}}_{k,\ell}$ by a one-dimensional quantifier-free interpretation since we can define the horizontal equivalence relation $\sim_h$ in terms of the colours by

$$x \sim_h y \;\equiv\; \bigvee_{i=1}^{k} P_i x \wedge P_i y\,. \qquad \text{Q.E.D.}$$

Actually, thinking in terms of Ehrenfeucht-Fraïssé games, the argument implies that Duplicator wins a game on cliquey grids using the same strategy that is winning in the corresponding game on coloured grids since a strategy preserving the colours of selected elements particularly preserves the equivalence relation $\sim_h$.

Our next goal is to obtain a result about the expressive limits of CMSO on cliquey grids. In particular, we are interested in a result similar to Lemma 3.16 for sets about when two grids with a different number of columns are indistinguishable by a CMSO sentence.

Before approaching this lemma, we will first prove a combinatorial and rather technical result about the existence of equivalence relations satisfying certain constraints, which we need to synthesise Duplicator's winning strategy in the forthcoming game.

Definition 3.20. Two numbers $a, b \in \mathbb{N}$ are called *threshold t equal* (*mod M*), denoted $a =^M_t b$, if

(1) $a = b$ or
(2) $a, b \geq t$ and $a \equiv b \pmod{M}$.

Intuitively, $a =_t^M b$ means that the numbers are equal if they are small, or that they are at least congruent modulo all $m \in M$ if they are both larger than or equal to the threshold t.

Lemma 3.21. For every $p, t \in \mathbb{N}$, and $M \subseteq_{\text{fin}} \mathbb{N}^+$, we can choose an arbitrary threshold $T \geq p \cdot (t + \text{lcm}(M) - 1)$ such that for all sets A and B with $|A| =_T^M |B|$ and for every equivalence relation $\approx_A$ on A of index at most p there exists an equivalence relation $\approx_B$ on B and a bijection $g\colon A/_{\approx_A} \to B/_{\approx_B}$ satisfying $|\{a' \in A : a \approx_A a'\}| =_t^M |g(\{a' \in A : a \approx_A a'\})|$ for all $a \in A$.

Proof. We let $\{a_1, \ldots, a_{p'}\}$, where $p' \leq p$ denotes the index of $\approx_A$, be a system of class representatives of $A/_{\approx_A}$, and we let $[a]_{\approx_A} := \{a' \in A : a' \approx_A a\}$ denote the equivalence class of a in A. Note that we will usually omit the subscript $\approx_A$ if it is clear from the context and instead reserve the letters a and b for elements denoting equivalence classes in A and B, respectively. Furthermore, a set will be called *small* in the following if it contains less than t elements and *large* otherwise.

The equivalence relation $\approx_B$ on B is constructed by partitioning the set into p' disjoint non-empty subsets $\{B_1, \ldots, B_{p'}\}$ as follows. If $|A| = |B|$, we choose a set B_i with exactly $|[a_i]|$ many elements for each class $[a_i]$. If $|A|, |B| \geq T$, we have to distinguish between the treatment of small and large classes. Since $|A| \geq T \geq p \cdot (t + \text{lcm}(M) - 1)$, and $\text{lcm}(M) \geq 1$, and the index of $\approx_A$ is at most p, at least one of the equivalence classes contains at least t elements, i.e. it is large, and without loss of generality, we assume that this is the case for $[a_1]$. For each small class $[a_i]$, we choose a set B_i with exactly $|[a_i]|$ many elements. If $[a_i]$ is large, we choose a set B_i containing $t + \ell$ many elements where ℓ is the smallest non-negative integer such that $|[a_i]| \equiv |B_i| \pmod{M}$. The number of elements selected according to these rules is at most $p \cdot (t + \text{lcm}(M) - 1) \leq T \leq |B|$. Since $[a_1]$ is large by assumption, any possibly remaining elements in B that have not been assigned to one of the subsets $B_1, \ldots, B_{p'}$ yet can be safely added to B_1 without violating the condition that $|[a_1]| \equiv |B_1| \pmod{M}$.

This partitioning uniquely defines the equivalence relation $\approx_B := \bigcup_{i=1}^{p'} (B_i \times B_i)$ on B. By selecting an arbitrary element of each B_i, we get a set of class representatives $\{b_1, \ldots, b_{p'}\}$ which directly yields the bijection

$g\colon [a_i] \mapsto [b_i]$ for all $1 \leq i \leq p'$ satisfying $|[a]| =_t^M |g([a])|$ for all $a \in A$ by construction. Q.E.D.

Descriptions of winning strategies for Duplicator in Ehrenfeucht-Fraïssé games usually involve thresholds depending on the number of rounds that remain to be played that yield notions such as sets *being large enough* or elements *being far enough apart* to render the logic indifferent about the exact size or distance.

The following lemma that extends the result on CMSO-equivalence of *large and similar sets* to *large and similar grids* states that *threshold t equality* (*mod M*) yields an appropriate notion of sets *being similar enough* to be indistinguishable by CMSO formulae. We will prove this by an induction on the number of rounds to be played in an Ehrenfeucht-Fraïssé game on two cliquey grids with the same number of rows, and use the preceding lemma to show that Duplicator has a winning strategy against Spoiler due to certain equivalence relations that she can construct on her grid to appropriately answer Spoiler's set moves. To obtain a condition on the number of rows that are sufficient for Duplicator to do so repeatedly, we consider, for fixed parameters $p \in \mathbb{N}$ (determined by the grid) and $M \subseteq_{\text{fin}} \mathbb{N}^+$ (determined by the allowed moduli), a function $f_{p,M} : \mathbb{N} \to \mathbb{N}$ such that, for all $r \in \mathbb{N}^+$, letting $t = f_{p,M}(r-1)$ we can choose $T = f_{p,M}(r)$ in the previous lemma. A solution to the recursive inequalities obtained by the condition that $T = f_{p,M}(r) \geq p \cdot (f_{p,M}(r-1) + \operatorname{lcm}(M) - 1)$, which was imposed by Lemma 3.21, is $f_{p,M}(r) = 2 \cdot (p^r - 1) \cdot \operatorname{lcm}(M)$.

Lemma 3.22. Let $M \subseteq_{\text{fin}} \mathbb{N}^+$, $r \in \mathbb{N}$ and $k > 1$ be fixed. Then for $f(r) := f_{2^k,M}(r) = (2^{kr+1} - 2)\operatorname{lcm}(M)$, as given above, $\mathfrak{G}_{k\ell_1} \equiv_r^M \mathfrak{G}_{k\ell_2}$ if $\ell_1 =_{f(r)}^M \ell_2$.

Proof. As justified by Lemma 3.19, we consider the r-round (mod M) Ehrenfeucht-Fraïssé game on the corresponding horizontally coloured cliquey grids $\mathfrak{G}^{\text{col}}_{k\ell_1}$ and $\mathfrak{G}^{\text{col}}_{k\ell_2}$, and we show by induction on the number of rounds that Duplicator has a winning strategy in this game.

Intuitively, the proof proceeds as follows. Spoiler's set move induces an equivalence relation determined by the colours of the elements he chooses on the set of columns forming the grid he plays in, and we exploit the previous lemma to show that Duplicator is actually able to construct an equivalence relation on the columns of the other grid that is similar in the sense that corresponding equivalence classes satisfy certain cardinality

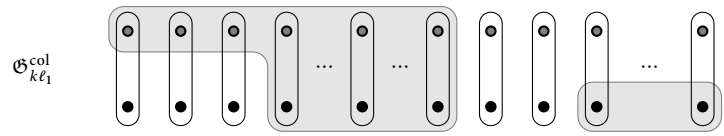

Figure 3.4. Spoiler's set move

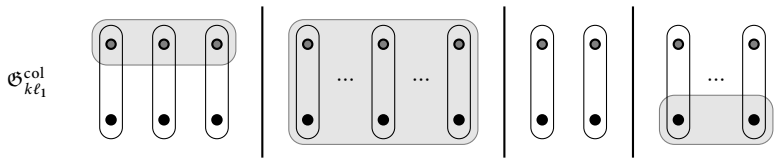

Figure 3.5. Equivalence relation induced by Spoiler's move—the grid falls into $\leq 2^k$ classes

constraints. Since the grids can be regarded as disjoint unions of these equivalence classes, we can argue by induction that corresponding subparts of the two grids, being similar enough, cannot be distinguished during the remaining $r-1$ rounds of the game. A point move can basically be regarded as a singleton-set move.

Assume first that Spoiler performs a set move. The case where $\ell_1 = \ell_2$ is trivial since grids of the same size are isomorphic. Thus, we assume in the following that $\ell_1, \ell_2 \geq f(r)$ and $\ell_1 \equiv \ell_2 \pmod{M}$. The claim is obviously true for $r = 0$, and hence we assume that it holds for $r-1$ and proceed with the inductive step. As before, we assume without loss of generality that Spoiler makes his moves in $\mathfrak{G}^{\mathrm{col}}_{k\ell_1}$ since the other case is symmetric.

A *coloured k-column* $\mathfrak{C}^{\mathrm{col}}_k$ is a $\{\sim_v, P_1, \ldots, P_k\}$-structure that is isomorphic to $\mathfrak{G}^{\mathrm{col}}_{k,1}$, i.e. each coloured grid can be regarded as a disjoint union of coloured columns. Given a subset S of vertices of a grid and one of its coloured k-columns $\mathfrak{C}$ with universe C, the *colour-type of* $\mathfrak{C}$ *induced by* S is defined as the isomorphism type of the expansion $(\mathfrak{C}, S \cap C)$ denoted by $\mathrm{tp}(\mathfrak{C}, S)$. Given a set $\mathcal{F}$ of k-columns, each subset S of all of their vertices gives rise to an equivalence relation $\approx_S$ on $\mathcal{F}$ by virtue of $\mathfrak{C}_1 \approx_S \mathfrak{C}_2$ if and only if $\mathrm{tp}(\mathfrak{C}_1, S) = \mathrm{tp}(\mathfrak{C}_2, S)$. Note that the index of $\approx_S$ is at most 2^k.

Assume, Spoiler performs a set move and chooses a subset S in $\mathfrak{G}^{\mathrm{col}}_{k\ell_1} = \mathfrak{C}_1 \dot\cup \cdots \dot\cup \mathfrak{C}_{\ell_1}$, see Figure 3.4. As described above, S induces an equivalence relation $\approx_S$ with at most 2^k equivalence classes on the set $\mathcal{F} = \{\mathfrak{C}_1, \ldots, \mathfrak{C}_{\ell_1}\}$

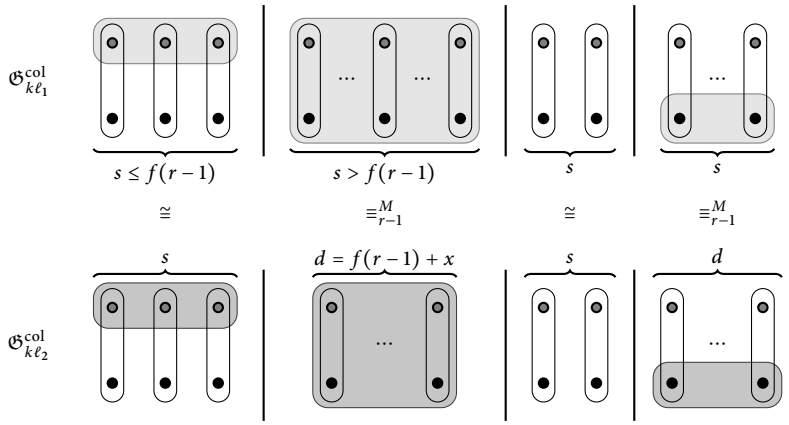

Figure 3.6. Duplicator's response yielding equivalent matching parts

of columns forming the grid, see Figure 3.5. Since $\ell_1, \ell_2 \geq f(r)$, for $p = 2^k$, $t = f(r-1)$ and M as given, the previous lemma ensures the existence of an equivalence relation $\approx'_S$ on the set $\mathcal{F}' = \{\mathfrak{C}'_1, \ldots, \mathfrak{C}'_{\ell_2}\}$ of columns on the Duplicator's grid $\mathfrak{G}^{\mathrm{col}}_{k\ell_2}$ as well as a bijection g mapping equivalence classes of columns in one grid to equivalence classes in the other one such that the cardinalities of the corresponding equivalence classes are threshold t equal (mod M).

Given that the index of both $\approx_S$ and $\approx'_S$ is $p' \leq p = 2^k$, we can assume $\{\mathfrak{C}_1, \ldots, \mathfrak{C}_{p'}\}$ and $\{\mathfrak{C}'_1, \ldots, \mathfrak{C}'_{p'}\}$ to be the sets of class representatives of $\approx_S$ and $\approx'_S$, respectively. Duplicator now selects the unique set D of elements such that $\mathrm{tp}(\mathfrak{C}, S) = \mathrm{tp}(\mathfrak{C}', D)$ for all $1 \leq i \leq p'$, $\mathfrak{C} \in [\mathfrak{C}_i]$ and $\mathfrak{C}' \in g([\mathfrak{C}_i])$, see Figure 3.6.

For each $1 \leq i \leq p'$, we let $\langle \mathfrak{C}_i \rangle := \mathfrak{G}^{\mathrm{col}}_{k\ell_1} \cap [\mathfrak{C}_i]$ and $\langle \mathfrak{C}'_i \rangle := \mathfrak{G}^{\mathrm{col}}_{k\ell_2} \cap [\mathfrak{C}'_i]$ denote the substructures of the grids $\mathfrak{G}^{\mathrm{col}}_{k\ell_1}$ and $\mathfrak{G}^{\mathrm{col}}_{k\ell_2}$ induced by the sets of columns $[\mathfrak{C}_i]$ and $[\mathfrak{C}'_i]$, respectively. By construction, we have $|[\mathfrak{C}_i]| =^M_{f(r-1)} |[\mathfrak{C}'_i]|$ for all i. Thus, depending on whether $[\mathfrak{C}_i]$ (and hence $[\mathfrak{C}'_i]$) are small or large with respect to the threshold $f(r-1)$, either $\langle \mathfrak{C}_i \rangle \cong \langle \mathfrak{C}'_i \rangle$ or $\langle \mathfrak{C}_i \rangle \equiv^M_{r-1} \langle \mathfrak{C}'_i \rangle$ by the induction hypothesis. Since S and D induce the same colour-types on the columns in $[\mathfrak{C}_i]$ and $[\mathfrak{C}'_i]$, respectively, we have

$$\big(\langle \mathfrak{C}_i \rangle, S \cap \langle \mathfrak{C}_i \rangle\big) \equiv^M_{r-1} \big(\langle \mathfrak{C}'_i \rangle, D \cap \langle \mathfrak{C}'_i \rangle\big) \quad \text{for all } i.$$

Thus, a repeated application of Lemma 3.13 shows that Duplicator has a winning strategy in the remaining rounds of the game $\mathcal{G}^M_{r-1}(\mathfrak{G}^{\mathrm{col}}_{k\ell_1}, \mathfrak{G}^{\mathrm{col}}_{k\ell_2})$ from position (S, D).

If Spoiler makes a point move s, say in column $\mathfrak{C}_1$ of the grid $\mathfrak{G}^{\mathrm{col}}_{k\ell_1}$, Duplicator picks an arbitrary element d of the same colour in her grid, say in column $\mathfrak{C}'_1$. As the substructures consisting of just the columns containing the chosen elements are isomorphic, i.e. $\big(\mathfrak{C}_1, s\big) \cong \big(\mathfrak{C}'_1, d\big)$, and by the induction hypothesis we have $\mathfrak{C}_2 \dot\cup \cdots \dot\cup \mathfrak{C}_{\ell_1} \equiv^M_{r-1} \mathfrak{C}'_2 \dot\cup \cdots \dot\cup \mathfrak{C}'_{\ell_2}$, Duplicator has a winning strategy in the remaining $(r-1)$-round game from position (s, d) by Lemma 3.13. Q.E.D.

Proposition 3.23. The class $\mathcal{K}_{\mathrm{div}} := \{\, \mathfrak{G}_{k\ell} : k \mid \ell \,\}$ is not definable in CMSO.

Proof. We show that, for any choice of $r \in \mathbb{N}$ and $M \subseteq_{\mathrm{fin}} \mathbb{N}^+$, there are $k, \ell_1, \ell_2 \in \mathbb{N}$ such that $\mathfrak{G}_{k\ell_1} \in \mathcal{K}$, $\mathfrak{G}_{k\ell_2} \notin \mathcal{K}$, and $\mathfrak{G}_{k\ell_1} \equiv^M_r \mathfrak{G}_{k\ell_2}$ which contradicts the CMSO-definability of $\mathcal{K}$.

Let $r \in \mathbb{N}$ and $M \subseteq_{\mathrm{fin}} \mathbb{N}^+$ be fixed. We choose $s \geq r+1$ such that $2^s \nmid \mathrm{lcm}(M)$. Let $k = 2^s$, $\ell_1 = 2^{kr+1}\,\mathrm{lcm}(M)$, and $\ell_2 = \ell_1 + \mathrm{lcm}(M)$. Obviously, ℓ_1 and ℓ_2 satisfy the conditions of Lemma 3.22, and thus $\mathfrak{G}_{k\ell_1} \equiv^M_r \mathfrak{G}_{k\ell_2}$.

Furthermore, $\ell_1 = k \cdot 2^{2^s \cdot r - s + 1}\,\mathrm{lcm}(M)$, and hence $k \mid \ell_1$ and $\mathfrak{G}_{k\ell_1} \in \mathcal{K}$. But $k \nmid \ell_2 = \ell_1 + \mathrm{lcm}(M)$ by the choice of s, and hence $\mathfrak{G}_{k\ell_2} \notin \mathcal{K}$. Q.E.D.

Theorem 3.24. $\mathrm{CMSO} \subsetneq \mathrm{MSO}_{<\text{-inv}}$.

Proof. We show that the class $\mathcal{K}_{\mathrm{div}}$ is order-invariantly definable in MSO by the sentence $\psi_{\mathrm{grid}} \wedge \varphi$, where

$$\varphi = \exists min \exists c \left(\begin{array}{l} \forall x (min \leq x) \wedge \neg \exists z (E_h(c,z) \vee E_v(c,z)) \\ \wedge\ \forall T \Big(Tmin \wedge \forall x \forall y (Tx \wedge \varphi_{\mathrm{diag}}(x,y) \rightarrow Ty) \\ \qquad\qquad \rightarrow Tc \Big) \end{array} \right),$$

and

$$\varphi_{\mathrm{diag}}(x,y) = \big(\exists z (E_v(x,z) \wedge E_h(z,y))\big) \\ \vee \big(\neg \exists z E_v(x,z) \wedge \exists z (z \sim_h min \wedge z \sim_v x \wedge E_h(z,y))\big),$$

$$E_h(x,y) = x \sim_h y \wedge \exists x_0 \exists y_0 \left(\begin{array}{l} x_0 \sim_h min \wedge y_0 \sim_h min \\ \wedge\ x \sim_v x_0 \wedge y \sim_v y_0 \wedge x_0 < y_0 \\ \wedge\ \forall z_0 (z_0 \sim_h min \to z_0 \leq x_0 \vee z_0 \geq y_0) \end{array} \right),$$

$$E_v(x,y) = x \sim_v y \wedge \exists x_0 \exists y_0 \left(\begin{array}{l} x_0 \sim_v min \wedge y_0 \sim_v min \\ \wedge\ x \sim_h x_0 \wedge y \sim_h y_0 \wedge x_0 < y_0 \\ \wedge\ \forall z_0 (z_0 \sim_v min \to z_0 \leq x_0 \vee z_0 \geq y_0) \end{array} \right),$$

As hinted above, the horizontal and vertical edge relations (E_h and E_v, respectively) are defined using the successor relation which is induced by an arbitrary ordering on the row (and column) containing the minimal element (*min*) which itself serves as the lower left corner of the grid. φ_{diag} defines diagonal steps through the grid that wrap around from the top to the bottom row. Finally, φ states that the pair consisting of the lower left corner (*min*) and the upper right corner (c) of the grid is contained in the transitive closure of φ_{diag}. Obviously, there is such a sawtooth-shaped path starting at *min* and ending exactly in the upper right corner if and only if $k \mid \ell$.

By the previous proposition, $\mathcal{K}_{\text{div}}$ is not CMSO-definable, and hence the claim follows. Q.E.D.

Using a straightforward interpretation argument, we obtain Courcelle's original conjecture as a corollary.

Corollary 3.25. CMSO is strictly less expressive than $\text{MSO}_{<\text{-inv}}$ on plain graphs.

Proof. By taking the union of the two equivalence relations and removing tuples of the form (v,v), we obtain an undirected graph from a cliquey grid formalised by the following interpretation $\mathfrak{I}$:

$$\begin{aligned} \delta(x) &= (x = x) \\ \varepsilon(x,y) &= (x = y) \\ \varphi_E(x,y) &= (x \neq y) \wedge (x \sim_h y \vee x \sim_v y) \ . \end{aligned}$$

If there were a formula $\varphi \in \text{CMSO}[E]$ defining the class of k by ℓ grids such that $k \mid \ell$, then, since $\mathfrak{I}$ is a one-dimensional interpretation,

$\mathfrak{I}(\varphi) \in \mathrm{CMSO}[\sim_h, \sim_v]$ would define the class $\mathcal{K}_{\mathrm{div}}$ which contradicts Proposition 3.23. Q.E.D.

It is well known that unary TC formulae, i.e. formulae applying the transitive closure operator to binary relations only, can be translated into MSO via

$$[\mathrm{tc}_{xy}\, \varphi(x,y,z)](x,y) \mapsto \forall X(\forall u \forall v(Xu \wedge \varphi(u,v,z) \to Xv) \wedge Xx \to Xy) \ .$$

The same holds for DTC formulae since they can be translated into TC formulae of the same arity.

Corollary 3.26.

(1) $\mathrm{DTC}^1 \subsetneq \mathrm{DTC}^1_{<\text{-inv}}$
(2) $\mathrm{TC}^1 \subsetneq \mathrm{TC}^1_{<\text{-inv}}$

Proof. We argue that the order-invariant MSO sentence defining the class $\mathcal{K}_{\mathrm{div}} := \{ \mathfrak{G}_{k\ell} : k \mid \ell \}$ can be expressed in DTC. The horizontal and vertical edge relations and the diagonal edges are FO definable, and the remaining part defines the set of vertices reachable by a path of diagonal edges. Since the diagonal edge relation is deterministic, its transitive closure is DTC definable, and hence

$$\varphi = \exists min \exists c \begin{pmatrix} \forall x(min \leq x) \wedge \neg \exists z(E_h(c,z) \vee E_v(c,z)) \\ \wedge\, [\mathrm{dtc}_{xy}\, \varphi_{\mathrm{diag}}(x,y)](min, c) \end{pmatrix}$$

defines the class $\mathcal{K}_{\mathrm{div}}$. Q.E.D.

3.5 Conclusion

We have provided a characterisation of the expressiveness of CMSO in terms of an Ehrenfeucht-Fraïssé game that naturally extends the known game capturing MSO-definability, and we have presented an order-invariantly MSO definable class of structures that are shown, using the proposed game characterisation, not to be definable by a CMSO-sentence. This establishes that order-invariant MSO is strictly more expressive than counting MSO

in the finite and proves Courcelle's conjecture about the expressiveness of order-invariant MSO.

There are further useful classes that are order-invariantly definable such as the class of square grids (by not letting the diagonal edges wrap around from the top to the bottom row of the grid) or the class of square grids whose width (and thus height) is prime (by stating that there is no non-trivial subset of the rows such that restricting the diagonal edges to these rows still leaves the border reachable).

Further, Otto's separating example for order-invariant first-order logic (for details see [Otto00]) is based on the idea that a property of a graph, which is embedded in a larger structure, becomes definable if one has access to a Boolean algebra over the vertices of the graph. To prevent plain FO from making use of that Boolean algebra, he extends the structure by inserting an intermediate part, which we can regard as an *information hiding gadget* as it is connected to the graph and the Boolean algebra in such a way that a bijection between the graph and the atoms of the Boolean algebra is order-invariantly definable but not in plain FO. Since it is possible to define a bijection between a row and a column in a square cliquey grid order-invariantly, such a grid could serve as an information hiding gadget inside other separating examples.

4 LOCAL ORDERINGS

The descriptive complexity of transitive closure logics on ordered structures is well understood since Immerman showed that DTC and TC capture Logspace and NLogspace, respectively [Imm87, Imm88]. However, on unordered structures, computationally simple properties such as the Logspace-computable property EVEN are not DTC definable in general. In fact, there are classes of graphs on which DTC collapses to FO as shown by Grädel and McColm [GM95].

In this chapter, we advance results of Etessami and Immerman who addressed the question of whether weaker forms of orderings, in particular local orderings on graphs, are already sufficient to obtain similar capturing results. An essential insight originates from the reachability algorithm recently presented by Reingold in his proof that undirected reachability is decidable in Logspace.

We begin with formally introducing the model of locally ordered graphs and discuss different representations. Afterwards, we introduce basic notions on expander graphs, which are an integral part of Reingold's algorithm, to provide the necessary intuitions. Then we show how Reingold's algorithm deciding the undirected reachability problem gives rise to a DTC formula defining the transitive closure of undirected locally ordered graphs with an additional linearly ordered number sort. Combining this result with the techniques used by Etessami and Immerman in their proof that TC+C, the extension of TC by counting terms, captures NLogspace on two-way locally ordered graphs, we finally obtain the main theorem that DTC+C captures Logspace on undirected locally ordered graphs.

4.1 Locally Ordered Graphs

A locally ordered graph is a graph in which each vertex is equipped with an ordering that linearly orders its 1-neighbourhood. In case of directed graphs, we distinguish orderings on the set of successors of a vertex, i.e. the set of vertices incident to the outgoing edges, from orderings on the set of predecessors of a vertex, i.e. the set of vertices incident to the incoming edges. Since each successor or predecessor of a vertex uniquely defines an edge in the graph, a local ordering simultaneously defines an order on incoming and outgoing edges. It is also useful to imagine that the edges are labelled by numbers according to their position in the ordering such that we will equivalently speak of *the i-th incoming or outgoing edge* of a vertex or *the edge with label i*.

4.1.1 Representing locally ordered graphs

There are two different approaches present in the literature towards representing locally ordered graphs as relational structures. First, the model considered by Etessami and Immerman [EI95a, EI00] represents a locally ordered graph as an $\{E, I, O\}$-structure (or $\{E, \lessdot, \lesssim\}$-structure) where the local orderings are represented as ternary relations O or $\lesssim$ (for an order on the outgoing edges or successor vertices) and I or $\lessdot$ (for an order on the incoming edges or predecessor vertices) such that, for every vertex v, $O(v, \cdot, \cdot)$ or $\lesssim_v$ is a linear order on the set of successors of v, and analogously, $I(v, \cdot, \cdot)$ or $\lessdot_v$ is a linear order on the set of predecessors of v. If there is only one local ordering present (usually the one on the outgoing edges/successors), we speak of a *one-way* locally ordered graph, otherwise we call the graph *two-way* locally ordered.

A second possibility is to describe a graph in terms of its rotation map introduced by Reingold et al. [RVW02]. A *rotation map* defining an undirected d-regular graph, possibly with self-loops and parallel edges, in which the edges incident to a vertex are labelled by $0, \dots, d-1$ is a permutation $\rho : V \times [d]_0 \to V \times [d]_0$ on pairs of a vertex and an edge label with the semantics that $\rho(v, \alpha) = (w, \beta)$ denotes that the α-th edge leaving v is the β-th edge entering w. Due to the explicit use of edge labels, this presentation can only be realised in a structure with a sufficient number of constants or in a two-sorted structure (e.g. in the canonical two-sorted graph structure) using

two functions or a single (2,2)-ary edge relation $E \subseteq V \times [n]_0 \times V \times [n]_0$, interpreted as the graph of the rotation map, such that $(v, \alpha, w, \beta) \in E$ if and only if $\rho(v, \alpha) = (w, \beta)$.

