Order-Invariance of Two-Variable Logic is Decidable

Thomas Zeume
TU Dortmund University
thomas.zeume@cs.tu-dortmund.de

Frederik Harwath
Goethe Universitt Frankfurt am Main
harwath@cs.uni-frankfurt.de

Wednesday 20$^{\text{th}}$ April, 2016

Abstract

It is shown that order-invariance of two-variable first-logic is decidable in the finite. This is an immediate consequence of a decision procedure obtained for the finite satisfiability problem for existential second-order logic with two first-order variables (ESO^2) on structures with two linear orders and one induced successor. We also show that finite satisfiability is decidable on structures with two successors and one induced linear order. In both cases, so far only decidability for monadic ESO^2 has been known. In addition, the finite satisfiability problem for ESO^2 on structures with one linear order and its induced successor relation is shown to be decidable in non-deterministic exponential time.

1 Introduction

Order-invariance plays a crucial role in several areas of finite model theory. In descriptive complexity theory, for example, various well-known results establish that a logic captures a complexity class on structures that are equipped with a linear order. Usually, in such results, the particular order on a given structure is not important. That is, the formulas constructed in the proofs are *order-invariant*, i.e. they do not distinguish different linear orders on a given structure (see Section 6 for a precise definition).

First-order logic (FO) is a logic of great importance in model theory and, consequently, order-invariant FO-sentences have been studied in the literature before, see e.g. [11, 1, 5], and the survey [23].

It is well-known that the question whether an FO-sentence is order-invariant is undecidable. Two possible remedies are to restrict either the class of structures or the logic. For the former case, it is known that order-invariance remains undecidable for colored directed paths [1] and colored star graphs [5].

In this article we study the decidability of invariance for the *two-variable fragment of first-order logic* (*two-variable logic* or FO^2 for short). This fragment is reasonably expressive and yet its satisfiability and finite satisfiability problems

are decidable [26, 20]. As a query language, it has a strong connection to the XML query language XPATH on trees [19]. Furthermore, many modal logics can be translated to FO^2 and inherit its good algorithmic properties. Those applications as well as the inability of FO^2 to express transitivity of a relation has led to an exhaustive study of the complexity of the (finite) satisfiability problem of the logic where some relation symbols are interpreted by transitive relations, equivalence relations, linear orders, successor relations and preorders (see e.g. [21, 15, 17, 2, 14, 25, 18, 27, 13]).

We establish the decidability of order-invariance for two-variable logic.

Main Result. *Order-invariance of FO^2 is decidable.*

The key to our main result is a simple observation which relates the problem to a satisfiability problem on ordered structures. A FO^2-sentence φ is *not* order-invariant if and only if there are a structure and two linear orders on its domain which are distinguished by φ. More precisely, φ is *not* order-invariant if and only if there are two linear orders $<_1$ and $<_2$ on a finite set D and a tuple of relations $\bar{R}$ on D that interprets all relation symbols in φ except for $<$ such that $(D, \bar{R}, <_1) \models \varphi$ and $(D, \bar{R}, <_2) \not\models \varphi$.

The latter statement can be seen as a satisfiability question for a sentence of the *two-variable fragment of existential second-order logic* (ESO^2) on finite ordered $(<_1, <_2)$-structures. Here, a $(<_1, <_2)$-structure is *ordered* if $<_1, <_2$ are interpreted by linear orders. Thus, if the finite satisfiability problem for ESO^2 is decidable on ordered $(<_1, <_2)$-structures, then so is order-invariance.

We take a slightly more comprehensive approach and study the finite satisfiability problem for ESO^2 on structures that are equipped with linear orders and successor relations; henceforth called *ordered structures*.

Some prior work on decidability of ESO^2 and its monadic fragment $EMSO^2$ (where only quantification of unary relations is allowed) on ordered structures has been done under the guise of the finite satisfiability problem for FO^2. An almost complete characterization of the classes of finite ordered structures for which $EMSO^2$ is decidable was obtained in a sequence of articles [17, 24, 12, 25, 18]. Only the case of finite ordered structures with at least three successor relations remains open. For ESO^2 it is only known that it is decidable on the class of finite ordered $(<, S)$-structures [10, 21, 4] and NEXPTIME-complete for (S_1, S_2)-structures [3]. We note that the latter result combined with the observation from above establishes the decidability of successor-invariance. Here and in general, if both a symbol $<$ (or $<_i$) and S (or S_i) occur in the signature of a structure, we assume that S (or S_i) is interpreted by the induced successor of $<$ (or $<_i$).