The rotation map or its relational presentation is particularly useful in an algorithmic context since it combines the complete description of a graph in a single data structure and offers direct access to particular neighbours of a vertex. Moreover, it is straightforward to represent graphs with self-loops and parallel edges between vertices as these are distinguishable by their associated labels.

From a logical point of view, the first model is more universal in the sense that the relations are not defined in terms of additional constants or an additional number sort which is usually only assumed to be present in the context of counting logics. Furthermore, the original definition of a rotation map is only suitable for the special case of representing undirected regular multigraphs. With respect to logical definability, however, the two approaches, at least in cases where both are applicable, are equivalent in the sense that the relations are inter-definable in any logic in which a bijection between numbers and positions in the local ordering is definable.

Since the graph construction we describe in the following introduces parallel edges, we will work with graphs defined by a quaternary edge relation which can be considered as representing a partially defined rotation map where a vertex in a non-regular graph does not necessarily need to have an i-th neighbour for each i, and which naturally supports the representation of multiple edges between pairs of vertices. In particular, using the following interpretation $\mathfrak{I}_{\mathrm{rot}}$, we obtain $\mathfrak{I}_{\mathrm{rot}}(\mathfrak{G}^*)$ as the canonical two-sorted graph represented by its rotation map interpreted in a two-sorted graph $\mathfrak{G}^* = (V, E, \leq, \leq) \cup (\{0, \dots, |V|\}, <)$ where the local orderings are given by ternary relations:

$$\begin{aligned}
\delta_1(v) &= (v = v) \\
\delta_2(\iota) &= (\iota = \iota) \\
\varepsilon_1(v, w) &= (v = w) \\
\varepsilon_2(\iota, \kappa) &= (\iota = \kappa) \\
\varphi_R((v, \iota), (w, \kappa)) &= \\
&\exists z\big(\zeta(v, z) \wedge [\mathrm{dtc}_{x\alpha, y\beta}\, \mathrm{succ}_{\mathrm{out}}(v, x, y) \wedge \beta = \alpha + 1](z, 0, w, \iota)\big) \\
&\wedge \exists z\big(\zeta(w, z) \wedge [\mathrm{dtc}_{x\alpha, y\beta}\, \mathrm{succ}_{\mathrm{in}}(w, x, y) \wedge \beta = \alpha + 1](z, 0, v, \kappa)\big)
\end{aligned}$$

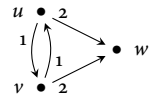

Figure 4.1. A non-rigid one-way locally ordered graph

where

$$\zeta(v,z) = E(v,z) \wedge \forall w(E(v,w) \to (z \leq_v w \vee w = z))$$
$$[z \text{ is the smallest neighbour of } v]$$
$$\mathrm{succ}_{\mathrm{out}}(v,x,y) = x \lesssim_v y \wedge \forall u(x \lesssim_v u \wedge u \lesssim_v y \to (x = u \vee y = u))$$
$$[x \text{ and } y \text{ are successors w. r. t. } \lesssim_v]$$
$$\mathrm{succ}_{\mathrm{in}}(v,x,y) = x \lesssim_v y \wedge \forall u(x \lesssim_v u \wedge u \lesssim_v y \to (x = u \vee y = u))$$
$$[x \text{ and } y \text{ are successors w. r. t. } \lesssim_v] \ .$$

4.1.2 One-Way vs. two-way local orderings

In the case of directed graphs, there is a difference between one-way locally ordered graphs and two-way locally ordered ones. Consider, for example, the one-way locally ordered graph $\mathfrak{G}$ in Figure 4.1. The mapping

$$\pi : u \mapsto v,\ v \mapsto u,\ w \mapsto w$$

is an automorphism of $\mathfrak{G}$, and hence the two predecessors of w, u and v, cannot be distinguished by only a local order on the outgoing edges. Intuitively, if we walk along one of the edges to w, upon entering w we cannot possibly know where we came from. However, in presence of a second local ordering on the predecessors of vertices, the graph becomes rigid since either u or v is the smallest predecessor of w, and the vertices become distinguishable. This potentially renders logics more powerful on two-way than on one-way locally ordered graphs. Let us remark that, contrary to totally ordered graphs, two-way locally ordered graphs are not necessarily rigid.

Since the edge relation in undirected graphs is symmetric, each predecessor of a vertex is also one of its successors. Hence, the distinction between

orderings on predecessors and successors becomes obsolete, and the notions of one-way and two-way locally ordered graphs coincide.

4.2 Reingold's Algorithm

In 2005, Reingold proved that the undirected s-t-reachability problem (in short, USTCON), i.e. the problem whether two distinguished vertices s and t are connected by some path in an undirected graph, can be reduced to the reachability problem in expander graphs. In particular, he presented a method to transform arbitrary undirected graphs into regular expander graphs with a predefined degree and expansion parameter while preserving reachability, and he showed that the transformation is in fact LOGSPACE-computable. Since the diameter of an expander graph is logarithmic in the number of its vertices, deciding the existence of a connecting path can be realised, for regular graphs, by a logarithmically space bounded computation that performs an exhaustive search of all paths of length bounded by the diameter.

4.2.1 Expander Graphs

We introduce some notions and properties of expander graphs needed to get a basic understanding of how and why the algorithm works. For a more comprehensive overview, we refer to the detailed survey on applications of expander graphs [HLW06].

Intuitively, an undirected (multi)graph is called an *expander graph* if it is sparse, i.e. the degree of its vertices is small compared to its size, but nevertheless highly connected, i.e. neighbourhoods around vertices or sets of vertices are large. To make this precise, there are several notions of *expansion* and *expansion parameters* relating to different but tightly connected essential properties of expander graphs.

First, the notion of *edge expansion* takes the number of edges from a subset of the vertices to its neighbourhood into account.

Definition 4.1. Let $\mathfrak{G} = (V, E)$ be an undirected multigraph (self-loops are allowed) on n vertices, and let ∂S denote the *edge boundary* of a set $S \subseteq V$, i.e. the set of edges from S into its complement. Then the *edge expansion*

ratio of $\mathfrak{G}$ is defined as

$$h(\mathfrak{G}) := \min_{S \subseteq V, |S| \leq \frac{|V|}{2}} \frac{|\partial S|}{|S|} \ .$$

The second notion stems from spectral graph theory and defines the *spectral expansion ratio* of a *regular* graph in terms of the eigenvalues of its normalised adjacency matrix.

Definition 4.2. Let $\mathfrak{G} = (V, E)$ be an undirected d-regular multigraph (self-loops are allowed) on n vertices, and let $A = A(\mathfrak{G})$ be its adjacency matrix, i.e. the entry $A(u, v)$ denotes the number of edges between the vertices u and v. Let $\hat{A} = \frac{1}{d} A$ be the normalised adjacency matrix and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ be its eigenvalues. Since $\mathfrak{G}$ is undirected, the matrices A and $\hat{A}$ are symmetric and have n real eigenvalues. Furthermore, as $\hat{A}$ is normalised, $\lambda_1 = 1$. The *spectral expansion ratio* of $\mathfrak{G} = (V, E)$ is defined as the second-largest eigenvalue $\lambda(\mathfrak{G}) := \lambda_2$.

The analysis of randomised reachability algorithms yields an intuitive interpretation of the spectral expansion ratio and the associated *spectral gap* $1 - \lambda(\mathfrak{G})$, i.e. the gap between the largest and the second-largest eigenvalue. In fact, the spectral expansion ratio gives a good estimate on the *mixing time*, i.e. the expected number of steps after which a random walk on the graph starting from an arbitrary initial distribution converges to a uniform distribution on the set of reachable vertices (note that the normalised adjacency matrix is the probabilistic transition matrix of the graph).

We consider a graph to be a good expander if it has a large edge expansion ratio, or, in terms of the spectrum, if it has a large spectral gap (which means a *small* second-largest eigenvalue).

By definition, every connected graph has non-zero but possibly very small edge-expansion; in fact, $h(\mathfrak{G}) = \Omega(1/|V|)$, i.e. this trivial bound tends to zero as the size of the graph grows larger. On the other hand, graphs with a large minimal degree naturally have large edge-expansion. Hence, there are two important questions. Are there infinite families of larger and larger graphs whose degree grows very slowly or is bounded by a constant but whose expansion parameter is nevertheless bounded away from zero by a positive constant? And, given a particular size, degree and edge-expansion ratio, is there a graph with these parameters, and, if so, can we construct it?

Indeed, an explicit construction of a family of 8-regular expander graphs was given by Margulis [Mar73], and later many more constructions were presented. Towards the second question, Pinsker gave a probabilistic proof that almost all graphs are good expanders [Pin73]. However, the problem to decide whether a graph has a given edge expansion ratio is coNP-hard [BKV+81], so the naïve approach of randomly constructing a graph and verifying its expansion properties is not feasible for actually obtaining the desired expander graph.

The following theorem obtained by Alon and Milman [AM85, Alo86] stating that the two expansion ratios are related in the sense that a small edge expansion ratio implies a small spectral gap and vice versa allows for such an approach since the spectral gap of the adjacency matrix is easy to compute and provides a lower bound on the edge expansion ratio.

Theorem 4.3 ([AM85, Alo86])**.** Let $\mathfrak{G}$ be an undirected d-regular multigraph, and let $\lambda_1 \geq \cdots \geq \lambda_n$ be the spectrum of its normalised adjacency matrix. Then

$$\frac{1}{2}(1-\lambda_2) \leq \frac{h(\mathfrak{G})}{d} \leq \sqrt{2(1-\lambda_2)} \ .$$

Besides the possibility to obtain expanders by randomly guessing and checking the spectral expansion ratio, there are several other explicit construction methods available. This justifies the following proposition ensuring the existence of an expander graph with particular parameters which is necessary for the construction performed by Reingold's algorithm.

Proposition 4.4 ([Rei05])**.** There exists a number $d \in \mathbb{N}$ such that there exists a d-regular graph $\mathfrak{G} = (V, E)$ with d^{16} vertices and spectral expansion ratio $\lambda(\mathfrak{G}) \leq 1/2$.

We conclude this section by stating the following crucial property of expander graphs the algorithm relies upon.

Proposition 4.5. Let $\mathfrak{G} = (V, E)$ be an undirected d-regular connected multigraph with edge expansion ratio $h(\mathfrak{G}) = h > 0$ and $|V| = n$. Then the diameter of $\mathfrak{G}$, $\Delta(\mathfrak{G})$, is bounded by $c \log n$ for some constant $c > 0$.

Proof. Let $\delta(u, v)$ denote the distance between vertices u and v in $\mathfrak{G}$, let $B_r(u) := \{v \in V : \delta(u, v) \leq r\}$ be the r-ball around u consisting of all

vertices whose distance to u is at most r, and let $B_r(U) := \bigcup_{u \in U} B_r(u)$. Due to the expansion property of $\mathfrak{G}$, for every U such that $|B_r(U)| \leq \frac{|V|}{2}$,

$$|B_{r+1}(U)| \geq \underbrace{\left(1 + \frac{h}{d}\right)}_{>1} |B_r(U)| \quad .$$

Hence, for any vertex u, $|B_r(\{u\})| \geq \frac{n}{2}$ for some $r \leq \log_{(1+\frac{h}{d})} \frac{n}{2}$, which implies that the length of a shortest path between any two vertices is bounded by $2\lceil \log_{(1+\frac{h}{d})} \frac{n}{2} \rceil + 1$, and hence, $\Delta(\mathfrak{G}) \in \mathcal{O}(\log n)$. Q.E.D.

4.2.2 Graph transformations

The goal of the algorithm is to transform an arbitrary input graph into a regular expander graph with short paths between any two vertices, i.e. to reduce the diameter while keeping the degree constant. The transformation is mainly based on two operations: Powering and the zig-zag product. *Powering* a graph, i.e. raising its adjacency matrix to some power, shortens the paths and improves the expansion parameter but increases the degree polynomially. The *zig-zag product* taken with a good expander graph, on the other hand, yields a graph with slightly worse expansion properties but greatly reduced degree. Interestingly, although the operations achieve rather opposite goals and seem to mutually foil each other's improvements, by choosing the parameters carefully, it can be ensured that successive transformations yield a good expander graph with a fixed degree in the end.

Following the original exposition, and due to the introduction of parallel edges by powering a graph, we describe the operations in terms of the representation via rotation maps.

Definition 4.6 (Powering)**.** The *p-power of* $\mathfrak{G}$ is obtained by raising the adjacency matrix $A = A(\mathfrak{G})$ of $\mathfrak{G}$ to the p-th power. This yields a matrix in which each entry $A^p(u, v)$ denotes the number of paths of length exactly p between u and v in $\mathfrak{G}$. Obviously, given a d-regular graph $\mathfrak{G}$, its p-power $\mathfrak{G}^p$ is d^p-regular, and generally contains parallel edges since the entries in A^p can be larger than 1. If the graph $\mathfrak{G}$ is locally ordered, the induced edge labelling itself naturally induces a labelling of the edges of $\mathfrak{G}^p$ by sequences of length p (see Figure 4.2). Hence, the rotation map of the p-th power of a locally ordered graph $\mathfrak{G}$ is a permutation of pairs of a vertex with a sequence

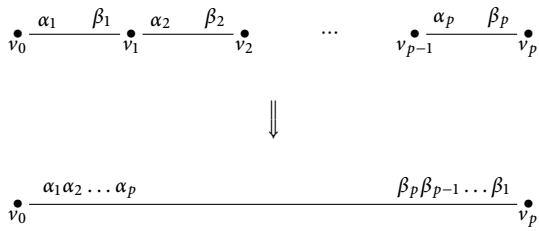

Figure 4.2. p-Powering: A path of p edges is transformed into a single edge, and the respective edge labels are concatenated.

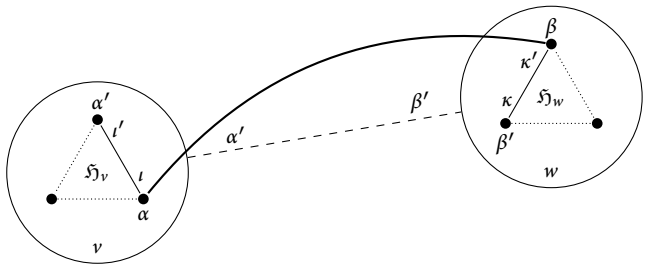

Figure 4.3. Construction of the edge $\big((v,\alpha),(\iota,\kappa)\,;\,(w,\beta),(\iota',\kappa')\big)$ in the zig-zag product

of p edge labels and is FO definable in terms of the edge relation of $\mathfrak{G}$ as follows:

$$E^p(v_0,\alpha_1\ldots\alpha_p,v_p,\beta_p\ldots\beta_1) = \exists v_1\ldots v_{p-1}\bigwedge_{i=1}^{p} E(v_{i-1},\alpha_i,v_i,\beta_i)\quad .$$

Definition 4.7. The *zig-zag product* $\mathfrak{G}\circledcirc\mathfrak{H}$ of a D-regular graph $\mathfrak{G} = (V, E^{\mathfrak{G}})$ on n vertices with a d-regular graph on D vertices $\mathfrak{H} = ([D]_0, E^{\mathfrak{H}})$ is defined as follows:

- $V^{\mathfrak{G}\circledcirc\mathfrak{H}} := V \times [D]_0$
- $E^{\mathfrak{G}\circledcirc\mathfrak{H}}\big((v,\alpha),(\iota,\kappa)\,;\,(w,\beta),(\iota',\kappa')\big)$ iff there exist $\alpha',\beta' \in [D]_0$ s. th.
  (1) $\big((\alpha,\iota);(\alpha',\iota')\big) \in E^{\mathfrak{H}}$
  (2) $\big((v,\alpha');(w,\beta')\big) \in E^{\mathfrak{G}}$
  (3) $\big((\beta',\kappa);(\beta,\kappa')\big) \in E^{\mathfrak{H}}$.

It is crucial that the parameters are chosen such that there is a one-to-one correspondence between the vertices of the graph $\mathfrak{H}$ and the edge labels in $\mathfrak{G}$. Intuitively, the zig-zag product of $\mathfrak{G}$ and $\mathfrak{H}$ replaces each vertex of $\mathfrak{G}$ by a copy of $\mathfrak{H}$, and we will refer to the set of vertices replacing a vertex v in $\mathfrak{G}$ as *the cloud of* v. These new vertices are connected as depicted in Figure 4.3 by following three edges. The first edge label (ι) determines the so-called *zig*-step in the local copy of $\mathfrak{H}$ the target vertex of which corresponds to an edge label (α') in $\mathfrak{G}$. The second step is taken in $\mathfrak{G}$ leading to another copy of $\mathfrak{H}$ ($\mathfrak{H}_w$). The edge label seen upon entering this copy (β') determines the vertex in the copy of $\mathfrak{H}$ from which to proceed with the third and final *zag*-step following the second given edge label (κ) in that target copy of $\mathfrak{H}$. Performing this construction for all possible combinations of two edge labels yields a d^2-regular graph, and the intention is to operate with a fixed d-regular graph $\mathfrak{H}$ on D-regular graphs where D is much larger than d such that in the resulting zig-zag product graph the number of vertices increases (by the constant factor D) while the degree is considerably reduced to d^2.

The effects of these transformations on the expansion parameter of the constructed graph are crucial for the analysis of the space requirements of the algorithm. By definition of the powering operation, the adjacency matrix $A(\mathfrak{G}^p) = A(\mathfrak{G})^p$. Hence, the eigenvalues of $A(\mathfrak{G}^p)$ are $\lambda_1^p, \ldots, \lambda_n^p$ where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of $A(\mathfrak{G})$. This proves the following proposition implying that powering, as intended, widens the spectral gap of a graph since the value of the second-largest eigenvalue decreases.

Proposition 4.8. Let $\mathfrak{G}$ be an undirected d-regular connected multigraph. Then the spectral expansion ratio $\lambda(\mathfrak{G}^p) = \lambda(\mathfrak{G})^p$.

The following theorem gives an estimate on the spectral expansion of the zig-zag product of two graphs.

Theorem 4.9 ([Rei05])**.** Let α and λ be fixed, let $\mathfrak{G}$ be a D-regular graph such that $\lambda(\mathfrak{G}) \leq \lambda$, and let $\mathfrak{H}$ be a d-regular graph with D vertices such that $\lambda(\mathfrak{H}) \leq \alpha$. Then

$$\lambda(\mathfrak{G} \circledcirc \mathfrak{H}) \leq \frac{1}{2}(1-\alpha^2)\lambda + \frac{1}{2}\sqrt{(1-\alpha^2)^2\lambda^2 + 4\alpha^2} \quad .$$

This theorem implies that the spectral gap of the product graph does not decrease too much, and indeed the following corollary shows that it can be bounded from below by a factor that solely depends on the spectral expansion α of $\mathfrak{H}$.

Corollary 4.10 ([Rei05]). Let α and λ be fixed, let $\mathfrak{G}$ be a D-regular graph such that $\lambda(\mathfrak{G}) \leq \lambda$, and let $\mathfrak{H}$ be a d-regular graph with D vertices such that $\lambda(\mathfrak{H}) \leq \alpha$. Then

$$1 - \lambda(\mathfrak{G} \circledcirc \mathfrak{H}) \geq \frac{1}{2}(1 - \alpha^2) \cdot (1 - \lambda) \quad .$$

4.2.3 Reingold's Expander Construction

We first give an informal high-level description of the algorithm as presented in [Rei05] and sketch the proof of its correctness and the space bound. Later, we show how the configurations arising during the computation can be encoded in a two-sorted graph structure, and explain the steps and their description in DTC in more detail.

Reingold states that the expander graph needed for the construction can be obtained by a Logspace computation. However, since it does not depend on the input graph, it may as well be obtained independently and hard-coded in the algorithm. Hence, for every $d \geq 3$ such that there exists a d-regular graph $\mathfrak{X}$ with d^{16} vertices and $\lambda(\mathfrak{X}) \leq 1/2$, we obtain an algorithm deciding the reachability problem in logarithmic space.

By Proposition 4.4 such a constant d exists. Let us fix this d and the corresponding expander graph $\mathfrak{X}$ for the rest of the section. The algorithm deciding whether two vertices s and t are connected in $\mathfrak{G}$ consists of the following three steps.

(1) Transform the input graph $\mathfrak{G}$ into a $D = d^{16}$-regular graph $\mathfrak{G}_{\text{reg}}$ while preserving its connectivity.
(2) Improve the expansion properties of $\mathfrak{G}_{\text{reg}}$ while keeping its degree constant by iterating the zig-zag product with the expander $\mathfrak{X}$ and 8-powering $\ell = \mathcal{O}(\log n)$ many times. This finally yields a D-regular graph $\mathfrak{G}_{\text{exp}}$ with sufficiently good expansion properties.
(3) Check for all paths of length at most $k \cdot \log n$, where k depends only on the expansion parameter of $\mathfrak{X}$, whether one of them connects the clouds of s and t in $\mathfrak{G}_{\text{exp}}$.

Lemma 4.11 ([Reio5]). Let $\mathfrak{G} = (V, E)$ and $\mathfrak{X}$ be the graphs used in the previous transformation. If $\mathfrak{G}$ is connected and non-bipartite, $\lambda(\mathfrak{X}) \leq 1/2$, and ℓ is the smallest integer such that $(1 - \frac{1}{D|V|^2})^{2^\ell} < \frac{1}{2}$, then $\lambda(\mathfrak{G}_{\text{exp}}) \leq \frac{1}{2}$.

Concerning the logarithmic space bound it is important to note that the expander graph resulting from the transformation cannot be constructed explicitly since in each iteration of Step (2) the size of the graph increases by a constant factor, and hence, the size of the final expander graph is polynomial in the size of the input and thus too large to be stored in memory. Instead, the algorithm, in its outer loop, performs Step (3) by repeatedly computing values of the rotation map of the final expander in logarithmic space on the fly (see Procedure 4.1).

4.3 Defining Reachability in DTC

We describe how the computation performed by Reingold's algorithm can be captured by formulae in DTC on locally ordered graphs with an additional number sort where the edge relation and the local ordering are represented by a $(2, 2)$-ary relation as discussed in Section 4.1.

We aim at a formula $\varphi(u, v) \in \text{DTC}$ defining the transitive closure of the edge relation of an undirected locally ordered graph, and for technical reasons, without loss of generality, we restrict our attention to graphs with at least D vertices (the value of D will become clear during the construction) for the remainder of this section; the transitive closure of the edge relation of a graph with less than D vertices is definable by a plain FO formula.

Note that we will use numeric constants such as $0, 1, 2, \mathit{max}$ (denoting the maximal element of the number sort) as well as arithmetic operations such as addition, subtraction, multiplication and exponentiation, which are all definable in DTC on a linearly ordered number sort. (See Section 2.2.3 for further details.)

We begin with following the steps of the reachability algorithm which will eventually yield a DTC-interpretation of the expander graph that is implicitly constructed by the algorithm in a given two-sorted locally ordered graph $\mathfrak{G}^*$.

Step 1. Transformation of the input graph $\mathfrak{G}$ into a D-regular graph $\mathfrak{G}_{\text{reg}}$ where the value of D follows from constraints motivated in Step 2.

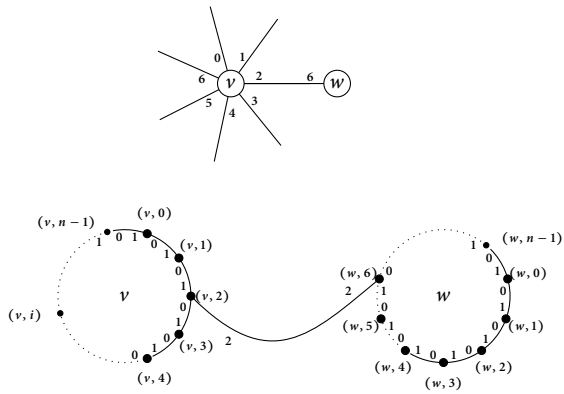

Figure 4.4. Transformation of a graph $\mathfrak{G}$ (upper part) into the regular graph $\mathfrak{G}^{\text{reg}}$ (lower part); exemplified by an edge connecting two vertices v and w

This is accomplished by replacing each vertex v in the graph $\mathfrak{G} = (V, E)$ by a gadget consisting of a cycle of $n = |V|$ vertices $(v, 0), \ldots, (v, n-1)$ such that (v, ι) and (v, κ) are connected via their first and second incident edge if $\iota - \kappa = \pm 1 \pmod{n}$ (see Figure 4.4). In Reingold's construction, the edge relation of $\mathfrak{G}$ is preserved by introducing an edge $((v, w), 2, (w, v), 2)$ in $\mathfrak{G}_{\text{reg}}$ for every edge $(v, \cdot, w, \cdot)$ of $\mathfrak{G}$. Since $\mathfrak{G}$ is not ordered, there is no correspondence between vertices and the numbers used to index the vertices on the cycle, so we need to deviate from the original construction. We do so by translating an edge (v, ι, w, κ) of the input graph into an edge $((v, \iota), 2, (w, \kappa), 2)$ in $\mathfrak{G}_{\text{reg}}$, and thus even retain the information about the labels of the original edge. Eventually, to obtain a D-regular graph, we add an appropriate number of self-loops to each vertex. Note that each self-loop only contributes 1 to the valence of the vertex.

This yields the following interpretation $\mathfrak{I}_{\text{reg}}$ interpreting the two-sorted regular graph $\mathfrak{G}_{\text{reg}}$ in $\mathfrak{G}^*$ by representing a vertex in $\mathfrak{G}_{\text{reg}}$ by a pair consisting of a vertex and a number in $\mathfrak{G}^*$.

$$\begin{aligned}
\delta_1(v, \iota) &= \exists \mu(\iota < \mu) && \big[\text{restrict the second component} \\
\delta_2(\iota) &= (\iota = \iota) && \text{to values from } \{0, \ldots, |V| - 1\}\big] \\
\varepsilon_1((v, \iota), (w, \kappa)) &= (v = w) \wedge (\iota = \kappa) \\
\varepsilon_2(\iota, \kappa) &= (\iota = \kappa)
\end{aligned}$$

$$\varphi_<(\iota,\kappa) = \iota < \kappa$$
$$\begin{aligned}\varphi_E\big((\nu,\iota),\alpha,(w,\kappa),\beta\big) = \\ &(\alpha = 0 \wedge \beta = 1 \wedge \nu = w \wedge \kappa = \iota + 1 \pmod{max}) \\ \vee\ &(\alpha = 1 \wedge \beta = 0 \wedge \nu = w \wedge \kappa = \iota - 1 \pmod{max}) \\ \vee\ &(\alpha = 2 \wedge \beta = 2 \wedge E(\nu,\iota,w,\kappa)) \\ \vee\ &(\alpha = 2 \wedge \beta = 2 \wedge \forall\nu'\forall\iota'\neg E(\nu,\iota,\nu',\iota') \wedge \nu = w \wedge \iota = \kappa) \\ \vee\ &(3 \leq \alpha < D \wedge \beta = \alpha \wedge \nu = w \wedge \iota = \kappa)\ .\end{aligned}$$

The graph $\mathfrak{G}_{\text{reg}} \cong \mathfrak{I}(\mathfrak{G}^*)$ is regular, and the connectivity of $\mathfrak{G}^*$ is preserved in the sense that there is a path from vertex (ν,ι) to (w,κ), for any ι and κ, if and only if there is a path from ν to w in $\mathfrak{G}$.

Step 2. Improving the expansion properties.

The regular graph $\mathfrak{G}_{\text{reg}}$ obtained by the transformation in Step 1 is transformed into an expander graph by iterating a zig-zag product with the expander graph $\mathfrak{X}$ and powering according to the following rules:

$$\mathfrak{H}_0(\mathfrak{G},\mathfrak{X}) := \mathfrak{G}_{\text{reg}}\ ,$$
$$\mathfrak{H}_i(\mathfrak{G},\mathfrak{X}) := \big(\mathfrak{H}_{i-1}(\mathfrak{G},\mathfrak{X}) \circledslash \mathfrak{X}\big)^8\ .$$

Note that we have to apply at least a $(p = 8)$-powering step to obtain a graph with sufficiently good expansion properties which follows from the proof of Lemma 4.11. For larger values of p we would need fewer iterations to construct the final expander; however, its degree would be larger. Hence, this definition justifies the choice of $D = d^{16} = (d^2)^p$ since otherwise the zig-zag product would not be defined. Starting with the D-regular graph $\mathfrak{G}_{\text{reg}}$, the zig-zag product $\mathfrak{G}_{\text{reg}} \circledslash \mathfrak{X}$ is d^2-regular, and 8-powering yields a D-regular graph again. Thus, all graphs $\mathfrak{H}_i$ are D-regular, and in the end, we obtain the graph

$$\mathfrak{H}(\mathfrak{G},\mathfrak{X}) := \mathfrak{H}_\ell(\mathfrak{G},\mathfrak{X})$$

for the smallest integer ℓ satisfying

$$\left(1 - \frac{1}{d^{16}n^2}\right)^{2^\ell} < \frac{1}{2}\ . \qquad (*)$$

Moreover, $\ell \approx \lceil 2 \log n + \log D \rceil$, i.e. the value of ℓ is logarithmically bounded in the size of the input graph.

Since the zig-zag product is defined over the Cartesian product of the vertices, each graph $\mathfrak{H}_i$, for $i \leq \ell$, consists of $n^2 \cdot D^i$ vertices, and a vertex can be represented by one pointer into the set of vertices and numbers each (defining a vertex of the regular graph $\mathfrak{G}_{\text{reg}}$) and a sequence of i numbers in $[D]_0$ determining a particular sub-vertex by which a vertex of $\mathfrak{G}_{\text{reg}}$ is replaced in the i-fold zig-zag product.

To represent vertices of $\mathfrak{H} = \mathfrak{H}_\ell$ by elements of our structure (the input graph $\mathfrak{G}$), we fix a k such that $n^k \geq D^\ell$. Since

$$\begin{aligned} D^\ell &\approx D^{2\log n + \log D} = D^{2\log n} D^{\log D} \\ &\leq n^{2\log D} n^{\log D} = n^{3\log D} \quad , \end{aligned}$$

the number k only depends on D, i.e. we can represent vertices of $\mathfrak{H}_\ell$ by a tuple of numbers whose length is independent of the size of the input graph.

Another important aspect of the chosen parameters is the following. The algorithm relies on tracing, remembering and recombining edge labels. The zig-zag product is defined such that vertices of one graph represent edge labels of the other graph, and after powering, the edge labels of the resulting graph are sequences of edge labels of the former one. Since during the construction we only work with d-regular and D-regular graphs where $D = d^{16}$, sequences of edge labels can be interpreted as numbers represented in base D as well as in base d, and in particular, the conversion between these representations can be done digit by digit since a single digit in the base D representation corresponds to exactly 16 digits in the base d representation (similar to the conversion of hexadecimal numbers into binary).

In summary, we use k-tuples of number variables to represent sequences of labels from $[D]_0$ of length at most ℓ, and we will commonly view a label from $[D]_0$ representing a sequence of 16 values from $[d]_0$. This conversion is definable in DTC by the following formula (similar to the bit-predicate) extracting the κ-th digit (where $0 \leq \kappa < \ell = c \log n$ and the 0-th bit is least significant) of a number in $[D^\ell]_0$ represented by a k-tuple $\bar{\imath}$ of number variables:

$$\begin{aligned} \text{digit}_D(\bar{\imath}, \kappa, \delta) = \exists \bar{\imath}^* \big(& 0 \leq \bar{\imath} - \bar{\imath}^* < D^\kappa \\ & \wedge \left[\text{dtc}_{\bar{\imath}\alpha, \bar{\imath}'\alpha'} \ \bar{\imath}' = \bar{\imath} + D^\kappa \wedge \alpha' = \alpha + 1 \pmod D \right] (\bar{0}, 0, \bar{\imath}^*, \delta) \big) \quad , \end{aligned}$$

Procedure 4.1. $\mathrm{Rot}_{\mathfrak{H}}\left(i, (v, a_0 \dots a_{i-1}), \overline{a}_i\right)$

```
1 for j = 0 to 7 do
2     (a_{i-1}, k_{i,2j}) ← Rot_X(a_{i-1}, k_{i,2j})          /* ā_i = (k_{i,j})_{j=0}^{15} */
3     ((v, a_0, ..., a_{i-2}), ā_{i-1}) ← Rot_H((i − 1), (v, a_0, ..., a_{i-2}), ā_{i-1})
4     (a_{i-1}, k_{i,2j+1}) ← Rot_X(a_{i-1}, k_{i,2j+1})
5 endfor
6 k_{i,0}, ..., k_{i,15} ← k_{i,15}, ..., k_{i,0}          /* reverse edge labels */
7 return ((v, a_0 ... a_{i-1}), ā_i)
```

where $\bar{\imath}^*$ is intended to represent the largest number smaller or equal to $\bar{\imath}$ and divisible by D^κ.

Note that we rely here on the assumption that $n > D$ since otherwise we could not hold values from $[D]_0$ in a single number variable.

Procedure 4.1, computing values of the rotation map, is the core of the algorithm, and it is a syntactically modified version of the procedure originally described by Reingold [Rei05]. The for-loop is partly unravelled to emphasise the correspondence with the definition of the graphs $\mathfrak{H}_i$. Lines 2–4 compute the zig-zag product, i.e. making a zig-step in $\mathfrak{X}$ followed by an edge in $\mathfrak{H}_{i-1}$ and finishing by a zag-step in $\mathfrak{X}$ again. The for-loop corresponds to the powering operation by concatenating 8 edges in the zig-zag product graph. Altogether, the procedure computes, given an input $\left(\ell, (v, a_0 \dots a_{\ell-1}), a_\ell\right)$, the value of the rotation map of $\mathfrak{H}_\ell$, i.e. some pair $\left((w, \beta_0 \dots \beta_{\ell-1}), \beta_\ell\right)$ consisting of a vertex and an edge label. Due to the correspondence of edge labels of $\mathfrak{H}_i$ and vertices of $\mathfrak{X}$, given an identifier $(v, a_0, a_1, \dots, a_{\ell-1})$ of a vertex in $\mathfrak{H}_\ell$, the initial subsequence of the form $(v, a_0, a_1, \dots, a_{i-1}, a_i)$ can be regarded as an identifier of a vertex in $\mathfrak{H}_i$ and an edge label (a_i). Hence, recursive calls of the procedure simply work on an initial part of the sequence stored in memory. Furthermore, while traversing edges, we do not need to memorise which outgoing edge we took, but instead it suffices to know via which incoming edge we entered the target vertex. Thus it is possible to successively overwrite the input sequence with the output obtained from the recursive calls, and due to this in-place modification each level of the recursion only needs D more "bits" of memory. For more details on the correctness of the procedure see [Rei05].

The current state of the program during its execution is completely determined by the values of the following variables:

- the current recursion depth $\delta \leq \ell$,
- an ℓ-tuple $\iota = (j_1, \ldots, j_\ell)$ storing the values of the loop variables $j \in \{0, \ldots, 7\}$ on each recursion level, and
- a (single) program counter ρ determining the current line in the program.

Actually, we do not have to maintain the value of the program counter on the call stack since there is only one possible return address from the recursive call of the procedure, i.e., after a return statement, the program state can always be reconstructed from the recursion depth and the value of the loop variable on that level. Further, a configuration of the program is given by the current program state, determined by the variables δ, $\bar{\iota}$ and ρ as discussed before, and the memory content $(\nu, a_0, \ldots, a_\ell)$, represented by a vertex variable ν (a vertex in the regular graph $\mathfrak{G}_{\text{reg}}$) and an ℓ-tuple of number variables $\overline{\alpha}$.

Note that, in the formulae, we have to represent the ℓ tuples $\bar{\iota}$ and $\overline{\alpha}$ by k-tuples $\bar{\iota}^*$ and $\overline{\alpha}^*$ ranging over the whole numeric domain. Nevertheless, to keep the formulae more readable, we directly refer to the components α_i of the ℓ-tuple $\overline{\alpha}$ as the value of α_i can be defined by

$$(\alpha_i = \mu) \;\equiv\; \mathrm{digit}_D(\overline{\alpha}^*, i, \mu) \;\; .$$

The same applies for the tuple $\bar{\iota}$. Further, a value $\alpha_i \in [D]_0$ represents a sequence $(\kappa_0, \ldots, \kappa_{15}) \in [d]_0^{16}$, and we use the shorthand notation

$$\overline{\kappa} = \alpha_i \;\text{ for }\; \bigwedge_{j=0}^{15} \mathrm{digit}_d(\alpha_i, j, \kappa_j) \;\; .$$

Furthermore, we use atomic formulae $\bar{\iota} = \bar{\iota}'$ to say that the tuples agree in all components as well as the shorthand notation $\bar{\iota}_{-\Delta} = \bar{\iota}'_{-\Delta}$ for a set Δ of indices to indicate that the tuples agree in all but the components listed in Δ.

The following formulae NEXT, ZIG, ZAG, REVERSE, EDGE, and END describe parts of the computation of the algorithm. NEXT describes the transition from one program state to the next one and keeps track of the changes of the program counter ρ, the recursion level δ and the loop variables $\bar{\iota}$. The formulae ZIG and ZAG describe the manipulations on the memory determined

by the zig and zag steps of the zig-zag product performed in lines 2 and 4, respectively. Note that line 3 is not explicitly mentioned in the formula since it consists of either a recursive call of the procedure continuing in line 2, or, on the bottom-most level of the recursion, when $\delta = 0$, we follow an edge in the graph $\mathfrak{G}_{\text{reg}}$, which is captured by the formula EDGE, and the control flow continues at line 4.

$$
\begin{aligned}
\text{NEXT} \coloneqq\ & (\delta > 0 \wedge \rho = 2) \\
& \quad \to (\delta' = \delta - 1 \wedge \bar{\iota}'_{-\{\delta'\}} = \bar{\iota}_{-\{\delta'\}} \wedge \iota'_{\delta'} = 1 \wedge \rho' = 2) \\
& \qquad [\text{recursively call procedure after line 2;} \\
& \qquad\quad \text{continue in line 2 and reset loop variable on new rec. level}] \\
& \wedge (\delta = 0) \to (\delta' = 1 \wedge \rho' = 4 \wedge \bar{\iota}' = \bar{\iota}) \\
& \quad [\text{unless bottom level of recursion reached; then return to line 4}] \\
& \wedge (\delta > 0 \wedge \rho = 4 \wedge \iota_\delta < 8) \\
& \quad \to (\delta' = \delta \wedge \bar{\iota}'_{-\{\delta\}} = \bar{\iota}_{-\{\delta\}} \wedge \iota'_\delta = \iota_\delta + 1 \wedge \rho' = 2) \\
& \quad [\text{if loop not finished, increment loop variable, continue at line 2}] \\
& \wedge (\delta > 0 \wedge \rho = 4 \wedge \iota_\delta = 8) \to (\delta' = \delta \wedge \bar{\iota}' = \bar{\iota} \wedge \rho' = 6) \\
& \qquad [\text{if the loop is finished, continue at line 6}] \\
& \wedge (\ell > \delta > 0 \wedge \rho = 6) \to (\delta' = \delta + 1 \wedge \bar{\iota}' = \bar{\iota} \wedge \rho' = 4) \\
& \quad [\text{after reversing, return from recursive call and continue at line 4}] \\
& \wedge (\delta = \ell \wedge \rho = 6) \to (\delta' = \delta \wedge \bar{\iota}' = \bar{\iota} \wedge \rho' = 7) \\
& \qquad [\text{if already at the top level, go to end configuration instead}]
\end{aligned}
$$

$$
\begin{aligned}
\text{ZIG} \coloneqq\ & (\rho = 2) \to \exists \kappa_0, \dots, \kappa_{15} \exists \kappa'_0, \dots, \kappa'_{15} \big(\bar{\kappa} = \alpha_\delta \wedge \bar{\kappa}' = \alpha'_\delta \\
& \qquad \wedge \bar{\kappa}_{-\{2\iota_\delta\}} = \bar{\kappa}'_{-\{2\iota_\delta\}} \wedge \bar{\alpha}_{-\{\delta,\delta-1\}} = \bar{\alpha}'_{-\{\delta,\delta-1\}} \\
& \qquad\quad \wedge E_{\mathfrak{X}}(\alpha_{\delta-1}, \kappa_{2\iota_\delta}, \alpha'_{\delta-1}, \kappa'_{2\iota_\delta}) \big) \\
& \qquad [\text{make a zig-step in } \mathfrak{X} \text{ in line 2}]
\end{aligned}
$$

$$
\begin{aligned}
\text{ZAG} \coloneqq\ & (\rho = 4) \to \exists \kappa_0, \dots, \kappa_{15} \exists \kappa'_0, \dots, \kappa'_{15} \big(\bar{\kappa} = \alpha_\delta \wedge \bar{\kappa}' = \alpha'_\delta \\
& \qquad \wedge \bar{\kappa}_{-\{2\iota_\delta+1\}} = \bar{\kappa}'_{-\{2\iota_\delta+1\}} \\
& \qquad \wedge \bar{\alpha}_{-\{\delta,\delta-1\}} = \bar{\alpha}'_{-\{\delta,\delta-1\}} \\
& \qquad\quad \wedge E_{\mathfrak{X}}(\alpha_{\delta-1}, \kappa_{2\iota_\delta+1}, \alpha'_{\delta-1}, \kappa'_{2\iota_\delta+1}) \big) \\
& \qquad [\text{make a zag-step in } \mathfrak{X} \text{ in line 4}]
\end{aligned}
$$

$$\textsc{edge} \coloneqq (\delta = 0) \to \big(\overline{\alpha}_{-\{0\}} = \overline{\alpha}'_{-0} \wedge E(v, \alpha_0, v', \alpha'_0)\big)$$
$$\text{[at the bottom-most level follow an edge in } \mathfrak{G}_{\text{reg}}\text{]}$$

$$\textsc{reverse} \coloneqq (\delta > 0 \wedge \rho = 6) \to \exists \kappa_0, \dots, \kappa_{15} \exists \kappa'_0, \dots, \kappa'_{15}$$
$$\big(\overline{\kappa} = \alpha_{\delta+1} \wedge \overline{\kappa}' = \alpha'_{\delta+1} \wedge \bigwedge_{i=0}^{15} \kappa_i = \kappa'_{15-i}\big)$$
$$\text{[reverse the edge labels since we keep incoming labels in memory]}$$

$$\textsc{end} \coloneqq (\rho = 7) \to (\overline{\alpha} = \overline{\alpha}' \wedge \bar{\imath} = \bar{\imath}' \wedge \rho = \rho' \wedge \delta = \delta')$$
$$\text{[do not change anything, loop in the end configuration.]}$$

In the formulae ZIG and ZAG, $E_{\mathfrak{X}}$ is the formula describing the edge relation of the finite expander $\mathfrak{X}$ hard-coded into the algorithm.

In summary, using these subformulae, we can formalise the computation of the procedure and obtain the following interpretation $\mathfrak{I}_{\text{exp}}$ such that $\mathfrak{I}(\mathfrak{G}_{\text{reg}})$ is isomorphic to the expander graph $\mathfrak{G}_{\text{exp}}$ constructed by Reingold's algorithm.

$$\begin{aligned}
\delta_1(v, \alpha_1, \dots, \alpha_k) &\coloneqq \bigwedge_{i=1}^{k} \exists \mu(\alpha_i < \mu) \\
\delta_2(\iota) &\coloneqq (\iota = \iota) \\
\varepsilon_1((v, \overline{\alpha}), (w, \overline{\beta})) &\coloneqq (v = w) \wedge (\alpha = \beta) \\
\varepsilon_2(\iota, \kappa) &\coloneqq (\iota = \kappa) \\
\varphi_<(\iota, \kappa) &\coloneqq (\iota < \kappa) \\
\varphi_E(u, \overline{\alpha}, v, \overline{\beta}) &\coloneqq \exists \ell \big(\text{“}\ell \text{ satisfies Equation } (*)\text{”} \\
&\qquad \wedge [\mathrm{dtc}_{v\overline{\alpha}\delta\bar{\imath}\rho, v'\overline{\alpha}'\delta'\bar{\imath}'\rho'} \varphi_{\text{procedure}}](u\overline{\alpha}\ell, \overline{0}, 2\,,\ v\overline{\beta}\ell, \overline{7}, 7)\big)
\end{aligned}$$

where

$$\varphi_{\text{procedure}} \coloneqq \textsc{next} \wedge \textsc{zig} \wedge \textsc{edge} \wedge \textsc{zag} \wedge \textsc{reverse} \wedge \textsc{end}\ .$$

Note that we crucially rely on the ability of DTC to define all arithmetic operations such as addition, multiplication, exponentiation etc. on the number sort. However, we do not make use of counting terms. The only connection needed between the number universe and the graph vertices is the addressing of edges by numbers smaller than the degree of the graph which is made possible by the given local ordering.

Step 3. Searching for a connecting path.
By Lemma 4.11, the constructed graph $\mathfrak{G}_{\text{exp}}$ has a large spectral gap; in particular $1 - \lambda(\mathfrak{G}_{\text{exp}}) \geq 1/2$. Hence, Theorem 4.3 implies that the edge expansion ratio of $\mathfrak{G}_{\text{exp}}$ is at least $D/4$. Finally, Proposition 4.5 gives a bound on the diameter Δ of the connected components of $\mathfrak{G}_{\text{exp}}$, and this bound is expressible in DTC.

Any start vertex v in $\mathfrak{G}_{\text{exp}}$ together with a sequence of edge labels $\overline{\pi} \in [D]_0^*$, called *directions*, defines a path through the graph. In the following, we restrict our attention to directions of length approximately Δ. In particular, there is an ℓ such that $n^\ell \geq D^\Delta$, such that we can use ℓ-tuples of number variables to represent directions of length at least Δ.

The formula σ describes a single step on a path according to a given tuple of directions $\overline{\pi}$, i.e. σ states that, if v is the ι-th vertex on a path, then v' is the next (i.e. the $\iota + 1$st) vertex when following the ι-th edge label in $\overline{\pi}$.

$$\sigma(v, \bar{\iota}, v', \bar{\iota}', \overline{\pi}) = (\bar{\iota}' = \bar{\iota} + 1) \wedge \exists \eta \exists \eta' \begin{pmatrix} \operatorname{digit}_D(\overline{\pi}, \bar{\iota}, \eta) \\ \wedge E(v, \eta, v', \eta') \end{pmatrix} .$$

The transitive closure of σ, for fixed directions $\overline{\pi}$, yields the relation "reachability via subsequences of directions $\overline{\pi}$" in $\mathfrak{G}_{\text{exp}}$. Since the directions define unique successors, the transitive closure is even DTC-definable by

$$\chi(s, t, \overline{\pi}) = [\operatorname{dtc}_{v\bar{\iota}, v'\bar{\iota}'} \sigma(v\bar{\iota}, v'\bar{\iota}', \overline{\pi})](s0, t\Delta)$$

stating that the vertex t is reachable from s in exactly Δ many steps by following the directions $\overline{\pi}$.

Finally, the following formula expresses that t is reachable from s via some path of length Δ.

$$\rho(s, t) = \exists \overline{\pi} \,.\, \chi(s, t, \overline{\pi}) \ .$$

Theorem 4.12. The formula

$$\hat{\rho}(u, v) = \exists \alpha_0 \alpha_1 \ldots \alpha_k \exists \beta_0 \beta_1 \ldots \beta_k (\mathfrak{I}_{\text{rot}} \circ \mathfrak{I}_{\text{reg}} \circ \mathfrak{I}_{\text{exp}})(\rho)(u\overline{\alpha}, v\overline{\beta})$$

defines the transitive closure of E in a two-sorted undirected locally ordered graph $\mathfrak{G}^*$, i.e. $\mathfrak{G}^* \vDash \hat{\rho}(u, v)$ if and only if the vertices u and v are connected in the graph $\mathfrak{G}$.

Proof. First of all, since the constructed expander graph $\mathfrak{G}_{\text{exp}}$ is D-regular, every ℓ-tuple of numbers represents a valid sequence of edge labels, and conversely, every sequence of edge labels of length at most Δ is represented by an ℓ-tuple of numbers.

Note that each vertex of $\mathfrak{G}$ is first replaced by a cycle of vertices to obtain the regular graph $\mathfrak{G}_{\text{reg}}$, and during the expander construction via the zig-zag product, these vertices are replaced by more and more vertices, yielding the cloud of v.

By construction, all self-loops in $\mathfrak{G}_{\text{reg}}$ give rise to edges in $\mathfrak{G}_{\text{exp}}$ that stay inside the same cloud. Hence, if the cloud of v is reachable from a particular vertex $(u, \overline{\alpha})$ in the cloud of u in $\delta \leq \Delta$ many steps in $\mathfrak{G}_{\text{exp}}$, then there also exists some vertex (v, β) in the cloud of v that is reachable from $(u, \overline{\alpha})$ in exactly Δ many steps. Q.E.D.

4.4 Canonisation of locally ordered undirected graphs

Even if we are not able to define a linear order on the universe of a structure, it may still be possible to interpret an isomorphic linearly ordered structure in a given one. This is more precisely captured by the concept of *canonisation*.

Definition 4.13. Let $\mathcal{K}$ be a class of finite τ-structures, and let $\sim$ be an equivalence relation on $\mathcal{K}$. Recall that $\mathcal{K}_< = \{(\mathfrak{A}, <) : \mathfrak{A} \in \mathcal{K},\ < \text{ a lin. order on } A\}$. A function $f : \mathcal{K} \to \mathcal{K}_<$ mapping a structure $\mathfrak{A}$ to the ordered structure $f(\mathfrak{A}) = (\mathfrak{A}', <)$ such that

(1) $\mathfrak{A} \sim \mathfrak{A}'$ and
(2) $\mathfrak{A} \sim \mathfrak{B}$ implies $f(\mathfrak{A}) \cong f(\mathfrak{B})$ for all $\mathfrak{A}, \mathfrak{B} \in \mathcal{K}$

is called a *canonisation function for $\sim$ on $\mathcal{K}$*.

That is, a canonisation function picks an ordered representative of each equivalence class. We are particularly interested in the definability of such ordered representatives which is made precise in the following.

Definition 4.14. Let $\mathcal{K}$ be a class of finite τ-structures, and let $\sim$ be an equivalence relation on $\mathcal{K}$. We say that $(\mathcal{K}, \sim)$ *admits $\mathcal{L}$-definable canonisation* if there exists an $\mathcal{L}[\tau, \tau \cup \{<\}]$-interpretation $\mathfrak{I}$ such that each $\mathfrak{A} \in \mathcal{K}$ fulfils the admissibility conditions and $f : \mathfrak{A} \mapsto \mathfrak{I}(\mathfrak{A})$ is a canonisation function

for $\sim$ on $\mathcal{K}$. We shortly say that $\mathcal{K}$ admits $\mathcal{L}$-definable canonisation if this is the case for $(\mathcal{K}, \cong)$.

Intuitively, if $\mathcal{K} \subseteq \mathrm{Str}[\tau]$ admits $\mathcal{L}$-definable canonisation, it is possible, for every structure $\mathfrak{A} \in \mathcal{K}$, to interpret a linearly ordered expansion $\mathfrak{A}_<$ whose unordered τ-reduct is isomorphic to $\mathfrak{A}$ in the structure $\mathfrak{A}$ itself.

Etessami and Immerman showed that the class of two-way locally ordered (directed) graphs admits TC+C-definable canonisation [EI95b, EI00]. The earlier result of Immerman [Imm87, Imm88] that TC captures NLogspace then implies that TC+C captures on the class of two-way locally ordered graphs.

The interpretation can be obtained using the following two techniques. First, using the edge labels induced by the local orderings, it is possible to linearly order each *weakly* connected component of the graph. Second, allowing counting terms permits the definition of a pre-order on the weakly connected components regarding their size and further criteria such that each equivalence class of the pre-order only contains isomorphic components.

In the following, we explain this approach in more detail and show that, by using the previously obtained DTC formula defining the transitive closure, we obtain a DTC+C-definable canonisation for locally ordered *un*directed graphs.

Lemma 4.15. Let $\mathfrak{G}$ be an undirected locally ordered graph. There is a formula defining an ordering on the vertices reachable from v in $\mathfrak{G}$ by lexicographically comparing the paths of length $\Delta(\mathfrak{G})$ connecting them to v in $\mathfrak{G}_{\mathrm{exp}}$.

Proof. Let $\chi(s, t, \overline{\pi})$ be the formula from Step 3 stating that the vertices s and t in the expander graph are connected via directions $\overline{\pi}$. Then $\varphi(v, x, y) := \exists\overline{\pi}(\chi(v, x, \overline{\pi}) \wedge \forall\overline{\pi}'(\chi(v, y, \overline{\pi}') \rightarrow \overline{\pi} <_{\mathrm{lex}} \overline{\pi}'))$ states that x is reachable from v via some path $\overline{\pi}$ and that all paths (if any) connecting v to y are lexicographically larger. Hence, $\hat{\varphi}(v, x, y) = (\mathfrak{I}_{\mathrm{rot}} \circ \mathfrak{I}_{\mathrm{reg}} \circ \mathfrak{I}_{\mathrm{exp}})\varphi(v\bar{0}, x\bar{0}, y\bar{0})$ is the desired formula. Q.E.D.

Lemma 4.16. The class of locally ordered undirected graphs (with constants) admits DTC+C-definable canonisation.

Proof. Let $\mathfrak{G} = (V, E, \leq, c_1, \ldots, c_k)$ be an undirected locally ordered graph with constants, and let $\mathfrak{G}^*$ be the corresponding canonical two-sorted structure.

For each connected component, the choice of some vertex v defines a total ordering on all vertices of the component by Lemma 4.15. Thus, the choice of a vertex v induces an adjacency matrix A_v in which the rows and columns are given in the order of the induced ordering. Let $\mathrm{code}(A_v)$ be a representation of the adjacency matrix A_v as a string that is obtained by concatenating all rows of the matrix. Towards picking a vertex v as the smallest one in a connected component, we consider an ordering $<_v$ to be smaller than $<_w$ if $\mathrm{code}(A_v) <_{\mathrm{lex}} \mathrm{code}(A_w)$, and we pick the vertex v inducing the smallest $\mathrm{code}(A_v)$ as the smallest vertex in the component.

Note that it might be the case that there is a unique minimal adjacency matrix, but different orderings $<_v$ and $<_w$, say, inducing this matrix. However, this can only happen if there is an automorphism of the connected component exchanging v and w.

First, we define a preorder $\preccurlyeq$ on the connected components of the graph based on four criteria in the following precedence:

- size of the component (non-decreasing);
- lexicographic value of the minimal adjacency sub-matrix of the component (non-decreasing);
- encoding of the minimal adjacency sub-matrix (lexicographic, non-decreasing);
- containment of constants $c_1, \ldots, c_k$ (increasing: $c_1 < c_2 < \cdots < c_k <$ none).

Note that any two vertices u and v belonging to the same connected component or to different isomorphic components are equivalent in the sense that $u \preccurlyeq v$ and $v \preccurlyeq u$ hold.