Our main technical contribution implies the decidability of order-invariance for FO^2.

Main Technical Result. *The satisfiability problem for ESO^2 is decidable on the class of finite ordered $(S_1, <_1, <_2)$-structures and on the class of finite ordered $(S_1, S_2, <_2)$-structures.*

The first part in particular closes the gap between the decidability of ESO^2 on ordered $<$-structures [21] and the undecidability of $EMSO^2$ on ordered $(<_1, <_2, <_3)$-structures [12]. More precisely we show that the decision problem can be solved in 2-NEXPTIME on $(S_1, <_1, <_2)$-structures and as fast as the emptiness problem for multicounter automata on $(S_1, S_2, <_2)$-structures. We conjecture that there is an EXPSPACE-algorithm for the former problem.

Class of ordered structures	EMSO^2	ESO^2
One linear order/successor		
S-structures	NExpTime-complete [7]	NExpTime-complete ★
$<$-structures	NExpTime-complete [7]	NExpTime-complete [21]
$(S, <)$-structures	NExpTime-complete [7]	NExpTime-complete ★
Two linear orders/successors		
(S_1, S_2)-structures	NExpTime-complete [3]	NExpTime-complete [3]
$(<_1, <_2)$-structures	in ExpSpace [24]	in 2-NExpTime ★
$(S_1, S_2, <_2)$-structures	in Multicounter-Emptiness [18]	in Multicounter-Emptiness ★
$(S_1, <_1, <_2)$-structures	in ExpSpace [25]	in 2-NExpTime ★
$(S_1, <_1, S_2, <_2)$-structures	Undecidable [17]	Undecidable [17]
Many linear orders/successors		
$(<_1, <_2, <_3)$-structures	Undecidable [12]	Undecidable [12]
(S_1, S_2, S_3)-structures	?	?
$(S_1, S_2, S_3, \dots)$-structures	?	?

Figure 1: Summary of results on satisfiability of ESO^2 and EMSO^2 on finite ordered structures. Results contained in this article are marked by ★.

For proving our results we generalize techniques used in [17], [24] and [18]. In the course of this, a technique developed by Otto [21] for ESO^2 on ordered $<$-structures turns out to be very valuable also for the more general structures considered here. We emphasize that while the basic techniques used for proving the results in this article are inspired by previous work, they are employed in a technically more demanding context.

As an introduction to the methods used for proving the technical main result, we show that ESO^2 is NExpTime-complete on finite $(<, S)$-structures. This is also of some independent interest. Previously, only the satisfiability problem for EMSO^2 was known to be NExpTime-complete on this class of structures. The extension of ESO^2 with counting quantifiers was recently shown to be decidable on such structures, but only with very high complexity [4].

Discussion

Our results on satisfiability of ESO^2 on finite ordered structures also help to make known results more robust, and therefore less vulnerable for confusion.

In work on two-variable first-logic originating from applications in verification and database theory often only unary relation symbols are allowed in formulas besides the linear orders and successors (see e.g. [7, 2]). In most articles this restriction is stated clearly, or, at least the intended meaning is clear from the context. Yet abbreviations like FO$^2(<)$ do not reflect this restriction. The arising ambiguity can lead to confusions for non-experts and readers only skimming an article. We definitely have been confused a couple of times, and we seem not be alone.

The second-order perspective taken in this article resolves this confusion. Furthermore, our results obviate the necessity to distinguish the two variants in the context of decidability of the finite satisfiability problem to some extent.

Outline

We introduce our notation and basic tool set in Section 2. In Section 3 we present two different ideas for deciding ESO^2 on finite $(<, S)$-structures. These ideas are generalized to finite ordered $(S_1, <_1, <_2)$-structures and finite ordered $(S_1, S_2, <_2)$-structures in Sections 4 and 5. We discuss order-invariance in Section 6. We conclude in Section 7.

2 Preliminaries

In this section we introduce ordered structures and two-variable logic.

2.1 Ordered Structures

In this article we consider logical formulas that are interpreted over *ordered* structures, i.e. relational structures where some relation symbols are interpreted by linear orders and their corresponding successor relations. We assume that all structures are *finite* structures with a *non-empty domain*.