The preorder is definable by

$$\begin{aligned} x \preccurlyeq y \;:=\; \exists\kappa_1 \exists\kappa_2 \big[& \mathrm{CompSize}(x, \kappa_1) \wedge \mathrm{CompSize}(y, \kappa_2) \\ & \wedge \big(\kappa_1 < \kappa_2 \vee (\kappa_1 = \kappa_2 \wedge (\mathrm{val}(x) < \mathrm{val}(y) \vee (\mathrm{val}(x) = \mathrm{val}(y) \\ & \wedge \text{“Comp}(y) \text{ contains no constants or, if Comp}(x) \text{ contains} \\ & \quad \text{a constant, Comp}(y) \text{ contains no smaller constant”}))) \big) \big] \end{aligned}$$

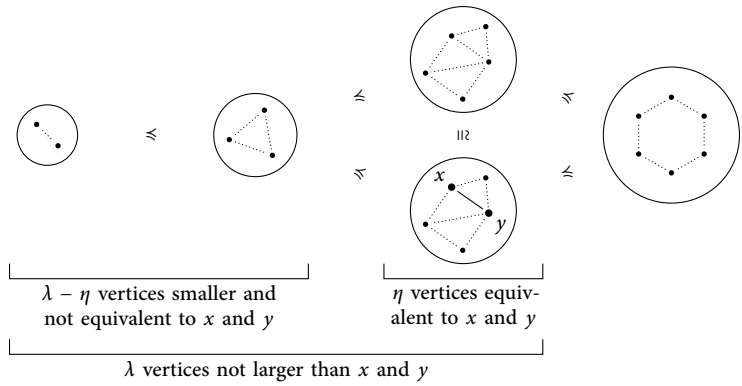

Figure 4.5. Ordering of the connected components

where

$$\mathrm{CompSize}(x,\kappa) \;:=\; \#y\hat{\rho}(x,y) = \kappa$$

$$\begin{aligned}\mathrm{val}(x) = \mathrm{val}(y) \;:=\; \exists\kappa\Big[&\mathrm{CompSize}(x,\kappa) \wedge \mathrm{CompSize}(y,\kappa)\\ &\wedge \forall\alpha\forall\beta\big((\alpha<\kappa\wedge\beta<\kappa)\to \mathrm{Adj}(x,\alpha,\beta) = \mathrm{Adj}(y,\alpha,\beta)\big)\Big]\end{aligned}$$

$$\begin{aligned}\mathrm{val}(x) < \mathrm{val}(y) \;:=\; \exists\kappa\Big[&\mathrm{CompSize}(x,\kappa) \wedge \mathrm{CompSize}(y,\kappa)\\ &\wedge \exists\alpha\exists\beta\big(\alpha<\kappa\wedge\beta<\kappa\wedge \mathrm{Adj}(x,\alpha,\beta) < \mathrm{Adj}(y,\alpha,\beta)\big)\\ &\wedge \forall\beta'\big(\beta'<\beta\to \mathrm{Adj}(x,\alpha,\beta') = \mathrm{Adj}(y,\alpha,\beta')\big)\\ &\wedge \forall\alpha'\forall\beta'\big((\alpha'<\alpha\wedge 0\le\beta'<\kappa)\\ &\qquad\to \mathrm{Adj}(x,\alpha',\beta') = \mathrm{Adj}(y,\alpha',\beta')\big)\Big]\end{aligned}$$

$$\begin{aligned}\mathrm{Adj}(x,\alpha,\beta) = \lambda \;:=\; \exists\kappa\exists y\exists z\Big[&\mathrm{CompSize}(x,\kappa)\wedge\alpha<\kappa\wedge\beta<\kappa\\ &\wedge\text{“}y\text{ is the }\alpha\text{-th element of the ordering }<_x\text{”}\\ &\wedge\text{“}z\text{ is the }\beta\text{-th element of the ordering }<_x\text{”}\\ &\wedge\exists\lambda'\,.\,E(y,\lambda,z,\lambda')\Big]\\ &\big[\text{the entry at position }\alpha,\beta\text{ in the adjacency matrix is }\lambda\\ &\qquad\text{such that }\beta\text{ is the }\lambda\text{-th neighbour of }\alpha.\big]\end{aligned}$$

Using this preorder, we obtain the following interpretation $\mathfrak{I}_{\mathrm{ord}}$ interpreting an ordered graph $\mathfrak{G}_{\mathrm{ord}}(V', E', <)$ in a two-sorted locally ordered undirected graph $\mathfrak{G}^*$ arising from $\mathfrak{G} = (V, E, \leq)$ and satisfying $(V', E') \cong (V, E)$.

$$
\begin{aligned}
\delta(\mu) &= \exists \nu(\mu < \nu) \\
\varepsilon(\mu, \nu) &= (\mu = \nu) \\
\varphi_<(\mu, \nu) &= (\mu < \nu) \\
\varphi_E(\mu, \nu) &= \exists x \exists y \exists \lambda \Big(\big(Exy \wedge \#z(z \preccurlyeq x) = \#z(z \preccurlyeq y) = \lambda \big) \\
&\qquad \big[\text{There exist two connected vertices } x \text{ and } y \\
&\qquad\quad \text{each comparable to } \lambda \text{ non-larger vertices,}\big] \\
&\qquad \wedge \exists \kappa(\mathrm{CompSize}(x, \kappa) \wedge \mathrm{CompSize}(y, \kappa) \\
&\qquad\quad \big[\text{living in connected components of the same size } \kappa,\big] \\
&\qquad \wedge \exists \eta \big(\#z(x \preccurlyeq z \wedge z \preccurlyeq x) = \#z(y \preccurlyeq z \wedge z \preccurlyeq y) = \eta \big) \\
&\qquad\quad \big[\text{each being equivalent to } \eta \text{ other vertices}\big] \\
&\qquad \wedge \exists \alpha \exists \beta (\alpha < \kappa \wedge \beta < \kappa \wedge |\alpha - \beta| < \kappa \\
&\qquad\quad \wedge \alpha \equiv (\mu - (\lambda - \eta)) \pmod{\kappa} \\
&\qquad\quad \wedge \beta \equiv (\nu - (\lambda - \eta)) \pmod{\kappa} \\
&\qquad\quad \big[\text{and being the } \alpha\text{-th and } \beta\text{-th vertex in the same component.}\big] \\
&\qquad \wedge \exists u \big(\forall v(\mathrm{val}(u) \leq \mathrm{val}(v)) \wedge \exists \xi \,.\, \mathrm{Adj}(u, \alpha, \beta) = \xi \big) \big) \\
&\qquad\quad \big[\text{And, according to the min. adjacency matrix,} \\
&\qquad\quad\quad \text{there is an edge between } \alpha \text{ and } \beta.\big]
\end{aligned}
$$

Q.E.D.

Corollary 4.17. DTC+C captures Logspace on locally ordered undirected graphs.

Proof. On the one hand, DTC+C formulae can be evaluated in Logspace as each formula has only a fixed number of counting terms, so these counting terms can be stored and evaluated using only logarithmic memory. On the other hand, DTC captures Logspace on ordered graphs [Imm87], and the class of locally ordered undirected graphs admits DTC+C-definable canonisation. Q.E.D.

4.5 Discussion

It is clear that DTC without counting fails to capture Logspace on locally ordered graphs since any local order relation is empty in a graph without edges, and hence DTC alone cannot express Logspace-computable queries such as EVEN.

On the other hand, it is not immediately obvious whether reachability is definable in plain DTC on locally ordered graphs or not. The approach taken in the previous section by describing Reingold's algorithm is not feasible; although the formula defining the transitive closure does not use counting terms, it heavily relies on being able to express arithmetic computations on the number sort and to encode paths for which a linear order is needed that must be at least as long as the size of the graph.

Also from a computational point of view it is still not clear how Logspace computations can be characterised. Various models were analysed and compared, including their shortcomings and possible remedies. One prominent feature usually associated with logspace computations is the use of only constantly many variables pointing into the input structure which is incorporated in so-called *jumping automata on graphs* (JAG) introduced by Cook and Rackoff [CR80] and DTC. However, this characterisation is not universal as is shown by Reingold's algorithm which also uses, besides constantly many pointers into the input structure, logarithmically many variables holding bit values. This is neither possible in DTC on structures without a total order nor in the JAG model.

In contrast to logics, which exhibit a clear conceptual difference between variables referencing elements of the structure and variables referencing numbers (at least when the structure is not totally ordered), in some programming languages allowing pointers, this distinction is blurred however, e.g. in languages such as C it is common to manipulate the address of a pointer by arithmetic operations. In an effort to characterise and get a better understanding of the computational limitations of various models of pointer languages, Hofmann and Schöpp [HS08] introduced and investigated the programming language purple (short for *Pur*e *P*ointer *L*anguag*e*), an imperative programming language allowing iteration via while- and forall-loops and using a constant number of so-called *pure* pointers, i.e. purple can be seen as a programming language extending the JAG formalism by forall-loops which provide a means to access all elements in a structure regardless

of whether they are reachable in the Gaifman graph or not. However, the semantics of PURPLE requires the programs to be order-invariant in the sense that the output must not depend on the order in which the elements in a forall-loop are iterated. Essentially, the pure pointers in PURPLE reference elements of the input structure (e.g. a graph) and may only be updated to point to either a constant or to an element related to the current one (i.e. connected in the Gaifman graph). This especially means that arbitrary placement and pointer arithmetic such as "move to the next vertex in the implicitly given input-ordering" are not allowed, which makes pointer variables in PURPLE comparable to variables in logics evaluated on unordered structures.

In fact, DTC formulae can be evaluated by PURPLE programs [HS08, HS10]. However, since it is straightforward to implement a program in PURPLE that tests whether the input structure has an even number of elements, it is strictly more powerful than DTC. Further, Hofmann and Schöpp [SH08] develop an involved argument showing that there exist classes of graphs on which every PURPLE program of bounded complexity can be simulated by a so-called *local program*, i.e. by a program that does not contain loops and hence cannot move its pointers out of a local neighbourhood around their initial position.

Theorem 4.18 ([SH08, Corollary 24]). There is no PURPLE program that decides undirected reachability on locally ordered graphs of degree 3.

Since PURPLE programs can evaluate DTC formulae, this implies a similar inexpressibility result for DTC.

Corollary 4.19 ([SH08, Corollary 25]). Reachability in locally ordered undirected graphs of degree 3 is not definable by a DTC formula.

This limitation reminds of a similar result obtained by Grädel and McColm [GM95]. They identify much simpler classes of graphs on which the expressive power of DTC collapses to plain FO. It would be interesting to further investigate the expressive power of DTC on the class of locally ordered graphs and to derive a locality result, possibly based on the class of structures described by Hofmann and Schöpp. However, at the moment, the available tools for proving bounds on the expressiveness of DTC are rather limited.

It is worth to note that, on the class of (plain) graphs, adding a counting mechanism to DTC does not prevent the collapse mentioned above; Etessami

and Immerman proved that even DTC+C collapses to FO+C on graphs [EI95a]. However, as we showed, in the presence of a local ordering, adding a counting mechanism increases the expressive power sufficiently to capture all of Logspace.

Part II

Compositional Model Checking

5 COMPOSITIONAL MODEL CHECKING OF WEAK MSO

In this chapter, we shift our attention to infinite structures that admit a representation by means of finite systems of equations. Such an equation system defines a (finite) set of structures inductively by describing isomorphism types and how to compose these to obtain new structures. One of the most intuitive examples, described later in more detail, is the infinite binary tree, which can be constructed from a root node with two successors that are themselves the root nodes of two infinite binary trees, again. Conversely, the defining equations also yield information about how a structure can be decomposed into pieces and thus make such structures ideally suited for compositional model checking techniques. These techniques rely on being able to evaluate formula decompositions, i.e. formulae derived from the given formula, on the parts of a structure and appropriately combine these results to answer whether the original formula holds in the whole structure.

We start with formally introducing the class of inductive structures and continue with presenting an algorithm for obtaining the desired formula decompositions. Further, we introduce a new model checking game for WMSO on inductive structures and prove its soundness and completeness. While the arena of a classical model checking game for (W)MSO is infinite if the structure is infinite but admits only finite plays, our new game exploits formula decompositions to obtain a finite arena. Admittedly, we sacrifice the simple (reachability) winning condition for a Büchi winning condition since the arena admits infinite plays, now. We conclude by showing how the

winning condition can further be modified to capture model checking of WMSO extended with the so-called unbounding quantifier.

5.1 Inductive Structures

We investigate weak monadic second-order logic on *inductive structures* that are defined by a system of equations, similar to the definition of vertex replacement graphs but strictly monotone. Further, such structures admit bounded clique-width decompositions with regular labels.

In the following, we will frequently speak of *indexed structures* and *indexed elements*. The latter are elements paired with a finite word (called index) over a specific alphabet Σ. An indexed structure consists of a universe of indexed elements. We usually identify a plain structure with the indexed structure in which all elements are indexed by the empty word ε.

Definition 5.1. Given, for each $R_i \in \tau = \{R_1, \dots, R_t\}$ with arity r_i, a function $f_i : \{1, \dots, k\}^{r_i} \to \{\bot, \top\}$, and indexed structures $\mathfrak{A}_1, \dots, \mathfrak{A}_k$ over $\Sigma \supseteq [k]$, we define the *(k-ary) disjoint sum with connections* $\mathfrak{B} = (B, R_1^{\mathfrak{B}}, \dots, R_t^{\mathfrak{B}})$, denoted $\bigoplus_{\bar{f}}(\mathfrak{A}_1, \dots, \mathfrak{A}_k)$, by

- $B := \{(a, cw) : (a, w) \in A_c,\ c \in [k]\}$ and
- $\big((b_1, c_1w_1), \dots, (b_{r_i}, c_{r_i}w_{r_i})\big) \in R_i^{\mathfrak{B}}$ if and only if
  - $|\{c_1, \dots, c_{r_i}\}| = 1$ and $(b_1, \dots, b_{r_i}) \in R_i^{\mathfrak{A}_{c_1}}$ or
  - $|\{c_1, \dots, c_{r_i}\}| > 1$ and $f_i(c_1, \dots, c_{r_i}) = \top$.

That is, $\mathfrak{B}$ is constructed by taking the disjoint union of the structures $\mathfrak{A}_c$, and adding tuples spanning multiple components according to the given functions f_i. It is implicit in the definition that unary relations are inherited from the components, whereas only at least binary relations are augmented with additional tuples. Intuitively, the indices keep track of the origin of elements. We let $\mathfrak{B}[c] := \mathfrak{B} \cap \{(b, w) \in B : w = cw'\}$ denote the c-th *component* of the disjoint sum, and, as expected, $\mathfrak{B}[c]$ is isomorphic to $\mathfrak{A}_c$ via $\pi_c : (b, cw') \mapsto (b, w')$. Furthermore, defining $\mathfrak{B}[\varepsilon] := \mathfrak{B}$, the notation naturally extends to $\mathfrak{B}[cw] := (\mathfrak{B}[c])[w] = \mathfrak{B} \cap \{(b, v) \in B : v = cww'\}$.

Example 5.2. Given $f(1,2) = \top$ and $f(c_1, c_2) = \bot$ otherwise, the binary disjoint sum with connections $\begin{smallmatrix}\bullet\\ \uparrow\\ \bullet\end{smallmatrix} \oplus_f \begin{smallmatrix}\bullet\\ \downarrow\\ \bullet\end{smallmatrix} = \begin{smallmatrix}1 & \bullet & \to & \bullet & 2\\ & \uparrow & \searrow\!\!\nearrow & \downarrow & \\ 1 & \bullet & \to & \bullet & 2\end{smallmatrix}$.

Definition 5.3. A *system of structure equations* $\mathcal{D}$ over $\tau = \{R_1, \ldots, R_t\}$ has the form

$$\mathcal{D} = \begin{cases} \Lambda_1 = \mathfrak{A}_1^1 \oplus \mathfrak{A}_1^2 \oplus \ldots \oplus \mathfrak{A}_1^{k_1} & \text{with } f_{1,1}, \ldots, f_{1,t} \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ \Lambda_n = \mathfrak{A}_n^1 \oplus \mathfrak{A}_n^2 \oplus \ldots \oplus \mathfrak{A}_n^{k_n} & \text{with } f_{n,1}, \ldots, f_{n,t} \end{cases}$$

where each $\mathfrak{A}_i^c$ is either a *finite* structure or one of the formal variables $\Lambda_1, \ldots, \Lambda_n$ and each $f_{i,j}$ is a function $\{1, \ldots, k_i\}^{r_j} \to \{\bot, \top\}$. Further, we let $\lambda(i,c) = m$ if $\mathfrak{A}_i^c = \Lambda_m$ and $\lambda(i,c) = \text{fin}$ otherwise, and we denote by $\mathrm{Str}(\mathcal{D}) := \{\mathfrak{A}_i^c : \lambda(i,c) = \text{fin}\}$ the set of finite structures occurring in $\mathcal{D}$.

Definition 5.4. A system of structure equations defines an operation on n-tuples of τ-structures as follows. Let $\mathfrak{B}_1, \ldots, \mathfrak{B}_n$ be τ-structures to substitute for variables on the right-hand side of the equations in $\mathcal{D}$. Then, we define the new left-hand side structures $(\mathfrak{C}_1, \ldots, \mathfrak{C}_n) =: \mathcal{D}(\mathfrak{B}_1, \ldots, \mathfrak{B}_n)$ by:

$$\mathfrak{C}_i = \bigoplus\nolimits_{\overline{f}^i} (\mathfrak{D}_1, \ldots, \mathfrak{D}_{k_i}) \text{ where } \mathfrak{D}_c = \begin{cases} \mathfrak{A}_i^c & \text{if } \lambda(i,c) = \text{fin}, \\ \mathfrak{B}_k & \text{if } \lambda(i,c) = k\,. \end{cases}$$

A successive application of this operator to a tuple $\overline{\mathfrak{C}}^0$ of empty structures yields the infinite sequence $\left(\overline{\mathfrak{C}}^i\right)_{i \in \omega}$ defined by

$$\begin{aligned} \overline{\mathfrak{C}}^0 &:= (\varnothing, \ldots, \varnothing) \text{ and} \\ \overline{\mathfrak{C}}^{i+1} &:= \mathcal{D}(\overline{\mathfrak{C}}^i)\ . \end{aligned}$$

Observe that the operator $\overline{\mathfrak{C}} \mapsto \mathcal{D}(\overline{\mathfrak{C}})$ is monotone in the sense that there exists an embedding of $\mathfrak{C}_i$ into $\mathcal{D}_i(\overline{\mathfrak{C}})$ preserving the indices of elements. This is due to the fact that unary relations are only inherited from the finite structures contained in the equations and the way tuples spanning components are added to the relations. Hence, the notion of the union $\mathfrak{C}_i \cup \mathcal{D}_i(\overline{\mathfrak{C}}) = \mathcal{D}_i(\overline{\mathfrak{C}})$ is well defined. This implies that the iteration of the operator yields, in the limit, the inductive fixed point $\mathcal{D}^\omega(\overline{\varnothing}) = \bigcup_{i \in \omega} \overline{\mathfrak{C}}^i = \bigcup_{i \in \omega} \mathcal{D}^i(\overline{\varnothing})$, which will be referred to as $\mathcal{S}(\mathcal{D})$. We denote the i-th structure of the fixed point by $\mathcal{S}_i(\mathcal{D})$, and we call a structure $\mathfrak{A}$ *inductive* if and only if there exists a system of equations $\mathcal{D}$ such that $\mathfrak{A}$ is isomorphic to some $\mathcal{S}_i(\mathcal{D})$.

Let $\mathcal{S}(\mathcal{D}) = (\mathfrak{A}_1, \ldots, \mathfrak{A}_n)$. By definition, each $\mathfrak{A}_m$ is an indexed structure over $\Sigma = [\max(k_1, \ldots, k_n)]$ obtained as a (k_m-ary) disjoint sum $\bigoplus_{\bar{f}m}(\mathfrak{D}_c)_{c\in[k_m]}$ with additional tuples spanning components according to $\mathcal{D}$, and hence, for each $c = 1, \ldots, k_m$, the component $\mathfrak{A}_m[c]$ is either isomorphic to the finite structure $\mathfrak{A}^c_m \in \mathrm{Str}(\mathcal{D})$ if $\lambda(m, c) = \mathrm{fin}$ or to $\mathfrak{A}_{\lambda(m,c)}$ otherwise. For easier referencing, we will partition the sets of indices into $\mathrm{Fin}_i = \{c : \lambda(i, c) = \mathrm{fin}\}$, and $\Delta_i = \{c : \lambda(i, c) \neq \mathrm{fin}\}$. Furthermore, for an indexed element $(a, w) \in A_i$, the *depth* of (a, w) is defined as $\mathrm{dp}(a, w) = |w|$, and the depth of a set is the maximal depth of its elements, $\mathrm{dp}(S) = \max\{\mathrm{dp}(s) : s \in S\}$.

Example 5.5. The system of equations defining the infinite binary tree $\mathfrak{T}_2$ with prefix ordering and unary predicates S_0 and S_1 for the left and right successor is:

$$\mathcal{D}_1 = \begin{cases} \Lambda_1 = \big(\{\bullet\}, S_0 = \varnothing, S_1 = \varnothing, < = \varnothing\big) \;\oplus\; \Lambda_2 \;\oplus\; \Lambda_3 & \text{with } f_< \\ \Lambda_2 = \big(\{\bullet\}, S_0 = \{\bullet\}, S_1 = \varnothing, < = \varnothing\big) \;\oplus\; \Lambda_2 \;\oplus\; \Lambda_3 & \text{with } f_< \\ \Lambda_3 = \big(\{\bullet\}, S_0 = \varnothing, S_1 = \{\bullet\}, < = \varnothing\big) \;\oplus\; \Lambda_2 \;\oplus\; \Lambda_3 & \text{with } f_< \end{cases}$$

where

$$f_<(c_1, c_2) = \begin{cases} \top & \text{if } c_1 = 1 \text{ and } c_2 \in \{2, 3\} \\ \bot & \text{otherwise .} \end{cases}$$

Note that, by definition, the values of the functions are only relevant for tuples in which at least two arguments differ, and the unary predicates S_0 and S_1 are solely inherited from the right-hand side structures. The defined structure is depicted in Figure 5.1.

The second example is the system $\mathcal{D}_2$ defining an infinite list of lists with two order relations, S on the primary list and L on the branching lists, as depicted in Figure 5.2.

$$\mathcal{D}_2 = \begin{cases} \Lambda_1 = \Lambda_2 \oplus \Lambda_1 & \text{with } f_{1,L}, f_{1,S} \\ \Lambda_2 = \big(\{\bullet\}, L = \varnothing, S = \varnothing\big) \;\oplus\; \Lambda_2 & \text{with } f_{2,L}, f_{2,S} \end{cases}$$

where $f_{1,S}(1, 2) = \top$ and $f_{2,L}(1, 2) = \top$, and $f_{i,j}(c_1, c_2) = \bot$ otherwise.

Observe that in the tree a direct successor relation is definable in WMSO from the constructed prefix ordering. Further, the dotted edges in Figure 5.2

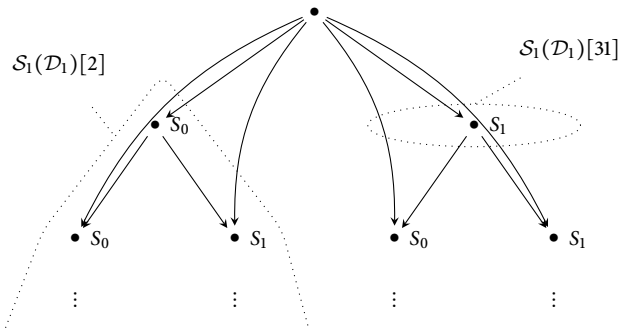

Figure 5.1. Inductive definition of the infinite binary tree $\mathfrak{T}_2 \cong \mathcal{S}_1(\mathcal{D}_1)$

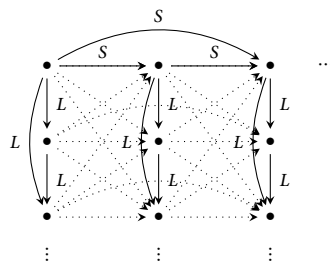

Figure 5.2. Inductive definition of the infinite list of lists

indicate unintentional S-edges introduced by the fixed-point construction. The intended edge relation S^* as indicated by the solid S-edges connecting the heads of the vertical lists only is definable by

$$S^*(x,y) = Sxy \wedge \neg\exists z(Lzx \vee Lzy) \quad .$$

5.2 Formulae with Restricted Variables

Intuitively, inductive structures are disjoint sums of other inductive structures with added relation tuples, and thus naturally decompose into *components*. When writing formulae over such structures, it is often convenient to restrict specific variables to specific components of the universe. Here, we introduce related notions and a procedure to *split* variables in a formula so as to convert a formula into one that only contains variables restricted to disjoint parts of the universe.

Formulae with restricted variables of k kinds are defined in the same way as WMSO formulae, but in addition to the standard first- and second-order variables $x_1, x_2, \ldots$ and $X_1, X_2, \ldots$ we allow the use of restricted variables $x_1^c, x_2^c, \ldots$ and $X_1^c, X_2^c, \ldots$ for $c = 1, \ldots, k$ distinguished from ordinary variables by superscripts. Intuitively, we think of formulae containing restricted variables as speaking about a structure where the universe A is partitioned into k pairwise disjoint sets $A^1, \ldots, A^k$ such that the variables x^c and X^c are understood as referring only to elements in the c-th component A^c of A.

We formally define the semantics of τ-formulae with restricted variables of k kinds by a translation into formulae over the expanded vocabulary $\hat{\tau} = \tau \cup \{P^1, \ldots, P^k\}$ where P^c are unary predicates not contained in τ. Given a formula φ with restricted variables, let $\hat{\varphi}$ be the formula obtained from φ by applying the following replacements

$$\begin{aligned}
x^c = y^d \quad &\mapsto \quad x \in P^c \wedge y \in P^d \wedge x = y \\
R(x_1^{c_1}, \ldots, x_r^{c_r}) \quad &\mapsto \quad \Big(\bigwedge_{j=1,\ldots,r} x_j \in P^{c_j}\Big) \wedge R(x_1, \ldots, x_r), \\
x^c \in Y^d \quad &\mapsto \quad x \in P^c \wedge x \in P^d \wedge x \in Y \\
\forall x^c \varphi(x^c) \quad &\mapsto \quad \forall z(z \in P^c \to \varphi(z)) \\
\exists x^c \varphi(x^c) \quad &\mapsto \quad \exists z(z \in P^c \wedge \varphi(z))
\end{aligned} \qquad [z \text{ is a fresh variable}]$$

$$\begin{array}{lcll} \forall X^c \varphi(X^c) & \mapsto & \forall Z(Z \subseteq P^c \to \varphi(Z)) & \\ \exists X^c \varphi(X^c) & \mapsto & \exists Z(Z \subseteq P^c \wedge \varphi(Z)) \ . & [Z \text{ is a fresh variable}] \end{array}$$

Note that the first three items are mainly important if the formula contains free variables since the range of quantified variables is already appropriately restricted by the guards. Given a τ-structure $\mathfrak{A}$ and a partition of its universe into k sets $A^1, \dots, A^k$, we refer to $\mathfrak{A}_{\langle A^1,\dots,A^k\rangle}$ as the *partitioned structure*, and denote the $\hat{\tau}$-expansion of $\mathfrak{A}$ in which each P^c is interpreted as the set A^c as usual by $(\mathfrak{A}, A^1, \dots, A^k)$. The semantics of φ evaluated on a partitioned τ-structure given an assignment β of the free first- and second-order variables is defined by $(\mathfrak{A}_{\langle A^1,\dots,A^k\rangle}, \beta) \vDash \varphi$ if and only if $(\mathfrak{A}, A^1, \dots, A^k, \beta) \vDash \hat{\varphi}$. Note that β is an assignment of the free original variables, and not of each restricted occurrence.

Concerning formulae with restricted variables we need an appropriate notion of quantifier rank. Classically, the quantifier rank corresponds to the deepest nesting of quantifiers in a formula. We extend this notion to a formula φ with restricted variables such that $\mathrm{qr}^i(\varphi)$ counts only the nesting of quantified variables that are restricted to the i-th component.

Definition 5.6. Let φ be a WMSO formula with restricted variables. Then the *quantifier rank over component c* is defined inductively by

- $\mathrm{qr}^c(\varphi) = 0$ if φ is an atomic formula,
- $\mathrm{qr}^c(\neg\varphi) = \mathrm{qr}^c(\varphi)$,
- $\mathrm{qr}^c(\varphi) = \max(\mathrm{qr}^c(\psi), \mathrm{qr}^c(\vartheta))$ if $\varphi = \psi \wedge \vartheta$ or $\varphi = \psi \wedge \vartheta$,
- $\mathrm{qr}^c(\exists X^d \varphi) = \mathrm{qr}^c(\exists x^d \varphi)$
  $= \mathrm{qr}^c(\forall X^d \varphi) = \mathrm{qr}^c(\forall x^d \varphi) = \begin{cases} \mathrm{qr}^c(\varphi) + 1 & \text{if } d = c \\ \mathrm{qr}^c(\varphi) & \text{otherwise.} \end{cases}$

The *restricted quantifier rank* $\mathrm{qr}^*(\varphi)$ is defined as the maximum over the quantifier ranks over the components: $\mathrm{qr}^*(\psi) = \max\{\mathrm{qr}^c(\psi) : 1 \le c \le k\}$.

5.2.1 Splitting Variables

Each formula of monadic second-order logic can be transformed into an equivalent formula in which all variables are restricted. The procedure `split` defined below computes, given a formula φ with variables $\{X_1, \dots, X_r, x_1, \dots, x_s\}$ and a positive integer k, a formula ψ with variables $\{X_1^c, \dots, X_r^c, x_1^c, \dots, x_s^c \ : \ 1 \le c \le k\}$ such that $\mathfrak{A}, \beta \vDash \varphi$ if and only if

Procedure 5.1. $\texttt{split}_k(\varphi)$

case φ *contains a free (unrestricted) FO variable* x
  return $\texttt{split}_k(\bigvee_{c=1,\dots,k} \varphi[x \leftarrow x^c])$
case φ *contains a free (unrestricted) MSO variable* X
  return $\texttt{split}_k(\varphi[X \leftarrow \bigcup_c X^c])$
case φ *is an atom* **return** φ
case $\varphi = \neg\psi$ **return** $\neg\texttt{split}_k(\psi)$
case $\varphi = \varphi_1 \vee \varphi_2$ **return** $\texttt{split}_k(\varphi_1) \vee \texttt{split}_k(\varphi_2)$
case $\varphi = \varphi_1 \wedge \varphi_2$ **return** $\texttt{split}_k(\varphi_1) \wedge \texttt{split}_k(\varphi_2)$
case $\varphi = \exists x\psi$ **return** $\bigvee_{c=1,\dots,k} \exists x^c \texttt{split}_k(\psi[x \leftarrow x^c])$
case $\varphi = \forall x\psi$ **return** $\bigwedge_{c=1,\dots,k} \forall x^c \texttt{split}_k(\psi[x \leftarrow x^c])$
case $\varphi = \exists X\psi$ **return** $\exists X^1 \dots X^k \texttt{split}_k(\psi[X \leftarrow \bigcup_c X^c])$
case $\varphi = \forall X\psi$ **return** $\forall X^1 \dots X^k \texttt{split}_k(\psi[X \leftarrow \bigcup_c X^c])$

$\mathfrak{A}_{\langle A^1,\dots,A^k\rangle}, \beta \vDash \psi$ for any partition $A^1, \dots, A^k$ of the universe of $\mathfrak{A}$ and any interpretation β of the free variables. In the notation used in the procedure `split`, we allow to substitute a sum, e.g. $X \cup Y$ for a second-order variable Z. This should be understood as replacing each atom $z \in Z$ by $z \in X \vee z \in Y$ (and $Z \leftarrow \varnothing$ means substituting an atom $z \in Z$ by $\bot$). To improve readability, we denote the input parameter k as an index.

Note that it is important that the replacement of the free variables in the procedure `split` is done first before splitting the rest of the formula.

Example 5.7.

$$\texttt{split}_3\big(\exists X \forall x (x \in X)\big) = \exists X^1 \exists X^2 \exists X^3 \big(\forall x^1 (\bigvee_{i=1}^{3} x^1 \in X^i) \wedge \forall x^2 (\bigvee_{i=1}^{3} x^2 \in X^i) \wedge \forall x^3 (\bigvee_{i=1}^{3} x^3 \in X^i)\big) \ .$$

Lemma 5.8 (Splitting Lemma)**.** For any positive integer k, any structure $\mathfrak{A}$, any partition $(A^1, \dots, A^k)$ of the universe of $\mathfrak{A}$ and any assignment β of the free variables occurring in φ,

(1) $\mathfrak{A}, \beta \vDash \varphi$ if and only if $\mathfrak{A}_{\langle A^1,\dots,A^k\rangle}, \beta \vDash \texttt{split}_k(\varphi)$, and
(2) $\mathrm{qr}^*(\texttt{split}_k(\varphi)) \leq \mathrm{qr}(\varphi)$.

Proof. Statement (2) immediately follows from the definition of quantifier rank and the definition of $\texttt{split}_k$. We show the first statement by an induction on the structure of formulae.
Atomic formulae:

- $\varphi = (x = y)$

$$
\begin{aligned}
&\mathfrak{A}, \beta \vDash x = y \iff \beta(x) = \beta(y) \\
\iff &\text{ex. } c, d \in [k] \text{ such that } \beta(x) \in A^c, \beta(y) \in A^d, \text{and } \beta(x) = \beta(y) \\
\iff &(\mathfrak{A}, A^1, \dots, A^k, \beta) \vDash \bigvee_{c=1}^{k} \bigvee_{d=1}^{k} \underbrace{x \in P^c \wedge y \in P^d \wedge x = y}_{\text{translation of } x^c = y^d} \\
\iff &(\mathfrak{A}_{\langle A^1, \dots, A^k \rangle}, \beta) \vDash \bigvee_{c=1}^{k} \bigvee_{d=1}^{k} (x^c = y^d) = \texttt{split}_k(\varphi)
\end{aligned}
$$

- $\varphi = R(x_1, \dots, x_r)$

$$
\begin{aligned}
&\mathfrak{A}, \beta \vDash R(x_1, \dots, x_r) \\
\iff &(\beta(x_1), \dots, \beta(x_r)) \in R^{\mathfrak{A}} \\
\iff &\text{ex. } c_1, \dots, c_r \text{ such that } \beta(x_1) \in A^{c_1}, \dots, \beta(x_r) \in A^{c_r}, \\
&\qquad\qquad \text{and } (\beta(x_1), \dots, \beta(x_r)) \in R^{\mathfrak{A}} \\
\iff &(\mathfrak{A}, A^1, \dots, A^k, \beta) \vDash \bigvee_{(c_1, \dots, c_r) \in [k]^r} \underbrace{\bigwedge_{j=1,\dots,r} x_j \in P^{c_j} \wedge R(x_1, \dots, x_r)}_{\text{translation of } R(x_1^{c_1}, \dots, x_r^{c_r})} \\
\iff &(\mathfrak{A}_{\langle A^1, \dots, A^k \rangle}, \beta) \vDash \bigvee_{(c_1, \dots, c_r) \in [k]^r} R(x_1^{c_1}, \dots, x_r^{c_r}) = \texttt{split}_k(\varphi)
\end{aligned}
$$

- $\varphi = x \in Y$

$$
\begin{aligned}
&\mathfrak{A}, \beta \vDash x \in Y \\
\iff &\text{ex. } c \in [k] \text{ such that } \beta(x) \in A^c, \text{and } \beta(x) \in \beta(Y) \\
\iff &\text{ex. } c \in [k] \text{ such that } \beta(x) \in A^c, \text{and } \beta(x) \in \bigcup_d (\beta(Y) \cap A^d) \\
\iff &(\mathfrak{A}, A^1, \dots, A^k, \beta) \vDash \bigvee_{c=1}^{k} \bigvee_{d=1}^{k} \underbrace{x \in P^c \wedge x \in P^d \wedge x \in Y}_{\text{translation of } x^c \in Y^d} \\
\iff &(\mathfrak{A}_{\langle A^1, \dots, A^k \rangle}, \beta) \vDash \bigvee_{c=1}^{k} \bigvee_{d=1}^{k} x^c \in Y^d = \texttt{split}_k(\varphi)
\end{aligned}
$$

Inductive step:

- If φ is a Boolean combination, the statement is obvious.
- $\varphi = \exists x \psi(x)$

$$
\begin{aligned}
& \mathfrak{A}, \beta \vDash \exists x \psi(x) \\
\iff & \text{ex. } a \in A \text{ such that } \mathfrak{A}, \beta[x \mapsto a] \vDash \psi(x) \\
\iff & \text{ex. } c \text{ and } a \in A^c \text{ such that } \mathfrak{A}, \beta[x \mapsto a] \vDash \psi(x) \\
\overset{\text{(IH)}}{\iff} & \text{ex. } c \text{ and } a \in A^c \text{ such that } \mathfrak{A}_{\langle A^1, \ldots, A^k \rangle}, \beta[x \mapsto a] \vDash \mathtt{split}_k \psi(x) \\
\iff & \mathfrak{A}_{\langle A^1, \ldots, A^k \rangle}, \beta \vDash \bigvee_{c=1}^{k} \exists x^c \mathtt{split}_k \psi(x^c)
\end{aligned}
$$

- $\varphi = \forall x \psi(x)$ is analogous.
- $\varphi = \exists X \psi(X)$

$$
\begin{aligned}
& \mathfrak{A}, \beta \vDash \exists X \psi(X) \\
\iff & \text{ex. } S \subseteq A \text{ such that } \mathfrak{A}, \beta[X \mapsto S] \vDash \psi(X) \\
\iff & \text{ex. } S_1 \subseteq A^1, \ldots, S_k \subseteq A^k \\
& \quad \text{such that } \mathfrak{A}, \beta[X_1 \mapsto S_1, \ldots, X_k \mapsto S_k] \vDash \psi[X \leftarrow \bigcup_{c=1}^{k} X_c] \\
\iff & \text{ex. } S_1 \subseteq A^1, \ldots, S_k \subseteq A^k \\
& \quad \text{such that } \mathfrak{A}_{\langle A^1, \ldots, A^k \rangle}, \beta[X_c \mapsto S_c] \vDash \mathtt{split}_k(\psi[X \leftarrow \bigcup_{c=1}^{k} X_c]) \\
\iff & \mathfrak{A}_{\langle A^1, \ldots, A^k \rangle}, \beta \vDash \exists X^1 \ldots X^k \mathtt{split}_k(\psi[X \leftarrow \bigcup_{c=1}^{k} X^c])
\end{aligned}
$$

- $\varphi = \forall X \psi(X)$ is analogous. Q.E.D.

5.3 Decomposing Formulae

Given a system of equations defining an inductive structure, we can decompose a WMSO formula into a Boolean combination of formulae to be checked on the constituent structures.

Definition 5.9. Let $\mathcal{D}$ be a system of n structure equations such that k_i structures appear on the right-hand side of the i-th equation. Let $\mathcal{S}(\mathcal{D}) =$

$(\mathfrak{A}_1, \ldots, \mathfrak{A}_n)$ and let φ be a WMSO formula with free variables $\mathrm{free}(\varphi) = \{X_1, \ldots, X_r, x_1, \ldots, x_s\}$. For each $m \in [n]$, a $\mathcal{D}_m$-*decomposition* of φ is a sequence of k-tuples ($k = k_m$) of formulae $(\psi_1^1, \ldots, \psi_1^k), \ldots, (\psi_l^1, \ldots, \psi_l^k)$ such that the free variables of each ψ_i^c are included in $\mathrm{free}(\varphi)$, $\mathrm{qr}(\psi_i^c) \leq \mathrm{qr}(\varphi)$, and

$$\mathfrak{A}_m, \beta \vDash \varphi \iff \text{for some } i \in [l] \text{ and each } c \in [k]\colon\ \mathfrak{A}_m[c], \hat{\beta}^c \vDash \hat{\psi}_i^c \ ,$$

where β is an assignment of the free first- and second-order variables, and $\hat{\beta}^c$ is a partial assignment defined as follows:

- $\hat{\beta}^c$ is undefined for those variables x such that $\beta(x) \notin A_m[c]$,
- $\hat{\beta}^c(x) = \beta(x)$ otherwise, and
- $\hat{\beta}^c(X) = \beta(X) \cap A_m[c]$.

Furthermore,

$$\hat{\psi}_i^c = \begin{cases} \bot & \text{if } \psi_i^c \text{ contains a free variable } x \text{ with } \beta(x) \notin A_m[c] \text{ and} \\ \psi_i^c & \text{otherwise .} \end{cases}$$

Modifying the formulae and the interpretation ensures that all free first-order variables in the formulae $\hat{\psi}_i^c$ are actually interpreted by $\hat{\beta}^c$ and captures the intuition that a first-order variable cannot be assigned values from different components at the same time.

If the formula φ does not contain free first-order variables, we do not need to modify the formulae ψ_i^c and only need to restrict the interpretation of the free second-order variables to the available universe.

The following theorem is the main result used to prove the correctness of our algorithm. Let us remark that it can be obtained from more general composition theorems of Shelah [She75], but those theorems do not yield a practical algorithm.

Theorem 5.10 (Decomposition Theorem)**.** For any WMSO formula φ (with arbitrary free variables), any system $\mathcal{D}$ with n structure equations, and any $m \in [n]$, there exists a $\mathcal{D}_m$-decomposition of φ, and the decomposition is computable.

The rest of this section is devoted to a proof of the Decomposition Theorem in the setting of inductive structures which will be used in the

following to define a model checking game. Towards this, we introduce a new normal form for WMSO formulae, called the *type normal form* (TNF), which is inspired by the classical Hintikka formulae. As we will see later, during the transformation of a split formula into TNF, subformulae speaking about the same component get grouped together such that, if we convert the finally obtained TNF into a DNF, we immediately obtain the desired decomposition which can be considered as a list of all possible combinations of types the components must have in order for the whole structure to satisfy the original formula. Note that the TNF is in a sense a converse of the prenex normal form since quantifiers are pushed as deep inside the formulae as possible.

5.3.1 Type Normal Form

For a set of formulae Φ, we denote by $\mathcal{B}^+(\Phi)$ all positive Boolean combinations of formulae from Φ, i.e. the set of formulae we obtain by closing Φ under finite disjunctions and conjunctions (but not negation).

Definition 5.11. We call a formula φ *normal* if it is of the following form:

$$\begin{aligned}\varphi = {} & R(\bar{x}) \mid \neg R(\bar{x}) \mid x = y \mid x \neq y \mid x \in X \mid x \notin X \\ & \mid \exists x \mathcal{B}^+(\Phi_x) \mid \forall x \mathcal{B}^+(\Phi_x) \mid \exists X \mathcal{B}^+(\Phi_X) \mid \forall X \mathcal{B}^+(\Phi_X)\end{aligned}$$

such that all formulae in Φ_x and Φ_X are normal, $x \in \mathrm{free}(\varphi)$ for all $\varphi \in \Phi_x$, and $X \in \mathrm{free}(\varphi)$ for all $\varphi \in \Phi_X$. A formula is in *type normal form*, TNF, if it is a positive Boolean combination of normal formulae.

Example 5.12. Consider $\varphi = \exists x\big(P(x) \wedge (Q(y) \vee R(x))\big)$. The formula is not in TNF since $Q(y)$ violates the constraint that the quantified variable x has to appear freely in every subformula. The TNF of φ is

$$\big(Q(y) \wedge \exists x P(x)\big) \vee \exists x\big(P(x) \wedge R(x)\big) \ .$$

We claim that for each formula φ there exists an equivalent formula ψ in TNF such that $\mathrm{qr}(\psi) \leq \mathrm{qr}(\varphi)$ (and $\mathrm{qr}^*(\psi) \leq \mathrm{qr}^*(\varphi)$ for formulae with restricted variables) and the set of atoms of ψ is a subset of the atoms of φ. Moreover, the procedure $\texttt{TNF}(\varphi)$ computes such a formula ψ given a formula φ in negation normal form. Note that it uses sub-procedures `DNF`

Procedure 5.2. $\mathtt{TNF}(\varphi)$

case φ *is a literal* **return** φ
case $\varphi = \varphi_1 \vee \varphi_2$ **return** $\mathtt{TNF}(\varphi_1) \vee \mathtt{TNF}(\varphi_2)$
case $\varphi = \varphi_1 \wedge \varphi_2$ **return** $\mathtt{TNF}(\varphi_1) \wedge \mathtt{TNF}(\varphi_2)$
case $\varphi = \exists x\psi$ *(or* $\exists X\psi$*) and* $\mathtt{DNF}(\mathtt{TNF}(\psi)) = \bigvee_i(\bigwedge_j \psi^i_j)$
  $J_i \leftarrow \{j \mid x \in \mathrm{free}(\psi^i_j)\}$
  return $\bigvee_i \left(\bigwedge_{j\notin J_i} \psi^i_j \wedge \exists x(\bigwedge_{j\in J_i} \psi^i_j)\right)$
case $\varphi = \forall x\psi$ *(or* $\forall X\psi$*) and* $\mathtt{CNF}(\mathtt{TNF}(\psi)) = \bigwedge_i(\bigvee_j \psi^i_j)$
  $J_i \leftarrow \{j \mid x \in \mathrm{free}(\psi^i_j)\}$
  return $\bigwedge_i \left(\bigvee_{j\notin J_i} \psi^i_j \vee \forall x(\bigvee_{j\in J_i} \psi^i_j)\right)$

and CNF which, given a Boolean combination of formulae, convert it to disjunctive or conjunctive normal form without introducing new atoms.

Theorem 5.13. The formula $\psi = \mathtt{TNF}(\varphi)$ is in TNF, equivalent to φ, and its atoms and free variables are included in those of φ and $\mathrm{qr}(\psi) \leq \mathrm{qr}(\varphi)$. If φ contains restricted variables, then $\mathrm{qr}^*(\psi) \leq \mathrm{qr}^*(\varphi)$.

Proof. We proceed inductively on the structure of φ. For literals all the claims are trivial since TNF is an identity. For Boolean combinations of formulae, the procedure TNF only calls itself recursively, thus all claims of the theorem follow inductively as well.

Consider the case when $\varphi = \exists x\psi$ and $\mathtt{DNF}(\mathtt{TNF}(\psi)) = \bigvee_i(\bigwedge_j \psi^i_j)$. We convert $\mathtt{TNF}(\psi)$ to disjunctive normal form in this case since the existential quantifier is distributive over disjunctions, and thus $\mathtt{TNF}(\varphi) \equiv \bigvee_i(\exists x \bigwedge_j(\psi^i_j))$. Since quantifiers are also distributive over formulae which do not contain the quantified variable, we get that the result, $\bigvee_i \left(\bigwedge_{j\in J_i} \psi^i_j \wedge \exists x(\bigwedge_{j\notin J_i} \psi^i_j)\right)$, is equivalent to $\exists x\mathtt{TNF}(\psi)$, and thus, by the induction hypothesis, also to φ. Since each formula ψ^i_j is normal by the induction hypothesis, in order to show that the result is in TNF, we only need to check that $\exists x(\bigwedge_{j\in J_i} \psi^i_j)$ is normal. Syntactically this is trivial, and the constraint on variables in the TNF is indeed satisfied by the choice of J_i. By the induction hypothesis and the assumption on the procedure DNF, the set of atoms does not increase, and due to the choice of J_i there are no new free variables

introduced. Furthermore, neither the quantifier rank nor the quantifier rank over any restricted variable increases. The case of universal quantification is analogous, but instead of the DNF, we consider the CNF of the inner formula. Q.E.D.

For proving the main lemma about decompositions, we need to exploit the following important but rather technical property of formulae in TNF.

Lemma 5.14. Let φ be a formula in TNF, and let $V_1, \ldots, V_n$ be a partition of the variables occurring in φ such that if two variables appear in the same atom in φ, these variables belong to the same V_i (we say that the partition is *compatible with the atoms*). Then φ is a Boolean combination of normal formulae $\varphi_1, \ldots, \varphi_m$ such that each φ_j contains only atoms with variables from one of the sets V_i.

Proof. By contradiction, assume that there exists a formula φ in TNF which does not satisfy the above condition. Consider the smallest such formula (with respect to its length). Then φ is normal since from a Boolean combination of normal formulae one could choose a single subformula with atoms from different sets.

Since the chosen partition is compatible with the atoms, the counterexample φ cannot be an atom, and thus it is of the form $\exists X \mathcal{B}^+(\Phi_X)$ or $\forall X \mathcal{B}^+(\Phi_X)$ (or of the same form for first-order quantification). Due to the minimality of φ, each formula $\psi_j \in \Phi_X$ satisfies the claim and contains atoms from only a single set V_{i_j}. But, by the constraint on TNF, we know that X occurs as a free variable in each formula in Φ_X, and thus X is contained in each of the sets V_{i_j}. Since the sets V_i are pairwise disjoint, all indices i_j must be the same. This contradicts the assumption that φ contains subformulae with variables from different sets V_i. Q.E.D.

5.3.2 Formula Decomposition Algorithm

Procedure 5.3 combines the procedures for splitting formulae and the TNF conversion to obtain an algorithm for actually computing a $\mathcal{D}_m$-decomposition. Note that the algorithm does not need the whole description of the system of equations, but it suffices to know the number k_m of components the m-th structure is composed of and the functions $(f_{m,i})_{i \in [k_m]}$ defining the relations between components.

Procedure 5.3. $\texttt{decomp}(\varphi, m)$

$\psi_m \leftarrow \texttt{split}_{k_m}(\varphi)$
$\vartheta_m \leftarrow$ replace in ψ_m
– $x^{c_1} \in X^{c_2}$ or $x^{c_1} = x^{c_2}$ with $\bot$ if $c_1 \neq c_2$
– $R_i(x_1^{c_1}, \ldots, x_{r_i}^{c_{r_i}})$ with $f_{m,i}(c_1, \ldots, c_{r_i})$ if not all c_l are equal
return $\texttt{DNF}(\texttt{TNF}(\vartheta_m))$ /* $= \bigvee_i \bigwedge_j \tau_{ij}$ */

Lemma 5.15 (Decomposition Lemma). Let φ be a formula with free variables $\{X_1, \ldots, X_r, x_1, \ldots, x_s\}$, and let $\mathcal{D}$ be a system of n structure equations

$$\mathcal{D} = \begin{cases} \Lambda_1 = \mathfrak{A}_1^1 \oplus \mathfrak{A}_1^2 \oplus \ldots \oplus \mathfrak{A}_1^{k_1} & \text{with } f_{1,1}, \ldots, f_{1,t} \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots & \\ \Lambda_n = \mathfrak{A}_n^1 \oplus \mathfrak{A}_n^2 \oplus \ldots \oplus \mathfrak{A}_n^{k_n} & \text{with } f_{n,1}, \ldots, f_{n,t} \end{cases}$$

with $\mathcal{S}(\mathcal{D}) = (\mathfrak{A}_1, \ldots, \mathfrak{A}_n)$. For each $m \in [n]$, $\texttt{decomp}(\varphi, m)$ computes a $\mathcal{D}_m$-decomposition of φ.

Proof. We first consider the case that φ does not contain any free first-order variables. By the Splitting Lemma and the definition of WMSO semantics we get that $\mathfrak{A}_m, \overline{P} \vDash \varphi \iff \mathfrak{A}_m, \overline{P}^c \vDash \psi_m$, where $P_i^c = P_i \cap \mathfrak{A}_m[c]$. Considering Step 2 of the algorithm, by the definition of the semantics of WMSO with restricted variables and the definition of $\mathcal{S}(\mathcal{D})$ we further get that

$$\mathfrak{A}_m, \overline{P}^c \vDash \psi_m \iff \mathfrak{A}_m, \overline{P}^c \vDash \vartheta_m \ .$$

After this simplification step, all variables occurring in the same atomic subformula in ϑ_m are restricted to the same component, and by Lemma 5.14, each subformula τ_{ij} in $\texttt{DNF}(\texttt{TNF}(\vartheta_m)) = \bigvee_i \bigwedge_j \tau_{ij}$ contains only atoms (and thus also quantifiers) with variables restricted to a single component. Let ψ_i^c be the conjunction of all τ_{ij} containing variables restricted to the component $c \in [k_l]$, or $\top$ if no such τ_{ij} occurs. Clearly $\texttt{TNF}(\vartheta_m)$ is equivalent to $\bigvee_i (\bigwedge_c \psi_i^c)$, and combining this with the previous equivalences we get that

$$\mathfrak{A}_m, \overline{P} \vDash \varphi \iff \mathfrak{A}_m, \overline{P}^c \vDash \bigvee_i \Big(\bigwedge_c \psi_i^c \Big) \ .$$

To show that ψ_i^c with restricted variables X^c, x^c replaced by the standard ones X, x is a $\mathcal{D}_m$-decomposition of φ, it only remains to prove that $\mathrm{qr}(\tau_{ij}) \leq \mathrm{qr}(\varphi)$ for all i, j. Observe that, by Lemma 5.8, we have $\mathrm{qr}^*(\psi_m) \leq \mathrm{qr}(\varphi)$. Replacing atoms does not change the quantifier rank, and Theorem 5.13 implies that $\mathrm{qr}^*(\mathtt{TNF}(\vartheta_m)) \leq \mathrm{qr}^*(\psi_m)$. But since each $\tau_{i,j}$ contains only quantification over variables from one component, we obtain that $\mathrm{qr}^*(\mathtt{TNF}(\vartheta_m)) = \max_{i,j} \mathrm{qr}(\tau_{i,j}) \leq \mathrm{qr}(\varphi)$.

Now consider the slightly more involved case that the formula contains free first-order variables. We show that the algorithm also yields a decomposition satisfying all required properties. In the following, for a formula φ with free first-order variables $\{x_1, \ldots, x_s\}$ and a tuple $(c_1, \ldots, c_s)$, we will use the notation $\varphi^{\bar{c}}$ to denote the formula $\varphi[x_1/x_1^{c_1}, \ldots, x_s/x_s^{c_s}]$. Furthermore, given an assignment β of the free first-order variables, we denote by $\bar{c}_\beta = (c_1, \ldots, c_s)$ the tuple of components such that $\beta(x_i) \in A^{c_i}$ for $i = 1, \ldots, s$.

The first step of the algorithm computes $\psi_m = \mathtt{split}_{k_m}(\varphi)$. Assume that φ contains the free variables $x_1, \ldots, x_s$. Then, after splitting all first-order variables, we obtain

$$\psi_m = \bigvee_{(c_1,\ldots,c_s)\in[k_m]^s} \underbrace{\mathtt{split}_{k_m}(\varphi^{\bar{c}})}_{\eta_m^{\bar{c}}},$$

i.e. ψ_m contains, for every possible combination of components, a disjunct in which each free variable is restricted to that specific component. Furthermore, by the Splitting Lemma,

$$\mathfrak{A}, \beta \vDash \varphi \iff \mathfrak{A}_{\langle A^1,\ldots,A^{k_m}\rangle}, \beta \vDash \mathtt{split}_{k_m}(\varphi) \ .$$

By the definition of the semantics of formulae with restricted variables, it is clear that a formula containing a variable x_j restricted to c_j is evaluated to $\bot$ under an assignment β with $\beta(x_j) \notin A^{c_j}$. Hence, for every assignment β, there is exactly one disjunct in ψ_m that does not trivially evaluate to $\bot$, which immediately implies that $\mathfrak{A}, \beta \vDash \varphi \iff \mathfrak{A}_{\langle A^1,\ldots,A^{k_m}\rangle}, \beta \vDash \mathtt{split}_{k_m}(\varphi^{\bar{c}_\beta})$.

Since the definition of the TNF does not impose constraints on occurrences of free variables, the transformation $\mathtt{TNF}(\varphi(\bar{x}))$ yields a formula with the same free variables in TNF. Furthermore, by definition of Proce-

dure 5.2, $\mathtt{TNF}(\psi_m) = \mathtt{TNF}(\bigvee_{\bar{c}} \eta_m^{\bar{c}}) = \bigvee_{\bar{c}} \mathtt{TNF}(\eta_m^{\bar{c}})$, and due to commutativity of disjunction also $\mathtt{DNF}(\mathtt{TNF}(\bigvee_{\bar{c}} \eta_m^{\bar{c}})) = \bigvee_{\bar{c}} \mathtt{DNF}(\mathtt{TNF}(\eta_m^{\bar{c}}))$.

As each disjunct represents a tuple of formulae in the sequence which we claim to be a $\mathcal{D}_m$-decomposition, this sequence can be seen as a concatenation of subsequences—one for each formula $\varphi^{\bar{c}}$.