A *linear order* $<$ is a transitive, total and antisymmetric relation, that is, $a < b$ and $b < c$ implies $a < c$; $a < b$ or $b < a$ holds; and not both $a < b$ and $b < a$ are satisfied at the same time for all elements a, b and c. The *induced successor relation* S of a linear order $<$ contains a tuple (a, b) if $a < b$ and there is no element c such that $a < c < b$.

Let $O \subseteq \{<_1, <_2, \ldots, \} \cup \{S_1, S_2, \ldots, \}$ where the $<_i$ and S_i are binary relation symbols. An O-structure $\mathfrak{A}$ is *ordered* if each relation $<_i^{\mathfrak{A}}$ is a linear order and each relation $S_i^{\mathfrak{A}}$ is the induced successor relation of a linear order. Furthermore, if $S_i \in O$ and $<_i \in O$, then $S_i^{\mathfrak{A}}$ is the induced successor relation of $<_i^{\mathfrak{A}}$. For convenience we often identify relation symbols with their respective interpretations. We use the symbols S and $<$ if an ordered structure only has one linear order.

We say that two distinct elements a and b of $\mathfrak{A}$ are $<_i$-*close* if $S_i(a, b)$ or $S_i(b, a)$. Otherwise, a and b are $<_i$-*remote*. We say that a and b are *close* if they are $<_i$-close for some $S_i \in O$. Otherwise, a and b are *remote*. If the signature O contains S_i but not $<_i$ we also say S_i-close and so on.

For an ordered structure $\mathfrak{A}$ with two underlying linear orders $<_1$ and $<_2$ it will often be convenient to think of $\mathfrak{A}$ as a point set in the two-dimensional plane where $<_1$ orders points along the x-axis and $<_2$ orders points along the y-axis (see Figure 2). Following this conception, we say that an element a is to the *left* or *right* of an element b if $a <_1 b$ or $b <_1 a$, respectively, and that it is *below* or *above* b if $a <_2 b$ or $b <_2 a$. Accordingly, we will speak of *leftmost* and *rightmost* elements.

2.2 Two-Variable Logic

The *two-variable fragment of first-order logic* (short: FO^2) is the restriction of first-order logic where only two variables can be used, though those two variables can be quantified multiple times. The *two-variable fragment of existential second-order logic* (short: ESO^2) consists of all formulas of the form $\exists \bar{R} \varphi$ where $\bar{R}$ is a tuple of relation variables and φ is a FO^2-sentence. Since each FO^2-atom can contain at most two variables we assume, in the entire article and without loss

of generality, that all relation symbols are of arity at most two; see [8, page 5] for a justification.

Two ESO^2-sentences ψ and ψ' are $\mathcal{K}$-*equivalent* for a class of structures $\mathcal{K}$, if $\mathfrak{A} \models \psi$ if and only if $\mathfrak{A} \models \psi'$ for all structures $\mathfrak{A} \in \mathcal{K}$. An ESO^2-sentence is *satisfiable on a class of structures $\mathcal{K}$* if it has a model $\mathfrak{A} \in \mathcal{K}$. We will also say that a sentence is $\mathcal{K}$-*satisfiable*.

Every ESO^2-formula can be translated into a formula of a very simple shape which is $\mathcal{K}$-equivalent (see e.g. [26] and [9, page 17]). In this article $\mathcal{K}$ will always be a class of ordered O-structures.

Lemma 2.1 (Scott Normal Form). *For every* ESO^2-*formula* $\varphi = \exists \bar{R}\, \xi$ *there is a $\mathcal{K}$-equivalent* ESO^2-*formula* $\varphi' = \exists \bar{R}'\, \xi'$ *such that the formula* ξ' *is of the form*

$$\forall x \forall y \psi(x, y) \wedge \bigwedge_i \forall x \exists y \psi_i(x, y)$$

where ψ and all ψ_i are quantifier-free formulas. Moreover, φ' can be computed in polynomial time.

Note that this lemma is slightly stronger than the usual Scott normal form lemma because interpretations for some relation symbols are restricted by $\mathcal{K}$. The usual proof carries over since interpretations of symbols that occur in φ are preserved while going from a model of φ to a model of φ' and vice versa.

In this work it will be convenient to use an even stronger normal form. Our plan is to rephrase the satisfiability problem for ESO^2 into the satisfiability of a set of existential and universal constraints. This is a simple generalization of the approach taken in [25].

We need the following notions. Let T be a signature. A *unary T-type* is a maximally consistent conjunction of unary T-literals using variable x only. A *(strictly) binary T-type* is a maximally consistent conjunction of binary T-literals using variables x and y, where the conjunction includes the conjunct $x \neq y$. Each element of a structure has a unique unary T-type and each pair of elements has a unique binary T-type. If γ is the binary type of (a, b), then we denote the type of (b, a) by $\bar{\gamma}$. The set of unary and binary T-types are denoted by Σ_T and Γ_T.

A *constraint problem over T* is a tuple $C = (C_\exists, C_\forall)$ where $C_\exists$ is a set of existential constraints and $C_\forall$ is a set of universal constraints. An *existential constraint $c_\exists$* is a tuple (σ, E) where $\sigma \in \Sigma_T$ and $E \subseteq \Gamma_T \times \Sigma_T$. A structure with domain D satisfies $c_\exists$ if for every element $a \in D$ of unary type σ there is a $(\gamma, \tau) \in E$ and an element $b \in D$ of unary type τ such that (a, b) has binary type γ. A *universal constraint $c_\forall$* is a tuple (σ, γ, τ) where $\sigma, \tau \in \Sigma_T$ and $\gamma \in \Gamma_T$. A structure with domain D satisfies $c_\forall$ if no tuple $(a, b) \in D^2$ has binary type γ if a and b have unary types σ and τ. A structure is a *solution* of C if it satisfies all constraints in $C_\exists$ and $C_\forall$. The problem C is solvable if it has a finite solution. The *size $|C|$* of C is $|\Sigma_T| + |\Gamma_T| + |C_\forall| + |C_\exists|$. We emphasize that $C_\exists$ specifies required patterns whereas $C_\forall$ specifies forbidden patterns.

Let $\mathcal{K}$ be class of structures over a signature $T' \subseteq T$. The problem C is *solvable on the class of structures $\mathcal{K}$* if it has a solution $\mathfrak{A}$ such that the restriction of $\mathfrak{A}$ to T' belongs to $\mathcal{K}$. Such a solution will also be called $\mathcal{K}$-*solution*.

The following lemma shows that ESO^2-sentences can be translated into existential second-order constraint problems preserving satisfiability.

Lemma 2.2. *For every* ESO^2*-sentence* φ *there is a constraint problem* C *such that* φ *is* $\mathcal{K}$*-satisfiable if and only if* C *has a* $\mathcal{K}$*-solution. The constraint problem* C *can be computed in exponential time in* $|\varphi|$.

Proof. Let φ be an ESO^2-sentence. Without loss of generality we can assume, by Lemma 2.1, that $\varphi = \exists \bar{R}\,\xi$ where ξ is of the form $\forall x \forall y \psi(x,y) \wedge \bigwedge_i \forall x \exists y \psi_i(x,y)$ with quantifier-free ψ and ψ_i. Let T be the signature of the FO^2-formula ξ. Further let Σ and Γ be the sets of unary and binary T-types.

A $\mathcal{K}$-satisfiability equivalent constraint problem for φ is now constructed by translating $\forall x \forall y \psi(x,y)$ and $\forall x \exists y \psi_i(x,y)$ into universal and existential constraints, respectively.

Observe that, for a quantifier-free formula in disjunctive normal form $\bigvee_i \psi_i$, an equivalent quantifier-free formula in disjunctive normal form can be constructed, where (1) each disjunct is of the form $\sigma(x) \wedge \tau(y) \wedge \gamma(x,y)$ where σ and τ are unary types and γ is a conjunction of binary literals that use variables x and y, and (2) for all unary types σ and τ there is such a disjunct.