Again, according to the definition of the restricted-variable semantics, $\mathfrak{A}, \beta \vDash \varphi$ if and only if $\mathfrak{A}_{\langle A^1,\dots,A^{k_m}\rangle}, \beta \vDash \varphi^{c_\beta}$. Let β be fixed, and assume that $\mathfrak{A}_m, \beta \vDash \varphi$. By the same reasoning as before, the subsequence obtained from the decomposition of $\varphi^{\bar{c}_\beta}$ contains a tuple $(\psi^1, \dots, \psi^{k_m})$ such that $\mathfrak{A}_m[c], \hat{\beta}^c \vDash \hat{\psi}^c$ for each $c = 1, \dots, k_m$. (Note that in this case, $\hat{\beta}^c(x_i) = \beta(x_i)$ and $\hat{\psi}^c = \psi^c$ for all $c \in [k_m]$.) Conversely, assume there is a tuple $(\psi^1, \dots, \psi^{k_m})$ such that $\mathfrak{A}_m[c], \hat{\beta}^c \vDash \hat{\psi}^c$ for each $c = 1, \dots, k_m$, then this tuple must originate from the decomposition of $\varphi^{\bar{c}_\beta}$. Otherwise, $\hat{\psi}^c = \bot$ for some c, contradicting the assumption. Hence, by the semantic equivalence of the TNF to the original formula and the Splitting Lemma, $\mathfrak{A}_m, \beta \vDash \varphi^{\bar{c}_\beta}$, and by the above observation also $\mathfrak{A}_m, \beta \vDash \varphi$. Q.E.D.

This eventually concludes the proof of Theorem 5.10; as we have seen, the Decomposition Lemma provides an effective solution to the problem of computing decompositions by means of Procedure 5.3. Generally analysing the size of the resulting decompositions is quite involved; the main source for massive inflation being quantifier alternations in connection with conjunctions and disjunctions necessitating DNF and CNF conversions during the computation of the TNF. Hence, implementations depend to a great extent on the performance of efficient conversion algorithms. Further, the algorithm still leaves much room for optimisations, which is also indicated in the following example.

Example 5.16. We consider the following system of equations defining the infinite binary tree with the prefix ordering, a root predicate R, and predicates S_L and S_R for left and right successors, respectively.

$$\mathcal{D} = \begin{cases} \Lambda_1 = \big(\{\bullet\}, R = \{\bullet\}, S_0 = \varnothing, S_1 = \varnothing, < = \varnothing\big) \oplus \Lambda_2 \oplus \Lambda_3\,, \quad f_< \\ \Lambda_2 = \big(\{\bullet\}, R = \varnothing, S_0 = \{\bullet\}, S_1 = \varnothing, < = \varnothing\big) \oplus \Lambda_2 \oplus \Lambda_3\,, \quad f_< \\ \Lambda_3 = \big(\{\bullet\}, R = \varnothing, S_0 = \varnothing, S_1 = \{\bullet\}, < = \varnothing\big) \oplus \Lambda_2 \oplus \Lambda_3\,, \quad f_< \end{cases}$$

where $f_<(c_1, c_2) = \top$ if and only if $c_1 = 1$ and $c_2 \in \{2, 3\}$.

We compute the $\mathcal{D}_1$-decomposition of the formula

$$\psi = \exists X \forall x \Big(x \in X \to S_L x \wedge \forall y (y \prec x \to (Ry \vee S_R y)) \Big)$$

on the binary tree defined by $\mathcal{D}$ following Procedure 5.2 and Procedure 5.3.

$$\texttt{split}_3\Big(\exists X \forall x \Big(x \in X \to S_L x \wedge \forall y (y \prec x \to (Ry \vee S_R y))\Big)\Big)$$

$$\begin{aligned}
= \exists X^1 \exists X^2 \exists X^3 \forall x^1 & \left(\bigvee_{i=1}^{3} x^1 \in X^i \to \begin{pmatrix} S_L x^1 \\ \wedge \forall y^1 (y^1 \prec x^1 \to (Ry^1 \vee S_R y^1)) \\ \wedge \forall y^2 (y^2 \prec x^1 \to (Ry^2 \vee S_R y^2)) \\ \wedge \forall y^3 (y^3 \prec x^1 \to (Ry^3 \vee S_R y^3)) \end{pmatrix} \right) \\
\wedge \forall x^2 & \left(\bigvee_{i=1}^{3} x^2 \in X^i \to \begin{pmatrix} S_L x^2 \\ \wedge \forall y^1 (y^1 \prec x^2 \to (Ry^1 \vee S_R y^1)) \\ \wedge \forall y^2 (y^2 \prec x^2 \to (Ry^2 \vee S_R y^2)) \\ \wedge \forall y^3 (y^3 \prec x^2 \to (Ry^3 \vee S_R y^3)) \end{pmatrix} \right) \\
\wedge \forall x^3 & \left(\bigvee_{i=1}^{3} x^3 \in X^i \to \begin{pmatrix} S_L x^3 \\ \wedge \forall y^1 (y^1 \prec x^3 \to (Ry^1 \vee S_R y^1)) \\ \wedge \forall y^2 (y^2 \prec x^3 \to (Ry^2 \vee S_R y^2)) \\ \wedge \forall y^3 (y^3 \prec x^3 \to (Ry^3 \vee S_R y^3)) \end{pmatrix} \right)
\end{aligned}$$

Since $x^j \in X^i \equiv \bot$ if $i \neq j$, and $y^i \prec x^j \equiv f_\prec(i,j)$ if $i \neq j$, the formula immediately simplifies to

$$\begin{aligned}
\exists X^1 \exists X^2 \exists X^3 \Bigg(& \underbrace{\forall x^1 \Big(x^1 \in X^1 \to S_L x^1 \wedge \forall y^1 (y^1 \prec x^1 \to (Ry^1 \vee S_R y^1)) \Big)}_{\eta_1} \Bigg) \\
& \underbrace{\wedge \forall x^2 \left(x^2 \in X^2 \to S_L x^2 \wedge \begin{pmatrix} \forall y^1 (Ry^1 \vee S_R y^1) \\ \wedge \forall y^2 (y^2 \prec x^2 \to (Ry^2 \vee S_R y^2)) \end{pmatrix} \right)}_{\eta_2} \\
& \underbrace{\wedge \forall x^3 \left(x^3 \in X^3 \to S_L x^3 \wedge \begin{pmatrix} \forall y^1 (Ry^1 \vee S_R y^1) \\ \wedge \forall y^3 (y^3 \prec x^3 \to (Ry^3 \vee S_R y^3)) \end{pmatrix} \right)}_{\eta_3} \Bigg)
\end{aligned}$$

Although η_1 is already in TNF, the procedure computes the CNF of the quantified subformula and pushes the quantifier inside yielding

$$\begin{aligned}\texttt{TNF}(\eta_1) = \forall x^1 (x^1 \notin X^1 \vee S_L x^1) \\ \wedge \forall x^1 (x^1 \notin X^1 \vee \forall y^1 (y^1 \prec x^1 \to (R y^1 \vee S_R y^1)))\ .\end{aligned}$$

Analogously, the TNF of η_2 is obtained by computing a CNF and pushing the universal quantifier inside:

$$\begin{aligned}\texttt{TNF}(\eta_2) = \forall x^2 (x^2 \notin X^2 \vee S_L x^2) \\ \wedge \underline{(\forall y^1 (R y^1 \vee S_R y^1) \vee \forall x^2 (x^2 \notin X^2))} \\ \wedge \forall x^2 (x^2 \notin X^2 \vee \forall y^2 (y^2 \prec x^2 \to (R y^2 \vee S_R y^2)))\ .\end{aligned}$$

The TNF of η_3 is identical up to the indices. Computing a DNF of $\text{TNF}(\eta_1) \wedge \text{TNF}(\eta_2) \wedge \text{TNF}(\eta_3)$ and distributing the existential second-order quantifiers, we finally obtain the TNF of the split formula as the following disjunctive normal form consisting of two clauses generated by the two non-trivial clauses in $\texttt{TNF}(\eta_2)$ and $\texttt{TNF}(\eta_3)$ (underlined above).

$$\left(\begin{array}{l} \forall y^1 (R y^1 \vee S_R y^1) \\ \wedge \exists X^1 \left(\begin{array}{l} \forall x^1 (x^1 \notin X^1 \vee S_L x^1) \\ \wedge \forall x^1 (x^1 \notin X^1 \vee \forall y^1 (y^1 \prec x^1 \to (R y^1 \vee S_R y^1))) \end{array}\right) \\ \wedge \exists X^2 \left(\begin{array}{l} \forall x^2 (x^2 \notin X^2 \vee S_L x^2) \\ \wedge \forall x^2 (x^2 \notin X^2 \vee \forall y^2 (y^2 \prec x^2 \to (R y^2 \vee S_R y^2))) \end{array}\right) \\ \wedge \exists X^3 \left(\begin{array}{l} \forall x^3 (x^3 \notin X^3 \vee S_L x^3) \\ \wedge \forall x^3 (x^3 \notin X^3 \vee \forall y^3 (y^3 \prec x^3 \to (R y^3 \vee S_R y^3))) \end{array}\right) \end{array}\right)$$

$$\vee \left(\begin{array}{l} \exists X^1 \left(\begin{array}{l} \forall x^1 (x^1 \notin X^1 \vee S_L x^1) \\ \wedge \forall x^1 (x^1 \notin X^1 \vee \forall y^1 (y^1 \prec x^1 \to (R y^1 \vee S_R y^1))) \end{array}\right) \\ \wedge \exists X^2 \left(\begin{array}{l} \forall x^2 (x^2 \notin X^2) \wedge \forall x^2 (x^2 \notin X^2 \vee S_L x^2) \\ \wedge \forall x^2 (x^2 \notin X^2 \vee \forall y^2 (y^2 \prec x^2 \to (R y^2 \vee S_R y^2))) \end{array}\right) \\ \wedge \exists X^3 \underbrace{\left(\begin{array}{l} \forall x^3 (x^3 \notin X^3) \wedge \forall x^3 (x^3 \notin X^3 \vee S_L x^3) \\ \wedge \forall x^3 (x^3 \notin X^3 \vee \forall y^3 (y^3 \prec x^3 \to (R y^3 \vee S_R y^3))) \end{array}\right)}_{\equiv \forall x^3 (x^3 \notin X^3)} \end{array}\right)\ .$$

Since the TNF is already in disjunctive normal form, it directly yields the following $\mathcal{D}_1$-decomposition $((\psi_1^1, \psi_1^2, \psi_1^3), (\psi_2^1, \psi_2^2, \psi_2^3))$ where

$$\begin{aligned}
\psi_1^1 &= \forall y(Ry \vee S_R y) \\
&\qquad \wedge \exists X \forall x(x \notin X \vee (S_L x \wedge \forall y(y \prec x \to (Ry \vee S_R y)))) \\
\psi_1^2 &= \exists X(\forall x(x \notin X \vee S_L x) \wedge \forall x(x \notin X \vee \forall y(y \prec x \to (Ry \vee S_R y)))) \\
\psi_1^3 &= \exists X(\forall x(x \notin X \vee S_L x) \wedge \forall x(x \notin X \vee \forall y(y \prec x \to (Ry \vee S_R y))))
\end{aligned}$$

$$\begin{aligned}
\psi_2^1 &= \exists X \forall x(x \in X \to (S_L x \wedge \forall y(y \prec x \to (Ry \vee S_R y)))) \\
\psi_2^2 &= \exists X \forall x(x \notin X) \qquad \text{(already simplified as indicated above)} \\
\psi_2^3 &= \exists X \forall x(x \notin X) \ .
\end{aligned}$$

5.4 Model Checking Game for WMSO

Game-theoretic formalisations of model checking problems provide useful insights into their complexity and manageability. For Boolean-valued logics, the problem can be reduced to the problem of determining the winner in a certain game between two players, called Verifier and Falsifier, in which Verifier tries to prove that the formula holds in a given structure, whereas Falsifier tries to prove that it does not.

For first-order model checking, reachability games suffice. The game graph is constructed from the formula in negation normal form and the structure such that positions consists of a formula together with an interpretation of its free variables. If the formula at a position is a disjunction (conjunction), Verifier (Falsifier) moves to either disjunct (conjunct) while leaving the variable assignment untouched. In a position with a formula of the form $\exists x \varphi(x)$ ($\forall x \varphi(x)$) and a variable assignment β, Verifier (Falsifier) chooses an element a from the structure and moves to the position determined by φ and the extended assignment $\beta[x \mapsto a]$. Positions with atomic or negated atomic formulae are the terminal positions, and Verifier wins at such a position if the formula holds under the assignment, otherwise Falsifier wins. The winning objective for each player is to reach such a terminal position where he or she wins. The size of the game graph is determined by the product of the size of the structure and the size of the formula, and hence it is finite if and only if the structure is finite. However, regardless of the

structure, all possible plays are finite since the length of a play is bounded by the length of the formula, and hence the game is determined, i.e. either Verifier or Falsifier has a winning strategy. Furthermore one can show that, as intended, Verifier has a winning strategy if and only if the formula holds in the given structure.

Since finite two-player reachability games can be regarded as alternating algorithms, we can directly conclude that the model checking problem for FO on finite structures is in ALogspace or Ptime. For a more detailed discussion see, for example, [GKL$^+$07].

The FO model checking game can be extended to capture the semantics of WMSO formulae by introducing new moves for formulae of the form $\exists X\varphi(X)$ ($\forall X\varphi(X)$) in which Verifier (Falsifier) may choose a finite subset of elements. But still, the arena is infinite if the structure is infinite.

We will present a game for model checking WMSO on inductive structures whose game graph is finite as it only depends on the size of the decompositions and the size of the system of equations defining the structure rather than on the size of the structure itself. Similarly to the standard FO or WMSO model checking games, we obtain, in essence, a reachability game; however, since the game graph is finite but the structures are in general infinite, we have to allow for arbitrarily long plays, and thus the game graph cannot be acyclic. This creates the possibility of playing infinitely long and the need to declare the winner of such an infinite play. We resolve this by introducing two priorities and considering a parity winning condition making it, as we will see, in every position of the game unfavourable for one of the players to pursue an infinite play.

While the moves for disjunctions and conjunctions can be handled as in the FO model checking game, quantified formulae are more challenging and cannot be captured by a single move but rather by a more complex subgame. We will first describe the subgames arising for quantified formulae, show how these subgames are linked to obtain a game for the whole formula, and prove the soundness and completeness of the game-based approach by induction.

The subgames we describe have exactly one starting position and distinguished exits which are terminal positions of the subgame in the sense that there are no outgoing edges leading back into that subgame. Subgames are linked to one another by gluing the exits to (identical) initial positions

of other appropriate subgames. Thus, exits are not terminal positions of the whole game anymore, but whenever the play passes through the exit of a subgame, it can never return to that subgame afterwards.

We will first describe the individual subgames for the respective types of formulae and prove their correctness in isolation. Reasoning that the whole game constructed by linking together the relevant subgames is the right model checking game then only requires an induction using those partial correctness results.

Note that the games are always intended to be constructed such that, for every position referencing a formula and a structure in the game graph, Verifier has a winning strategy if and only if the structure is a model of the formula under the interpretation of the free variables.

5.4.1 Subgame for the Existential Set Quantifier

Intuitively, to prove that an existentially quantified formula holds, Verifier must provide an appropriate *finite* set of elements. To do so, she unravels the structure according to its defining equations, chooses a subset of elements from the finite-structure components of the equations and provides the right tuple of formulae from the decomposition whose truth in the components witness the truth of the initial formula on the whole structure. This process of unravelling may have to be repeated for a finite number of steps until she has completed choosing the set of elements satisfying the formula.

Definition 5.17. Let $\exists X \varphi(X)$ be a WMSO sentence, i.e. X is the only free variable in φ. Let $\Phi = \{\psi : \mathrm{qr}(\psi) \leq \mathrm{qr}(\varphi), \mathrm{free}(\psi) \subseteq \{X\}\}$, and let $\mathcal{D}$ be a system of n structure equations. The subgame $\mathcal{G}_\exists(\varphi, m)$ is defined as follows.

- Positions of Verifier:
  $\{\langle\!\langle \psi, \Box, i \rangle\!\rangle : \psi \in \Phi, i \in [n]\}$.
- Positions of Falsifier:
  $\{\langle\!\langle (\psi^1, \ldots, \psi^{k_i}), S, i \rangle\!\rangle : \psi^c \in \Phi,\ S \subseteq \bigcup_{c \in \mathrm{Fin}_i} \mathfrak{A}_i^c,\ i \in [n]\}$.
- Initial position: $\langle\!\langle \varphi, \Box, m \rangle\!\rangle$.
- Exits:
  - $\{\langle\!\langle \mathfrak{A}, \psi, S, i \rangle\!\rangle : \mathfrak{A} \in \mathrm{Str}(\mathcal{D}),\ \psi \in \Phi,\ S \subseteq \bigcup_{c \in \mathrm{Fin}_i} \mathfrak{A}_i^c,\ i \in [n]\}$, and
  - $\{\langle\!\langle \varphi[X \leftarrow \varnothing], \Box, i \rangle\!\rangle : \varphi \in \Phi,\ i \in [n]\}$.

- Moves:
  - $\langle\!\langle \psi, \square, i \rangle\!\rangle \xrightarrow{\mathrm{V}} \langle\!\langle \psi[X \leftarrow \varnothing], \square, i \rangle\!\rangle$,
  - $\langle\!\langle \psi, \square, i \rangle\!\rangle \xrightarrow{\mathrm{V}} \langle\!\langle (\psi^1, \dots, \psi^{k_i}), S, i \rangle\!\rangle$, for each tuple $(\psi^1, \dots, \psi^{k_i})$ in the $\mathcal{D}_i$-decomposition of ψ, and each $S \subseteq \bigcup_{c \in \mathrm{Fin}_i} \mathfrak{A}_i^c$, and
  - $\langle\!\langle (\psi^1, \dots, \psi^{k_i}), S, i \rangle\!\rangle \xrightarrow{\mathrm{F}} \begin{cases} \langle\!\langle \mathfrak{A}_i^c, \psi^c, S, i \rangle\!\rangle & \text{if } \lambda(i,c) = \mathrm{fin} \\ \langle\!\langle \psi^c, \square, \ell \rangle\!\rangle & \text{if } \lambda(i,c) = \ell. \end{cases}$
- $F \coloneqq \varnothing$.

The positions should be understood as follows. In a position $\langle\!\langle \psi, \square, i \rangle\!\rangle$, the formula ψ is checked on $\mathcal{S}_i(\mathcal{D})$, i.e. on the i-th structure of the fixed point. The blank symbol ($\square$) serves as a placeholder denoting that a subset still has to be chosen for the only free set variable occurring in the formula. In a position $\langle\!\langle (\psi^1, \dots, \psi^{k_i}), S, i \rangle\!\rangle$, a set of elements S has been chosen, and each formula ψ^c must be checked in the c-th component of $\mathcal{S}_i(\mathcal{D})$. Finally, in positions $\langle\!\langle \mathfrak{A}, \psi, S, i \rangle\!\rangle$, the formula ψ is checked on the finite structure $\mathfrak{A}$ where X is interpreted by S.

Since the quantifier rank of the formulae in the decomposition tuples is bounded by the quantifier rank of φ and there are only finitely many non-equivalent formulae with fixed quantifier rank, Φ is finite. Furthermore, the size of the sets chosen by Verifier is bounded by the size of the structures in $\mathcal{D}$, and hence the arena of the subgame $\mathcal{G}_\exists(\varphi, m)$ is finite.

To reason about the correctness of the presented subgame, consider the game $\hat{\mathcal{G}}_\exists(\varphi, m)$ obtained from $\mathcal{G}_\exists(\varphi, m)$ by adding a self loop to each exit and re-defining

$$F \coloneqq \{\langle\!\langle \mathfrak{A}_i^c, \psi^c, S, i \rangle\!\rangle : \mathfrak{A}_i^c \vDash \psi^c(S)\} \cup \{\langle\!\langle \varphi[X \leftarrow \varnothing], \square, i \rangle\!\rangle : \mathcal{S}_i(\mathcal{D}) \vDash \varphi[X \leftarrow \varnothing]\} \ .$$

Lemma 5.18. Verifier wins the game $\hat{\mathcal{G}}_\exists(\varphi, m)$ if and only if $\mathcal{S}_m(\mathcal{D}) \vDash \exists X \varphi(X)$.

Proof. We prove that there is a direct correspondence between winning strategies for Verifier and finite sets satisfying formulae. Note that Verifier wins the Büchi game if she has a strategy to reach an exit in F in finitely many steps. (An infinite play that does not loop through an exit in F either loops in a bad exit or loops on the inner nodes which are all not contained in F.)

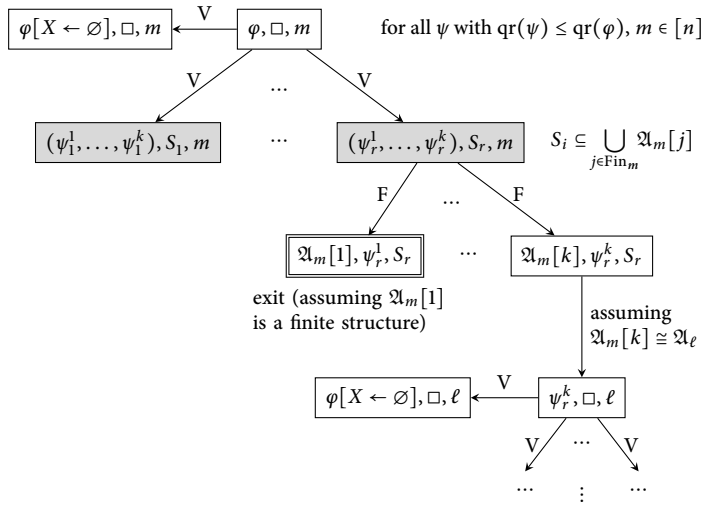

Figure 5.3. Subgame $\mathcal{G}_\exists(\varphi, m)$ for checking whether $\mathfrak{A}_m \vDash \exists X \varphi(X)$

($\Leftarrow$) Let $(\mathfrak{A}_1, \ldots, \mathfrak{A}_n) = \mathcal{S}(\mathcal{D})$ and assume that $\mathfrak{A}_m \vDash \exists X \varphi(X)$. Let S be a finite set such that $\mathfrak{A}_m, S \vDash \varphi(X)$. By the Decomposition Theorem there exists a $\mathcal{D}_m$-decomposition $(\psi_1^1, \ldots, \psi_1^{k_m}), \ldots, (\psi_r^1, \ldots, \psi_r^{k_m})$ of φ and an index $\ell \in [r]$ such that

$$(\mathfrak{A}_m[c], S \cap \mathfrak{A}_m[c]) \vDash \psi_\ell^c \text{ for all } c \in [k_m] \quad . \tag{$*$}$$

We prove the existence of a winning strategy for Verifier by induction on the depth of S.

Let $\mathrm{dp}(S) = 1$, i.e. $S \subseteq \bigcup_{c \in \mathrm{Fin}_m} \mathfrak{A}_m[c]$. Since $\mathrm{dp}(S) = 1$, all elements in S are from the finite components of $\mathfrak{A}_m$, i.e. $S \cap \bigcup_{c \in \Delta_m} \mathfrak{A}_m[c] = \varnothing$, and

$$\mathfrak{A}_{\lambda(m,c)}, \varnothing \vDash \psi_\ell^c \text{ for all } c \in \Delta_m \quad . \tag{$**$}$$

Hence, Verifier wins by moving to $\langle\!\langle (\psi_1^\ell, \ldots, \psi_{k_m}^\ell), S, m \rangle\!\rangle$. Falsifier cannot win by moving to a position $\langle\!\langle \mathfrak{A}_m[c], \psi_\ell^c, S, m \rangle\!\rangle$, for any $c \in \mathrm{Fin}_m$, since these positions are in F due to $(*)$, and from any position $\langle\!\langle \psi_\ell^c, \square, \lambda(m, c) \rangle\!\rangle$ with $c \in \Delta_m$, Verifier can move to $\langle\!\langle \psi_\ell^c[X \leftarrow \varnothing], \square, \lambda(m, c) \rangle\!\rangle$, which is in F due to $(**)$, and win.

Let $\mathrm{dp}(S) > 1$, and let $S_0 = S \cap \bigcup_{c \in \mathrm{Fin}_m} \mathfrak{A}_m[c]$. We show that Verifier has a winning strategy from $\langle\!\langle (\psi_\ell^1, \dots, \psi_\ell^{k_m}), S_0, m \rangle\!\rangle$. If Falsifier chooses to move to $\langle\!\langle \mathfrak{A}_m[c], \psi_\ell^c, S_0, m \rangle\!\rangle$ for some $c \in \mathrm{Fin}_m$, then Verifier wins because $(\mathfrak{A}_m[c], S \cap \mathfrak{A}_m[c]) \vDash \psi_\ell^c$ implies that $\langle\!\langle \mathfrak{A}_m[c], \psi_\ell^c, S_0, m \rangle\!\rangle$ is in F. If Falsifier chooses to move to $\langle\!\langle \psi_\ell^c, S_0, m \rangle\!\rangle$ for some $c \in \Delta_m$, then $\mathrm{dp}\big(\pi_c((S \setminus S_0) \cap \mathfrak{A}_m[c])\big) < \mathrm{dp}(S)$ (where $\pi_c : (s, cw) \mapsto (s, w)$), i.e. the depth of the remaining elements decreases upon descending into the c-th component. Since $(\mathfrak{A}_m[c], S \cap \mathfrak{A}_m[c]) \vDash \psi_\ell^c$, Verifier wins from position $\langle\!\langle \psi_\ell^c, \Box, \lambda(m, c) \rangle\!\rangle$ for each $c \in \Delta_m$ by the induction hypothesis.

($\Rightarrow$) Assume that Verifier has a strategy σ to win the game from the initial position $\langle\!\langle \varphi(X), \Box, m \rangle\!\rangle$. Since all plays won by Verifier reach an exit after finitely many steps, unravelling the game graph, and pruning all branches that are not reachable by plays consistent with Verifier's winning strategy or that arise from unravelling the self loops on the exits, we obtain a finite tree $\mathcal{T}_\sigma(\varphi, m)$ representing all possible plays of Falsifier against Verifier's winning strategy σ. The leaves of this tree are positions of the form $\langle\!\langle \mathfrak{A}_i^c, \psi^c, S, i \rangle\!\rangle$ or $\langle\!\langle \psi[X \leftarrow \varnothing], \Box, i \rangle\!\rangle$. We label the edges of the tree as follows: Edges representing Verifier's moves are labelled with ε, and edges representing Falsifier's moves are labelled with letters from $\{1, \dots, k_i\}$ corresponding to which part of the tuple Falsifier chooses, i.e. $\langle\!\langle (\psi^1, \dots, \psi^{k_i}), S, i \rangle\!\rangle \xrightarrow{j} \langle\!\langle \mathfrak{A}_i[c], \psi^c, S, i \rangle\!\rangle$ or $\langle\!\langle (\psi^1, \dots, \psi^{k_i}), S, i \rangle\!\rangle \xrightarrow{j} \langle\!\langle \psi^c, \Box, \lambda(i, c) \rangle\!\rangle$.

For each of Verifier's positions $p = \langle\!\langle \psi, i \rangle\!\rangle$ in the tree, we define the set $S(p)$ to be the unique set satisfying

$$S(p) \cap \mathfrak{A}_i[w] = S' \iff$$
$$\text{a leaf } \langle\!\langle \mathfrak{A}_w, \cdot, S', \cdot \rangle\!\rangle \text{ is reachable from } p \text{ via labels } w$$

(note that the structure $\mathfrak{A}_w$ in the leaf, being one of the finite structures in $\mathcal{D}$, is isomorphic to $\mathfrak{A}_i[w]$). Intuitively, this set is obtained by combining all structures in reachable leaves after appropriately indexing their elements by the path w leading to them. We prove by induction on the *height* of positions in the tree that $\big(\mathfrak{A}_i, S(\langle\!\langle \varphi, \Box, i \rangle\!\rangle)\big) \vDash \varphi$ holds for each position $\langle\!\langle \varphi, \Box, i \rangle\!\rangle$.

Let $\mathrm{h}(\langle\!\langle \varphi, \Box, i \rangle\!\rangle) = 0$. Then the only successor position is the leaf $\langle\!\langle \varphi[X \leftarrow \varnothing], \Box, i \rangle\!\rangle$. Therefore $S(\langle\!\langle \varphi, \Box, i \rangle\!\rangle) = \varnothing$, and since Verifier plays a winning strategy, the exit must be in F. Hence, by definition of the game, $\mathfrak{A}_i \vDash \varphi(\varnothing)$.

Let $\mathrm{h}(\langle\!\langle \varphi, \Box, i \rangle\!\rangle) > 0$. Then the only child $\langle\!\langle (\psi^1, \dots, \psi^{k_i}), S, i \rangle\!\rangle$ has successors $\langle\!\langle \mathfrak{A}_i[c], \psi^c, S', i \rangle\!\rangle$, which are leaves, and $\langle\!\langle \psi^c, \Box, \lambda(i,c) \rangle\!\rangle$ with $\mathrm{h}(\langle\!\langle \psi^c, \Box, \lambda(i,c) \rangle\!\rangle) < \mathrm{h}(\langle\!\langle \varphi, \Box, i \rangle\!\rangle)$. By the induction hypothesis, $\mathfrak{A}_{\lambda(i,c)} \vDash \psi^c\big(S(\langle\!\langle \psi_j, \Box, \lambda(i,j) \rangle\!\rangle)\big)$ for all $c \in \Delta_i$, and since we assume that Verifier plays a winning strategy, $\mathfrak{A}_i[c] \vDash \psi^c(S')$ for all $c \in \mathrm{Fin}_i$. Due to Theorem 5.10, we conclude that $\mathfrak{A}_i, S(\langle\!\langle \varphi, \Box, i \rangle\!\rangle) \vDash \varphi$. Considering the initial position $\langle\!\langle \varphi, \Box, m \rangle\!\rangle$, we obtain $\mathfrak{A}_m, S(\langle\!\langle \varphi, \Box, m \rangle\!\rangle) \vDash \varphi$, and hence $\mathfrak{A}_m \vDash \exists X \varphi(X)$. Q.E.D.

5.4.2 Subgame for the Existential First-Order Quantifier

The subgame we obtain for an existentially quantified formula $\exists x \varphi(x)$ is played as follows. Starting from the position in which the formula $\varphi(x)$ is evaluated on the m-th structure of the fixed point $\mathfrak{A}_m$, Verifier announces a component c in which the element satisfying $\varphi(x)$ can be found and chooses a tuple of formulae from the $\mathcal{D}_m$-decomposition of $\varphi(x)$. Then it is Falsifier's turn to challenge Verifier's claim in one of the components of $\mathfrak{A}_m$. If he challenges her in the c-th component and this component is finite, Verifier has to provide a witness from that component and exits the subgame. If the component is infinite, the game continues in that component, i.e. Verifier again chooses a component and a decomposition, and so on. If Falsifier challenges Verifier in a component different from the c-th one, the subgame is left towards an exit for model checking the respective formula which does not contain x as a free variable anymore.

Definition 5.19. Given a system of n structure equations $\mathcal{D}$ and a sentence $\exists x \varphi(x)$, let $\Phi = \{\psi : \mathrm{qr}(\psi) \leq \mathrm{qr}(\varphi), \mathrm{free}(\psi) \subseteq \{x\}\}$. The subgame $\mathcal{G}_{\exists_1}(\varphi, m)$ is defined as follows.

- Positions of Verifier:
  - $\{\langle\!\langle \psi, \Box, i \rangle\!\rangle : \psi \in \Phi,\ i \in [n]\}$, and
  - $\{\langle\!\langle \mathfrak{A}, \psi, c, \ell \rangle\!\rangle : \mathfrak{A} \in \mathrm{Str}(\mathcal{D}),\ \psi \in \Phi,\ c \in [k_i],\ c \neq \top,\ \ell \in [n]\}$.
- Positions of Falsifier:
  $\{\langle\!\langle (\psi^1, \dots, \psi^{k_i}), c, i \rangle\!\rangle : \psi^c \in \Phi,\ c \in [k_i],\ i \in [n]\}$.
- Initial position: $\langle\!\langle \varphi, \Box, m \rangle\!\rangle$.
- Exits:
  - $\{\langle\!\langle (\mathfrak{A}, \beta), \psi, \blacksquare, i \rangle\!\rangle : \mathfrak{A} \in \mathrm{Str}(\mathcal{D}),\ \beta : \{x\} \to A,\ \psi \in \Phi,\ i \in [n]\}$,
  - $\{\langle\!\langle \psi, \blacksquare, i \rangle\!\rangle : \psi \in \Phi,\ i \in [n]\}$.

- Moves:
  - $\langle\!\langle \psi, \square, i \rangle\!\rangle \xrightarrow{\mathrm{V}} \langle\!\langle (\psi^1, \dots, \psi^{k_i}), c, i \rangle\!\rangle$, for each tuple $(\psi^1, \dots, \psi^{k_i})$ in the $\mathcal{D}_i$-decomposition of ψ and each $c \in [k_i]$,
  - $\langle\!\langle \mathfrak{A}, \psi, \square, i \rangle\!\rangle \xrightarrow{\mathrm{V}} \langle\!\langle (\mathfrak{A}, x \mapsto a), \psi, \blacksquare, i \rangle\!\rangle$ for some $a \in \mathfrak{A}$
  - $\langle\!\langle (\psi_1, \dots, \psi_{k_i}), c, i \rangle\!\rangle \xrightarrow{\mathrm{F}} \begin{cases} \langle\!\langle \mathfrak{A}_i^c, \psi^c, \square, i \rangle\!\rangle & \text{if } \lambda(i,c) = \mathrm{fin}, \\ \langle\!\langle \mathfrak{A}_i^d, \psi^d, \blacksquare, i \rangle\!\rangle & \text{if } \lambda(i,d) = \mathrm{fin}, c \neq d, \\ \langle\!\langle \psi^c, \square, \ell \rangle\!\rangle & \text{if } \lambda(i,c) = \ell, \\ \langle\!\langle \psi^d, \blacksquare, \ell \rangle\!\rangle & \text{if } \lambda(i,d) = \ell, c \neq d. \end{cases}$
- $F \coloneqq \varnothing$.

The intuitive meaning of the positions is the same as in the existential set quantifier game, but we use an additional distinguished placeholder $\blacksquare$ denoting that the element variable is interpreted with an element from a different component such that it does not have to be chosen in the following.

As in the set quantifier game, the quantifier rank of the formulae in the decomposition tuples is bounded by the quantifier rank of φ and there are only finitely many non-equivalent formulae with fixed quantifier rank such that Φ is finite. Furthermore, there are only finitely many possible elements for Verifier to choose, and hence the arena of $\mathcal{G}_{\exists_1}(\varphi, m)$ is finite.

Again, to reason about the correctness of the subgame, we transform the game $\mathcal{G}_{\exists_1}(\varphi, m)$ into the game $\hat{\mathcal{G}}_{\exists_1}(\varphi, m)$ by adding a self loop to each exit and re-defining

$$F \coloneqq \{\langle\!\langle (\mathfrak{A}_i^c, \beta), \psi^c, \blacksquare, i \rangle\!\rangle : \mathfrak{A}_i^c, \beta \vDash \psi^c\} \cup \{\langle\!\langle \psi, \blacksquare, i \rangle\!\rangle : \mathfrak{A}_i \vDash \psi\} \quad .$$

Lemma 5.20. Verifier wins the game $\mathcal{G}_{\exists_1}(\varphi, m)$ if and only if $\mathcal{S}_m(\mathcal{D}) \vDash \exists x \varphi(x)$.

Proof. Again, we prove the correspondence of elements witnessing the truth of the formula and winning strategies for Verifier. Note that Verifier wins the Büchi game if and only if she has a strategy to reach an exit in F in a finite number of steps.

($\Leftarrow$) Let $(\mathfrak{A}_1, \dots, \mathfrak{A}_n) = \mathcal{S}(\mathcal{D})$ and assume that $\mathfrak{A}_m \vDash \exists x \varphi(x)$. We can assume that $\mathfrak{A}_m$ is an indexed structure, and we let a be the indexed element such that $\mathfrak{A}_m \vDash \varphi(a)$. By the Decomposition Theorem, there exists a $\mathcal{D}_m$-decomposition $(\psi_1^1, \dots, \psi_1^{k_m}), \dots, (\psi_r^1, \dots, \psi_r^{k_m})$ of φ and a number $\ell \in [r]$ such that, for $\beta : x \mapsto a$, we have $\mathfrak{A}_m[c], \hat{\beta}^c \vDash \hat{\psi}_\ell^c$ for all $c \in [k_m]$.

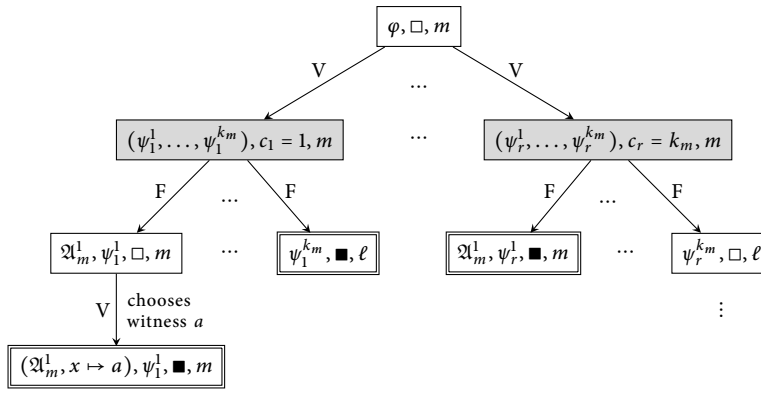

Figure 5.4. Subgame $\mathcal{G}_{\exists_1}(\varphi, m)$ for checking whether $\mathfrak{A}_m \vDash \exists x \varphi(x)$. On the first level, Verifier chooses a composition tuple and a component ($c_i \in [k_m]$) containing the witness for x, and on the second level, Falsifier chooses in which component to challenge Verifier's choice.

We prove the existence of a winning strategy for Verifier by induction on the depth of $a = (a_0, w)$. Let $\mathrm{dp}(a) = 1$, then $a = (a_0, c)$ is an element from the finite component $\mathfrak{A}_m[c]$. Since the witness a is chosen from component c, Verifier wins by moving to $\langle\!\langle (\psi_\ell^1, \ldots, \psi_\ell^{k_m}), c, m \rangle\!\rangle$. If Falsifier challenges her in component c, she wins by moving to $\langle\!\langle (\mathfrak{A}_m^c, x \mapsto a), \psi_\ell^c, \blacksquare, m \rangle\!\rangle \in F$.

Let $\mathrm{dp}(a) > 1$, i.e. $a = (a_0, cw')$ with $w' \neq \varepsilon$. Since a is to be found in the c-th component (which cannot be a finite one in this case), ψ_ℓ^c is the only formula of the tuple in which x occurs freely (otherwise, $\hat{\psi}_\ell^c = \bot$), and hence, by induction hypothesis, Verifier wins by moving to $\langle\!\langle (\psi_\ell^1, \ldots, \psi_\ell^{k_m}), c, m \rangle\!\rangle$. Choosing a component different from c eventually leads to an exit in F which is losing for Falsifier, so the best he can do is to challenge Verifier in the c-th component.

($\Rightarrow$) Assume that Verifier has a strategy σ to win the game from the initial position $\langle\!\langle \varphi(x), \square, m \rangle\!\rangle$. Since all plays won by Verifier lead to an exit in F after finitely many steps, unravelling the game graph to the tree $\mathcal{T}_\sigma(\varphi, m)$ and pruning branches that are not reachable by Verifier's strategy σ, or that start with a move by Falsifier that directly leads to an exit that is losing for him, we obtain a finite path (since Falsifier has no choice but to move to Verifier's chosen component if he does not want to lose immediately). Concatenating Verifier's choices of components along this path

to an index word w, we obtain, together with the final non-trivial move to the exit $\langle\!\langle (\mathfrak{A}_i^c, x \mapsto a), \varphi, \blacksquare, i \rangle\!\rangle$, an indexed element $a = (a_0, cw)$. Analogously to the proof of Lemma 5.18, by a straightforward induction using the Decomposition Theorem, we obtain that $\mathfrak{A}_m \vDash \varphi(a)$. Q.E.D.

5.4.3 The Constant Subgames

Besides the subgames for the quantifiers we need subgames for the constant formulae $\top$ and $\bot$ which are particularly simple. Both $\mathcal{G}(\top, m)$ and $\mathcal{G}(\bot, m)$ consist of only one (the initial) position p_0 with a self loop and have no exits. In $\mathcal{G}(\top, m)$, we let $F \coloneqq \{p_0\}$, and in $\mathcal{G}(\bot, m)$, we let $F \coloneqq \varnothing$. It is straightforward to see that, as intended, Verifier wins $\mathcal{G}(\top, m)$ and Falsifier wins $\mathcal{G}(\bot, m)$.

5.4.4 Finite-Structure Subgames

To check formulae, possibly with an assignment of the free variables, in a finite structure given by the equations, we use the common model checking game for WMSO as sketched at the beginning of the section.

Definition 5.21. Let φ be a WMSO formula in negation normal form, and $\Phi = \{\psi : \psi \text{ is a subformula of } \varphi\}$. Let $\mathfrak{A}$ be a finite structure, and let β be an assignment of the free variables in φ. The positions of the game $\mathcal{G}_{\text{fin}}(\mathfrak{A}, \beta, \varphi)$ are triples consisting of the structure $\mathfrak{A}$, a formula in Φ and an assignment β interpreting its free variables. The game starts at position $\langle\!\langle \mathfrak{A}, \beta, \varphi \rangle\!\rangle$, and the possible moves are as follows.

- $\langle\!\langle \mathfrak{A}, \beta, \vartheta \vee \eta \rangle\!\rangle \xrightarrow{\text{V}} \langle\!\langle \mathfrak{A}, \beta, \vartheta \rangle\!\rangle$ or $\langle\!\langle \mathfrak{A}, \beta, \vartheta \vee \eta \rangle\!\rangle \xrightarrow{\text{V}} \langle\!\langle \mathfrak{A}, \beta, \eta \rangle\!\rangle$
- $\langle\!\langle \mathfrak{A}, \beta, \vartheta \wedge \eta \rangle\!\rangle \xrightarrow{\text{F}} \langle\!\langle \mathfrak{A}, \beta, \vartheta \rangle\!\rangle$ or $\langle\!\langle \mathfrak{A}, \beta, \vartheta \wedge \eta \rangle\!\rangle \xrightarrow{\text{F}} \langle\!\langle \mathfrak{A}, \beta, \eta \rangle\!\rangle$
- $\langle\!\langle \mathfrak{A}, \beta, \exists x \psi(x) \rangle\!\rangle \xrightarrow{\text{V}} \langle\!\langle \mathfrak{A}, \beta[x \mapsto a], \psi(x) \rangle\!\rangle$ for some $a \in A$,
- $\langle\!\langle \mathfrak{A}, \beta, \forall x \psi(x) \rangle\!\rangle \xrightarrow{\text{F}} \langle\!\langle \mathfrak{A}, \beta[x \mapsto a], \psi(x) \rangle\!\rangle$ for some $a \in A$,
- $\langle\!\langle \mathfrak{A}, \beta, \exists X \psi(X) \rangle\!\rangle \xrightarrow{\text{V}} \langle\!\langle \mathfrak{A}, \beta[X \mapsto U], \psi(X) \rangle\!\rangle$ for some (finite) $U \subseteq A$,
- $\langle\!\langle \mathfrak{A}, \beta, \forall X \psi(X) \rangle\!\rangle \xrightarrow{\text{F}} \langle\!\langle \mathfrak{A}, \beta[X \mapsto U], \psi(X) \rangle\!\rangle$ for some (finite) $U \subseteq A$.

In order to obtain a Büchi game without terminal positions, we let one of the players, say Verifier, loop in positions with atomic or negated atomic formulae, and define

$$F \coloneqq \{\langle\!\langle \mathfrak{A}, \beta, \psi \rangle\!\rangle : \psi \text{ is (negated) atomic, and } \mathfrak{A}, \beta \vDash \psi\} \quad .$$

Lemma 5.22. Verifier wins the game $\mathcal{G}_{\text{fin}}(\mathfrak{A}, \varphi, \beta)$ if and only if $\mathfrak{A}, \beta \vDash \varphi$.

5.4.5 Negation in Games

In the FO and WMSO model checking game it is straightforward to come up with a move for universally quantified formulae that is dual to the move for existentially quantified formulae by letting Falsifier choose an element (or subset) instead of Verifier. This follows the intuition that Verifier has to provide a witness for which the formula holds, while Falsifier, in order to show that the formula does not hold, has to provide a suitable counterexample. Thus, we do not have to handle negation explicitly in the game, but instead we can restrict our attention to formulae given in negation normal form where the only negated subformulae are atomic. In the game graph, atoms and negated atoms correspond to terminal positions for which a winner is declared, and hence, if the atom is negated, we can simply swap the roles of winner and loser.

Unfortunately, when analysing the structure of the subgames we presented for existential quantification in an effort to come up with a subgame for universal quantification, it becomes clear that we cannot simply let Falsifier choose the subsets or components from which the single element should be chosen for the following reason. Consider the game for the existential second order quantifier. When Verifier chooses the sets (which is done incrementally), she is, at the same time, in charge of selecting appropriate decomposition tuples. And it is important that she can make her choice dependent on the remaining elements of the set that she is going to choose in the following iterations. If Falsifier were in charge of selecting the set incrementally, Verifier would not know in advance which elements Falsifier were going to choose next, so she might not be able to choose the appropriate decomposition tuple, or rather Falsifier might make his choice of elements dependent on Verifier's previous choices of decomposition tuples. Hence, the existence of a winning strategy for Verifier would still guarantee that the formula holds, but vice versa, the existence of a winning strategy for Falsifier would not necessarily imply that the formula is false.

For this reason, we cannot restrict ourselves to check formulae in negation normal form, but we have to introduce a mechanism handling the negation of subformulae such that we can replace a universally quantified subformula by the negation of an existentially quantified one. Towards this,

we introduce the notion of dualising a game with the intention that the dual game should be won by Verifier if the original game is won by Falsifier and vice versa.

Definition 5.23. Let $\mathcal{G}$ be one of the subgames described above. The *game dual to* $\mathcal{G}$, denoted $\mathcal{G}^*$, is defined as follows.

- $\mathcal{G}^*(\top, m) = \mathcal{G}(\bot, m)$.
- $\mathcal{G}^*(\bot, m) = \mathcal{G}(\top, m)$.
- $\mathcal{G}^*_{\mathrm{fin}}(\mathfrak{A}, \varphi, \beta) = \mathcal{G}_{\mathrm{fin}}(\mathfrak{A}, \neg\varphi, \beta)$.
- $\mathcal{G}^*_\exists(\varphi, m)$ is obtained from $\mathcal{G}_\exists(\varphi, m)$ by switching the roles of the players, i.e. Falsifier moves for Verifier, and vice versa, exits $\langle\!\langle \cdot \rangle\!\rangle$ are marked as dual exits $\langle\!\langle \cdot \rangle\!\rangle^*$, and F^* is the complement of F.
- $\mathcal{G}^*_{\exists_1}(\varphi, m)$ is constructed from $\mathcal{G}_{\exists_1}(\varphi, m)$ analogously.

Proposition 5.24. Verifier wins the subgame $\mathcal{G}$ if and only if Falsifier wins the dual game $\mathcal{G}^*$.

Proof. For the constant subgames and the finite-structure subgames, the claim follows from the definition and Lemma 5.22, respectively.

Considering the subgames for the existential quantifiers, we argue using the games $\hat{\mathcal{G}}_\exists$ and $\hat{\mathcal{G}}_{\exists_1}$ as defined above. Recall that $\hat{\mathcal{G}}_\exists(\varphi, m)$ is defined as $\mathcal{G}_\exists(\varphi, m)$ but has self loops on the exits and

$$F \coloneqq \{\langle\!\langle \mathfrak{A}^c_i, \psi^c, S, i \rangle\!\rangle : \mathfrak{A}^c_i \vDash \psi^c(S)\} \cup \{\langle\!\langle \varphi[X \leftarrow \varnothing], \Box, i \rangle\!\rangle : \mathcal{S}_i(\mathcal{D}) \vDash \varphi[X \leftarrow \varnothing]\}$$

such that Verifier wins if she has a strategy to reach an exit in F, and Falsifier wins if he has a strategy to reach an exit not in F or if he can force Verifier into a loop on the inner positions. In the dual game $\hat{\mathcal{G}}^*_\exists$ the assignment of the players to positions is switched, and F^* is defined as the complement of F. Thus, assuming that Verifier wins $\hat{\mathcal{G}}_\exists$, it is obvious that now Falsifier wins $\hat{\mathcal{G}}^*_\exists$ since he has a strategy to reach an exit not in F^*, and if Falsifier has a winning strategy in $\hat{\mathcal{G}}_\exists$, then Verifier wins $\hat{\mathcal{G}}^*_\exists$ playing this strategy by either reaching an exit in F^* or enforcing a loop on the inner positions which are now contained in F^*. Q.E.D.

Procedure 5.4. $\texttt{ENNF}(\varphi)$

case φ *is a literal* **return** φ
case $\varphi = \neg\psi$ **return** $\texttt{ENNF}(\texttt{NNF}(\neg\psi))$
case $\varphi = \psi \wedge \vartheta$ **return** $\texttt{ENNF}(\psi) \wedge \texttt{ENNF}(\vartheta)$
case $\varphi = \psi \vee \vartheta$ **return** $\texttt{ENNF}(\psi) \vee \texttt{ENNF}(\vartheta)$
case $\varphi = \exists x \psi(x)$ **return** $\exists x\, \texttt{ENNF}(\psi(x))$
case $\varphi = \exists X \psi(X)$ **return** $\exists X\, \texttt{ENNF}(\psi(X))$
case $\varphi = \forall x \psi(x)$ **return** $\neg\exists x\, \texttt{ENNF}(\neg\psi(x))$
case $\varphi = \forall X \psi(X)$ **return** $\neg\exists X\, \texttt{ENNF}(\neg\psi(X))$

5.4.6 Combining the Subgames

Due to the lack of a subgame for universal quantification, we consider the following normal form.

Definition 5.25. A WMSO formula φ is in *existential negation normal form* (ENNF) if it does not contain subformulae of the form $\forall x \varphi(x)$ or $\forall X \varphi(X)$ and if negation only occurs in front of atomic formulae and existential quantifiers.

Lemma 5.26. Every WMSO formula can be transformed effectively into an equivalent WMSO formula in existential negation normal form.

Proof. Procedure 5.4 provides a recursive transformation of a WMSO formula φ into existential negation normal form. Indeed, the transformed formula cannot contain universal quantifiers since they are replaced by existential ones, and due to the handling of negated subformulae, negations can eventually only occur before atoms and existential quantifiers. Using that each WMSO formula has an equivalent negation normal form, the equivalence of φ to its existential negation normal form $\texttt{ENNF}(\varphi)$ provided by Procedure 5.4 follows by a straightforward induction. Q.E.D.

Definition 5.27. Given a formula φ in ENNF, and a system of n structure equations $\mathcal{D}$, the model checking game $\mathcal{G}(\varphi, m)$, for $m \in [n]$, is constructed inductively as follows.

- $\varphi = \exists X \psi(X)$: Construct $\mathcal{G}_\exists(\psi, m)$ and merge each exit of the form $\langle\!\langle \mathfrak{A}_i^c, \psi^c, S, i \rangle\!\rangle$ with the initial vertex of $\mathcal{G}_{\text{fin}}(\mathfrak{A}_i^c, \psi^c, \beta[X \mapsto S])$, and each exit of the form $\langle\!\langle \varphi[X \leftarrow \varnothing], \square, i \rangle\!\rangle$ with the initial vertex of the game $\mathcal{G}(\text{ENNF}(\varphi[X \leftarrow \varnothing]), i)$. Note that in the latter case, the formula $\varphi[X \leftarrow \varnothing]$ does not contain X as a free variable anymore since each atom of the form $y \in X$ is replaced by $\bot$.
- $\varphi = \exists x \psi(x)$: Construct $\mathcal{G}_{\exists_1}(\psi, m)$ and merge each exit of the form $\langle\!\langle \mathfrak{A}_i^c, \psi^c, \beta : x \mapsto a, \blacksquare, i \rangle\!\rangle$ with the initial vertex of $\mathcal{G}_{\text{fin}}(\mathfrak{A}_i^c, \psi^c, \beta)$, and each exit of the form $\langle\!\langle \psi, \blacksquare, i \rangle\!\rangle$ with the initial vertex of the game $\mathcal{G}(\text{ENNF}(\psi), i)$. As above, we can assume that the formula ψ does not contain x as a free variable anymore (otherwise Verifier's choice of the decomposition tuple would be bad) since the formula speaks about a component that x is not interpreted in according to Verifier's claim.
- $\varphi = \neg\exists X \psi(X)$ or $\varphi = \neg\exists x \psi(x)$: Construct the dual game $\mathcal{G}^*_\exists(\varphi, m)$ or $\mathcal{G}^*_{\exists_1}(\varphi, \beta, m)$, respectively, and proceed as in the cases above. However, merge the *dual* exits of $\mathcal{G}^*_\exists(\varphi, m)$ or $\mathcal{G}^*_{\exists_1}(\varphi, m)$ with the initial vertices of the respective *dualised* games.
- $\varphi = \psi \vee \vartheta$: The game $\mathcal{G}(\varphi, m)$ consists of the initial position $\langle\!\langle \varphi, \square, m \rangle\!\rangle$, from which Verifier moves to the initial position of either $\mathcal{G}(\psi, m)$ or $\mathcal{G}(\vartheta, m)$.
- $\varphi = \psi \wedge \vartheta$: Analogous to the previous case, but it is Falsifier who moves to the initial position of $\mathcal{G}(\psi, m)$ or $\mathcal{G}(\vartheta, m)$.
- $\varphi = \bot$: as already described in Section 5.4.3, $\mathcal{G}(\bot, m)$ consists of only one (initial) position with a self loop, and $F := \varnothing$.
- $\varphi = \top$: $\mathcal{G}(\top, m)$ consists of only one (initial) position p_0 with a self-loop, and $F := \{p_0\}$.

The game construction continues until all exits are replaced by the appropriate subgames. The set F is obtained by taking the union of all sets F of the subgames. Note that only the subgames $\mathcal{G}_{\text{fin}}$, $\mathcal{G}(\top, i)$ and $\mathcal{G}(\bot, i)$ contribute to F.

The case distinction is complete since we never have to construct a game $\mathcal{G}(\varphi, m)$ for an atomic formula φ with free variables. In the formulae at those exits that lead into subgames of the form $\mathcal{G}(\varphi, m)$, the previously quantified free variable is eliminated. Only those exits leading to plain WMSO model checking games for finite structures handle formulae with free (and interpreted) variables.

Let us remark that the formulae in the exits of a subgame have either a smaller complexity than the formula at the initial position in terms of quantifier rank or the number of free variables, or the exit is the initial position for a model checking game on a finite structure. Hence, the subgames are linked in form of a DAG, and loops can only occur inside a subgame.

Theorem 5.28. Let φ be a WMSO sentence, let $\mathcal{D}$ be a system of n structure equations, and let $\mathcal{S}(\mathcal{D}) = (\mathfrak{A}_1, \dots, \mathfrak{A}_n)$. Then $\mathfrak{A}_m \vDash \varphi$ if and only if Verifier has a winning strategy in the Büchi game $\mathcal{G}(\varphi, m)$.

Proof. As remarked above, due to the special structure of the game graph, any infinite play either loops in one of the subgames or by reaching an (essentially terminal) vertex with a self loop in $\mathcal{G}(\top, i)$, $\mathcal{G}(\bot, i)$ or some $\mathcal{G}_{\text{fin}}(\mathfrak{A}, \varphi, \beta)$. Hence, we can prove the statement by induction over the structure of formulae and use the results we have proved about the isolated subgames.

Let $\varphi = \bot$. Then Falsifier has a winning strategy in $\mathcal{G}(\bot, m)$ since $F = \varnothing$, so no position in F is seen infinitely often. If $\varphi = \top$, Verifier has a winning strategy in $\mathcal{G}(\top, m)$ since, by definition, every position seen infinitely often is in F.

Let $\varphi = \psi \vee \vartheta$, and $\mathfrak{A}_m \vDash \varphi$. Then $\mathfrak{A}_m \vDash \psi$ or $\mathfrak{A}_m \vDash \vartheta$, and hence, by the induction hypothesis, Verifier has a winning strategy in the game $\mathcal{G}(\psi, m)$ or $\mathcal{G}(\vartheta, m)$. So she also has a winning strategy in $\mathcal{G}(\varphi, m)$ by moving to the appropriate initial position of $\mathcal{G}(\psi, m)$ or $\mathcal{G}(\vartheta, m)$. On the other hand, if she has a winning strategy in $\mathcal{G}(\varphi, m)$, then she must also have a winning strategy in at least one of $\mathcal{G}(\psi, m)$ or $\mathcal{G}(\vartheta, m)$ because these are the only two possible moves at the beginning. Hence, by induction hypothesis, $\mathfrak{A}_m \vDash \psi$ or $\mathfrak{A}_m \vDash \vartheta$, and thus also $\mathfrak{A}_m \vDash \psi \vee \vartheta = \varphi$. The case $\varphi = \psi \wedge \vartheta$ is analogous.

Let $\psi = \exists X \varphi(X)$, and assume that $\mathfrak{A}_m \vDash \exists X \varphi(X)$. By Lemma 5.18, Verifier has a strategy to win the game $\hat{\mathcal{G}}_\exists(\varphi, m)$, i.e. in the subgame $\mathcal{G}_\exists(\varphi, m)$, she has a strategy to reach an exit $\langle\!\langle \mathfrak{A}_i^c, \psi^c, S, i \rangle\!\rangle$ such that $\mathfrak{A}_i^c \vDash \psi^c(S)$ or an exit $\langle\!\langle \varphi[X \leftarrow \varnothing], \Box, i \rangle\!\rangle$ such that $\mathfrak{A}_i \vDash \varphi[X \leftarrow \varnothing]$ in finitely many steps. In the first case, she has a strategy to win the game $\mathcal{G}_{\text{fin}}(\mathfrak{A}_i^c, \psi^c, X \mapsto S)$ by Lemma 5.22, and in the latter case, she has a strategy to win from $\langle\!\langle \varphi[X \leftarrow \varnothing], \Box, i \rangle\!\rangle$ by the induction hypothesis since $\varphi[X \leftarrow \varnothing]$ is of lower complexity than ψ. Conversely, assume that Verifier has a winning strategy in $\mathcal{G}(\varphi, m)$. Since none of the positions in the subgame $\mathcal{G}_\exists(\varphi, m)$ is in F, she must not force the play into an infinite loop as she would lose then. Hence,

following her strategy, against any counter-strategy played by Falsifier, she will reach an exit of the form $\langle\langle \mathfrak{A}_i^c, \psi^c, S, i\rangle\rangle$ or $\langle\langle \varphi[X \leftarrow \varnothing], \Box, i\rangle\rangle$ from where she has a strategy to win the game $\mathcal{G}_{\text{fin}}(\mathfrak{A}_i^c, \psi^c, X \mapsto S)$ or $\mathcal{G}(\varphi[X \leftarrow \varnothing], i)$, respectively. By Lemma 5.22 and the induction hypothesis, this implies that $\mathfrak{A}_i^c \vDash \psi^c(S)$ and $\mathfrak{A}_i \vDash \varphi[X \leftarrow \varnothing]$, and finally Lemma 5.18 implies that $\mathfrak{A}_m \vDash \psi$. The case $\psi = \exists x \varphi(x)$ is analogous.

Let $\psi = \neg\exists X \varphi(X)$ (analogously for $\psi = \neg\exists x \varphi(x)$). In this case, since $\mathcal{G}(\neg\exists X\varphi(X), m) = \mathcal{G}^*(\exists X\varphi(X), m)$, by Proposition 5.24 Verifier has a winning strategy in $\mathcal{G}(\neg\exists X\varphi(X), m)$ if and only if Falsifier has a winning strategy in $\mathcal{G}^*(\exists X\varphi(X), m)$ if and only if (by induction hypothesis) $\mathfrak{A}_m \nvDash \exists X\varphi(X)$ if and only if $\mathfrak{A}_m \vDash \neg\exists X\varphi(X)$. Q.E.D.

5.5 Unbounding Quantifier

In contrast to many commonly used quantifiers, such as "there exists exactly one x", that do not increase the expressive power of MSO and are usually considered as "syntactic sugar", the so-called *unbounding quantifier* we consider in the following cannot be expressed in plain MSO. In particular, the formula $UX\varphi$ states that the size of finite sets X satisfying φ is unbounded, i.e.

$$UX\varphi(X) \equiv \text{ for all } n \in \mathbb{N}\ \exists X\varphi(X) \text{ with } X \text{ finite and } |X| \geq n \ .$$

First introduced by Bojańczyk [Boj04], MSO with this quantifier was proved to be decidable on trees with very restricted quantification patterns only. Recently, only a technical analysis of max-automata allowed to show that satisfiability of WMSO with the unbounding quantifier is decidable on the class of all labellings of $(\omega, <)$ [Boj09]. We prove that WMSO+U is decidable on all inductive structures, which is a more general result as far as the class of structures is concerned, but at the same time less general as we allow only regular labellings of the structures. For the proof, we only need to modify the construction of the model checking game presented above. Again, we fix a system $\mathcal{D}$ of n equations and let $\mathcal{S}(\mathcal{D}) = (\mathfrak{A}_1, \ldots, \mathfrak{A}_n)$.

Definition 5.29. A family $\mathcal{U} = \{S_i : i \in \mathbb{N}\}$ of finite sets is called *unbounded* in a component $\mathfrak{A}_m[c]$ if $\{i : \mathfrak{A}_m[c] \cap S_i \neq \varnothing\}$ is infinite.

The following lemma is a consequence of the fact that a system of structure equations only contains a finite number of structures.

Lemma 5.30. Let $\varphi(X)$ be a WMSO+U formula with a single free variable X. Let $\mathcal{D}$ be a system of n structure equations, $\mathcal{S}(\mathcal{D}) = (\mathfrak{A}_1, \ldots, \mathfrak{A}_n)$, and let $\mathcal{U} = \{S_i : (\mathfrak{A}_m, S_i) \models \varphi(X), |S_i| \geq i\}$ be a family of sets witnessing that $\mathfrak{A}_m \models UX\varphi(X)$. Then $\mathcal{U}$ is unbounded in some component $\mathfrak{A}_m[c]$.

Proof. For each $i \in \mathbb{N}$, let S_i be a set with more than i elements satisfying $\varphi(S_i)$. The tree obtained from the prefix closure of $\{w : (s, w) \in \bigcup_i S_i\}$ (w is the index of the element w.r.t. $\mathfrak{A}_m$) is a finitely branching tree with arbitrarily long paths (since the depth of the sets S_i is unbounded) such that, for each node v, the component $\mathfrak{A}_m[v]$ contains some element from a set S_i. By König's Lemma, this tree has an infinite path $\alpha = c\alpha_1 \in \Delta_{\mathcal{D}}^{\omega}$, with $\Delta_{\mathcal{D}} = \bigcup_\ell \Delta_\ell$, and since each S_i is finite, elements of infinitely many S_i are reachable from any node on this path. In particular, $\mathcal{U}$ is unbounded in $\mathfrak{A}_m[c]$. In general, for any $\mathfrak{A}_m[w]$, where w is a prefix of $\alpha = wc\alpha'$, the family $\mathcal{U}$ is unbounded in the component $\mathfrak{A}_m[wc]$. Q.E.D.

The above lemma justifies the following extension of the procedure $\mathtt{split}_k$ to handle formulae of the form $\varphi = UX\psi$ (where $\overline{X}_{-c}$ denotes $\overline{X}$ without X^c):

$$\mathtt{split}_k(\varphi) = \bigvee_{c=1,\ldots,k} \exists \overline{X}_{-c} UX^c \mathtt{split}_k(\psi)[X \leftarrow \bigcup_{i=1}^{k} X^i] \ .$$

The unbounding quantifier distributes over disjunctions, and the definition of TNF and the conversion procedure for the unbounding quantifier is the same as for the existential second-order quantifier. Thus, the Decomposition Theorem holds for WMSO+U as well.

We construct a model checking game for WMSO+U formulae in the same way as for WMSO formulae by handling the unbounding quantifier using the subgame for an existential set quantifier. However, it is not sufficient that Verifier has a winning strategy in the constructed game in order for a formula to hold, but instead she must have a family of winning strategies that allow her to loop arbitrarily often through particular cycles in the subgame before reaching an exit.

Definition 5.31. The subgame $\mathcal{G}_U(\varphi, m)$ is defined as $\mathcal{G}_\exists(\varphi, m)$ but Falsifier's positions of the form $\langle\!\langle (\psi_1, \ldots, \psi_n), S, i \rangle\!\rangle$ where $S \neq \varnothing$ are considered to be especially marked.

Again, to reason about the correctness of the subgame for the unbounding quantifier, we consider the game $\hat{\mathcal{G}}_U(\varphi, m)$ obtained from $\mathcal{G}_U(\varphi, m)$ by adding self loops to the exits and letting

$$F \coloneqq \{\langle\!\langle \mathfrak{A}_i^c, \psi^c, S, i \rangle\!\rangle : \mathfrak{A}_i^c \vDash \psi^c(S)\} \cup \{\langle\!\langle \varphi[X \leftarrow \varnothing], \square, i \rangle\!\rangle : \mathcal{S}_i(\mathcal{D}) \vDash \varphi[X \leftarrow \varnothing]\} \ .$$

Furthermore, we let $\mathcal{T}_\sigma(\varphi, m)$ denote the unravelling of the game graph from position $\langle\!\langle \varphi, \square, m \rangle\!\rangle$ where all branches that are not chosen by Verifier's strategy σ are pruned.

Lemma 5.32. $\mathfrak{A}_m \vDash UX\varphi(X)$ if and only if for each $n \in \mathbb{N}$, Verifier has a winning strategy σ_n such that $\mathcal{T}_{\sigma_n}(\varphi, m)$ contains at least n marked positions.

Proof. ($\Rightarrow$) Let M be the maximum number of elements in the universe of all structures in $\mathrm{Str}(\mathcal{D})$ and assume that $\mathfrak{A}_m \vDash UX\varphi(X)$. Thus, for each $n \in \mathbb{N}$ there is a set S_n with $|S_n| \geq n$ such that $\mathfrak{A}_m, S_n \vDash \varphi(X)$. Following the same arguments as in the proof of Lemma 5.18, each S_n gives rise to a winning strategy σ_n for Verifier, namely "choose the upcoming elements of S_n." Consider the strategy $\sigma_{n \cdot M}$. Since $\sigma_{n \cdot M}$ chooses elements from $S_{n \cdot M}$, and at each marked position at most M of those, it follows from $|S_{n \cdot M}| \geq n \cdot M$ that there are at least n marked positions in $\mathcal{T}_{\sigma_{n \cdot M}}(\varphi, m)$.

($\Leftarrow$) Given a winning strategy σ, we construct, as in the proof of Lemma 5.18, a set S_σ satisfying φ. Consider a strategy σ_n with at least n marked positions in $\mathcal{T}_{\sigma_n}(\varphi, m)$. Since each marked position corresponds to a choice of a non-empty subset and since these subsets are disjoint, $|S_{\sigma_n}| \geq n$. Hence, $\mathfrak{A}_m \vDash UX\varphi(X)$ as we have assumed the existence of a winning strategy for each $n \in \mathbb{N}$. Q.E.D.

The existence of such a family of strategies can be decided using a technique applied by Colcombet and Löding in the context of cost functions [CL10, Appendix]. The idea is to consider a Muller game $\tilde{\mathcal{G}}_U(\varphi, m)$ derived from $\hat{\mathcal{G}}_U(\varphi, m)$ that is constructed such that, if Verifier has a winning strategy in $\tilde{\mathcal{G}}_U(\varphi, m)$, she has a family of winning strategies in $\hat{\mathcal{G}}_U(\varphi, m)$ with the desired property.

Definition 5.33. The arena of $\tilde{\mathcal{G}}_U(\varphi, m)$ is obtained from the arena of $\hat{\mathcal{G}}_U(\varphi, m)$ by removing all positions from which Falsifier wins if playing according to the rules of the model checking subgame $\hat{\mathcal{G}}_\exists(\varphi, m)$, and further removing all exits except for those of the form $\langle\!\langle \psi[X \leftarrow \varnothing], \Box, i \rangle\!\rangle$. That is, the arena of $\tilde{\mathcal{G}}_U(\varphi, m)$ consists of that part of the winning region in the subgame $\hat{\mathcal{G}}_\exists(\varphi, m)$ on which Verifier can loop but still leave via particular exits. The priority assignment $\Omega : V \to \{0, 1, 2\}$ is defined as follows.

- Falsifier's vertices of the form $\langle\!\langle (\psi_1, \ldots, \psi_n), S, i \rangle\!\rangle$ where $S \neq \varnothing$ (i.e. the marked positions in $\mathcal{G}_U(\varphi, m)$) are assigned priority 1;
- exits and positions from where Verifier has a strategy to reach an exit are assigned priority 2;
- the remaining positions are assigned priority 0.

The Muller winning objective for Verifier is $\mathcal{F} = \{\{0, 1, 2\}, \{1, 2\}\}$.

Given a WMSO+U game and its corresponding Muller game, we obtain the following relation between the winning strategies.

Lemma 5.34. Verifier has a winning strategy in the Muller game $\tilde{\mathcal{G}}_U(\varphi, m)$ if and only if there exists a family $(\sigma_n)_{n \in \mathbb{N}}$ of winning strategies in $\hat{\mathcal{G}}_U(\varphi, m)$ satisfying the condition in Lemma 5.32.

Proof. Let $\tilde{\sigma}$ be a winning strategy for Verifier in $\tilde{\mathcal{G}}_U(\varphi, m)$, and let n be arbitrary. We claim that the strategy σ_n, to initially copy the moves of $\tilde{\sigma}$ until at least n positions with priority 1 have been visited and then use her strategy to reach the next exit and move to some $\langle\!\langle \psi[X \leftarrow \varnothing], \Box, \ell \rangle\!\rangle$, is a winning strategy for Verifier in $\hat{\mathcal{G}}_U(\varphi, m)$.

Playing according to $\tilde{\sigma}$ ensures that vertices with priority 1, i.e. vertices that are marked in $\hat{\mathcal{G}}_U(\varphi, m)$, are visited infinitely often. Furthermore, as also vertices with priority 2 are visited infinitely often, after any number of visits to vertices with priority 1, Verifier has a strategy to reach a vertex with priority 2, hence it is possible to enforce reaching some exit. Finally, since all positions in $\tilde{\mathcal{G}}_U(\varphi, m)$ are in Verifier's winning region of $\hat{\mathcal{G}}_\exists(\varphi, m)$, she can win from position $\langle\!\langle \psi[X \leftarrow \varnothing], \Box, \ell \rangle\!\rangle$ while $\mathcal{T}_{\sigma_n}(\varphi, m)$ contains at least n marked positions.

On the other hand assume that Verifier has no winning strategy in the Muller game $\tilde{\mathcal{G}}_U(\varphi, m)$. By finite-memory determinacy of Muller games, Falsifier has a winning strategy σ_F using only finite memory. Assume towards

a contradiction that there exists a family $(\sigma_n)_{n\in\mathbb{N}}$ of winning strategies for Verifier in $\hat{\mathcal{G}}_U(\varphi, m)$ with the desired properties. Since these strategies will never leave the winning region of Verifier in the game $\hat{\mathcal{G}}_\exists(\varphi, m)$, they are also suitable to play on $\hat{\mathcal{G}}_U$.

We can assume that, when playing according to some strategy σ_n, Verifier wins while visiting at least n marked nodes, i.e. by collecting at least n elements for the satisfying assignment, and finally moving to an exit, otherwise Falsifier would lose early. Let m be the size of the memory of Falsifier's winning strategy σ_F, and let $N \geq m \cdot |\tilde{\mathcal{G}}_U|$ (where the size of $\tilde{\mathcal{G}}_U$ is measured by the number of positions of the game graph). Then the play consistent with σ_N against σ_F contains a loop in the sense that some node is visited twice such that the memory state of Falsifier's strategy is exactly the same.

On the other hand, σ_N is a winning strategy for Verifier in $\hat{\mathcal{G}}_U(\varphi, m)$, so she has a strategy to reach an exit after visiting at least N marked vertices, and hence the loop contains at least one vertex with priority 2. But this implies that Verifier can enforce playing this loop ad infinitum in the Muller game $\tilde{\mathcal{G}}_U(\varphi, m)$ such that the set of priorities seen infinitely often is either $\{0, 1, 2\}$ or $\{1, 2\}$ which are both in her winning set and contradicts the assumption that σ_F is a winning strategy for Falsifier. Q.E.D.

The subgame $\mathcal{G}_U(\varphi, m)$ can be integrated in a complete model checking game exactly as the subgames $\mathcal{G}_\exists(\varphi, m)$.

5.6 Conclusion

The characterisation of the model-checking problem for WMSO on inductive structures by a finite Büchi game allows for new algorithmic approaches to model checking. On the one hand, the Büchi game could be solved directly, on the other hand, it is also possible to follow a more modular approach and implement a solver that is based on solving the individual subgames and calling itself recursively for checking the formulae at the exits of the subgame, which corresponds to the approach we took for proving the correctness. Since the subgames are in fact reachability games in which only loops must be avoided, this approach seems to be even simpler.

The merits of such a new algorithmic approach to model checking are two-fold. On the one hand, as was our main motivation, it avoids encoding structures into the binary tree. On the other hand, due to the modularity,

the algorithm is not forced to run in a bottom-up fashion like automata-based algorithms, but it can possibly simplify the occurring formulae at any point. This can be a strong advantage in contrast to the automaton construction where the structure of the subformulae is hidden in the state space and transitions. Hence, a purely automata-based technique can only benefit from advances in the methods available for complementing and determinising non-deterministic tree automata.

However, also the presented approach cannot circumvent the non-elementary lower complexity bound. But as there are instances that are hard for automata-based techniques but easy for our approach, and vice versa, the method can complement the existing techniques.

In addition to the pure algorithmic value, our method could provide new insights into the composition method and might help to understand the algebraic structure of tree languages definable in weak MSO. Similar to the presented modification of the game that yields a decision procedure for WMSO+U, the game might be extended to capture other quantifiers and maybe richer fragments of MSO.

BIBLIOGRAPHY

[Alo86] Noga Alon. Eigenvalues and expanders. *Combinatorica*, 6(2):83–96, 1986.

[AM85] Noga Alon and Vitali D. Milman. λ_1, isoperimetric inequalities for graphs, and superconcentrators. *Journal of Combinatorial Theory, Series B*, 38(1):73–88, 1985.

[BK95] David A. Basin and Nils Klarlund. Hardware verification using monadic second-order logic. In *Proceedings of the 7th International Conference on Computer Aided Verification, CAV '95*, volume 939 of *LNCS*, pages 31–41. Springer, 1995.

[BKV+81] Manuel Blum, Richard M. Karp, Oliver Vornberger, Christos H. Papadimitriou, and Mihalis Yannakakis. The complexity of testing whether a graph is a superconcentrator. *Information Processing Letters*, 13(4/5):164–167, 1981.

[Boj04] Mikołay Bojańczyk. A bounding quantifier. In *Proceedings of the 13th EACSL Annual Conference on Computer Science Logic, CSL 2004*, volume 3210 of *LNCS*, pages 41–55. Springer, 2004.

[Boj09] Mikołay Bojańczyk. Weak MSO with the unbounding quantifier. In *Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009*, volume 09001 of *LIPIcs*, pages 159–170. Schloss Dagstuhl (IBFI), 2009.

[BS05] Michael Benedikt and Luc Segoufin. Towards a characterization of order-invariant queries over tame structures. In C.-H. Luke

Ong, editor, *Proceedings of the 14th EACSL Annual Conference on Computer Science Logic, CSL 2005*, volume 3634 of *Lecture Notes in Computer Science*, pages 276–291. Springer-Verlag, 2005.

[BS09] Michael Benedikt and Luc Segoufin. Towards a characterization of order-invariant queries over tame graphs. *Journal of Symbolic Logic*, 74(1):168–186, 2009.

[Büc60] J. Richard Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6:66–92, 1960.

[Büc62] J. Richard Büchi. On a decision method in restricted second order arithmetic. In *International Congress on Logic, Methodology and Philosophy of Science*, pages 1–11. Stanford University Press, 1962.

[CL10] Thomas Colcombet and Christof Löding. Regular cost functions over finite trees. In *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010*, pages 70–79. IEEE Computer Society, 2010.

[Cou90] Bruno Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990.

[Cou96] Bruno Courcelle. The monadic second-order logic of graphs X: Linear orderings. *Theoretical Computer Science*, 160:87–143, 1996.

[CR80] Stephen A. Cook and Charles Rackoff. Space lower bounds for maze threadability on restricted machines. *SIAM Journal on Computing*, 9(3):636–652, 1980.

[EF99] Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite Model Theory*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 2 edition, 1999.

[EFT94] Heinz-Dieter Ebbinghaus, Jörg Flum, and Wolfgang Thomas. *Mathematical Logic*. Undergraduate texts in mathematics. Spinger-Verlag, 2nd edition, 1994.

[EI95a] Kousha Etessami and Neil Immerman. Reachability and the power of local ordering. *Theoretical Computer Science*, 148(2):261–279, 1995.

[EI95b] Kousha Etessami and Neil Immerman. Tree canonization and transitive closure. In *Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science, LICS 1995*, pages 331–341, 1995.

[EI00] Kousha Etessami and Neil Immerman. Tree canonization and transitive closure. *Information and Computation*, 157(1–2):2–24, 2000.

[EJ91] E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science, FoCS1991*, pages 368–377. IEEE Computer Society Press, 1991.

[EKM98] Jacob Elgaard, Nils Klarlund, and Anders Møller. MONA 1.x: new techniques for WS1S and WS2S. In *Proceedings of the 10th International Conference on Computer-Aided Verification, CAV 1998*, volume 1427 of *LNCS*, pages 516–520. Springer-Verlag, June/July 1998.

[Elg61] Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98:21–51, 1961.

[Fag74] Ronald Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In Richard M. Karp, editor, *Complexity of Computation*, volume 7 of *SIAM-AMS Proceedings*, pages 43–73. AMS, 1974.

[FV59] Solomon Feferman and Robert Lawson Vaught. The first-order properties of algebraic systems. *Fundamenta Mathematica*, 47:57–103, 1959.

[GH82] Yuri Gurevich and Leo Harrington. Trees, automata and games. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, STOC 1982*, pages 60–65. ACM Press, 1982.

[GKL+07] Erich Grädel, Phokion G. Koalitis, Leonid Libkin, Maarten Marx, Joel Spencer, Moshe Y. Vardi, Yde Venema, and Scott Weinstein. *Finite Model Theory and Its Applications*. Texts in Theoretical Computer Science. Springer, 2007.

[GM95] Erich Grädel and Gregory L. McColm. On the power of deterministic transitive closures. *Information and Computation*, 119:129–135, 1995.

[Gro96] Martin Grohe. Arity hierarchies. *Annals of Pure and Applied Logic*, 82(2):103–163, 1996.

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.

[HJJ+95] J.G. Henriksen, Jakob L. Jensen, M. Jørgensen, Nils Klarlund, B. Paige, T. Rauhe, and A. Sandholm. Mona: Monadic second-order logic in practice. In *Tools and Algorithms for the Construction and Analysis of Systems, First International Workshop, TACAS '95, LNCS 1019*, 1995.

[HLW06] Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43:439–561, 2006.

[HS08] Martin Hofmann and Ulrich Schöpp. Pure pointer programs with iteration. In Michael Kaminski and Simone Martini, editors, *Proceedings of the 17th EACSL Annual Conference on Computer Science Logic, CSL 2008*, volume 5213 of *Lecture Notes in Computer Science*, pages 79–93. Springer, 2008.

[HS10] Martin Hofmann and Ulrich Schöpp. Pure pointer programs with iteration. *ACM Transactions on Computational Logic*, 11(4):1–23, 2010.

[Imm86] Neil Immerman. Relational queries computable in polynomial time. *Information and Control*, 68(1–3):86–104, 1986.

[Imm87] Neil Immerman. Languages that capture complexity classes. *SIAM Journal of Computing*, 16(4):760–778, 1987.

[Imm88] Neil Immerman. Nondeterministic space is closed under complementation. *SIAM Journal of Computing*, 17(5):935–938, 1988.

[JJKS97] Jakob L. Jensen, Michael E. Jørgensen, Nils Klarlund, and Michael I. Schwartzbach. Automatic verification of pointer programs using monadic second-order logic. In *Proceedings of the ACM Conference on Programming Language Design and Implementation, PLDI '97*, pages 226–236, 1997.

[Lap98] Denis Lapoire. Recognizability equals monadic second-order definability for sets of graphs of bounded tree-width. In *Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science, STACS 1998*, pages 618–628, 1998.

[Lib04] Leonid Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[Mak04] Johann A. Makowsky. Algorithmic uses of the feferman-vaught theorem. *Annals of Pure and Applied Logic*, 126(1–3):159–213, 2004.

[Mar73] Grigorii Aleksandrovich Margulis. Explicit constructions of concentrators. *Problemy Peredachi Informatsii*, 9(4):71–80, 1973. Translation in: Problems of Information Transmission 10, Plenum, N.Y., 1975.

[Mar75] Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102:363–371, 1975.

[McN93] Robert McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993.

[Nur00] Juha Nurmonen. Counting modulo quantifiers on finite structures. *Information and Computation*, 160(1-2):62–87, 2000.

[Ott00] Martin Otto. Epsilon-logic is more expressive than first-order logic over finite structures. *Journal of Symbolic Logic*, 65(4):1749–1757, 2000.

[Pin73] Mark Semenovich Pinsker. On the complexity of a concentrator. In *Proceedings of the 7th International Telegraphic Conference, ITL 1973*, pages 318/1–318/4, 1973.

[Rab69] Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[Rei05] Omer Reingold. Undirected st-connectivity in log-space. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, STOC 2005*, pages 376–385. ACM, 2005.

[RVW02] Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders. *The Annals of Mathematics*, 155(1):157–187, 2002.

[SH08] Ulrich Schöpp and Martin Hofmann. Pointer programs and undirected reachability. *Electronic Colloquium on Computational Complexity (ECCC)*, 15(090), 2008.

[She75] Saharon Shelah. The monadic second order theory of order. *Annals of Mathematics*, 102:379–419, 1975.

[Tho97] Wolfgang Thomas. Ehrenfeucht games, the composition method, and the monadic theory of ordinal words. In *Structures in Logic and Computer Science*, volume 1261 of *Lecture Notes in Computer Science*, pages 118–143. Springer-Verlag, 1997.

[Tra61] Boris A. Trakhtenbrot. Finite automata and the logic of monadic predicates. *Dokl. Akad. Nauk SSSR*, 140:326–329, 1961.

[Var82] Moshe Y. Vardi. The complexity of relational query languages. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, STOC 1982*, pages 137–146. ACM, 1982.

INDEX

LEBENSLAUF

Name	Tobias Ganzow
Geburtsdatum	29. Juni 1979
Geburtsort	Seesen
Staatsangehörigkeit	deutsch

1998	Abitur
1999 – 2005	Informatikstudium an der RWTH Aachen
8. März 2005	Diplom in Informatik
2005 – 2011	Promotion am Lehr- und Forschungsgebiet Mathematische Grundlagen der Informatik, RWTH Aachen