Thus the first conjunct of ξ is equivalent to a formula

$$\chi = \forall x \forall y \bigvee_{\sigma,\tau \in \Sigma} \Big(\sigma(x) \wedge \tau(y) \wedge \gamma_{\sigma,\tau}(x,y) \Big)$$

The formula χ is equivalent to the following formula:

$$\mu = \forall x \forall y \bigwedge_{\sigma,\tau \in \Sigma} \Big(\sigma(x) \wedge \tau(y) \to \gamma_{\sigma,\tau}(x,y) \Big)$$

To see this, let us consider a model $\mathfrak{A}$ of χ. For all elements a,b of $\mathfrak{A}$ of types σ, τ, respectively, one of the disjuncts has to be satisfied. Since each element satisfies exactly one unary type, the disjunct $\sigma(x) \wedge \tau(y) \wedge \gamma_{\sigma,\tau}(x,y)$ is satisfied and therefore also μ is satisfied.

Now, consider a model $\mathfrak{A}$ of μ. Then for all elements a,b of $\mathfrak{A}$ the conjunction is satisfied. In particular the conjunct $\sigma(x) \wedge \tau(y) \to \gamma_{\sigma,\tau}(x,y)$ is satisfied, where σ and τ are the types of a and b, respectively. Therefore also $\sigma(x) \wedge \tau(y) \wedge \gamma_{\sigma,\tau}(x,y)$ and hence χ are satisfied.

The formula μ can be easily translated into a set of universal constraints $C_\forall$. For all unary types σ and τ, the set $C_\forall$ contains a constraint (σ, γ, τ) for each binary type γ that is not consistent with $\gamma_{\sigma,\tau}(x,y)$.

Next we show how to translate the second part of ξ into existential constraints. As before, we translate ψ_i into the disjunctive normal form $\bigvee_{\sigma,\tau \in \Sigma} \big(\sigma(x) \wedge \tau(y) \wedge \gamma_{\sigma,\tau}(x,y) \big)$. Sorting by σ and moving the existential quantifier inwards as far as possible yields that every conjunct of the second part of ψ can be written as follows:

$$\chi_i = \forall x \bigvee_{\sigma \in \Sigma} \Big(\sigma(x) \wedge \exists y \bigvee_{\tau \in \Sigma} \big(\tau(y) \wedge \gamma_{\sigma,\tau}(x,y) \big) \Big)$$

Using a similar argument as above one can show that each χ_i is equivalent to a formula:

$$\mu_i = \forall x \bigwedge_{\sigma \in \Sigma} \Big(\sigma(x) \to \exists y \bigvee_{\tau \in \Sigma} \big(\tau(y) \wedge \gamma_{\sigma,\tau}(x,y) \big) \Big)$$

The formulas μ_i can be easily translated into a set of existential constraints $C_\exists$. For all i and all σ, the set $C_\exists$ contains an existential constraint (σ, E) where E

contains all tuples (γ, τ) that are consistent with $\gamma_{\sigma,\tau}$ where τ is a unary type and γ is a binary type.

Observe that the sets $C_\exists$ and $C_\forall$ can be computed in exponential time (where the exponential blow-up comes from using the types). $\qquad\qquad\square$

The following notion will be used frequently. If an element has a unary type σ we also say that it is σ-*labeled*. Consider a constraint problem C over T, a T-structure $\mathfrak{A}$, a σ-labeled element a and an existential constraint (σ, E). An element b is a (σ, E)-*witness* of a if there is a $(\gamma, \tau) \in E$, the element b has unary type τ and (a, b) has T-type γ. A *witness* of a is an element that is a (σ, E)-witness for some existential constraint.

The following notations will be convenient for studying finite satisfiability of ESO^2 on ordered structures. A constraint problem has a (finite) $(<_1, <_2)$-*solution* if it is solvable on the class of finite ordered $(<_1, <_2)$-structures. We use similar definitions for $\mathrm{ESO}^2(S_1, S_2)$ and so on.

For a constraint problem over $O \cup T$ where O is a subset of the symbols $<, <_1, <_2, \dots$ and $S, S_1, S_2, \dots$ we will denote existential and universal constraints in a slightly different fashion. To this end observe that all unary O-types are trivial (e.g. $x < x \wedge \neg S(x, x)$) and can, without loss of generality, be omitted. Each binary $T \cup O$-type is a conjunction of a binary O-type and binary T-type. Therefore we can write existential constraints of constraint problems over $T \cup O$ as tuples (σ, E) where $\sigma \in \Sigma_T$ and E is a set of tuples (d, γ, τ) with $d \in \Gamma_O$, $\gamma \in \Gamma_T$ and $\tau \in \Sigma_T$. Similarly universal constraints are written as tuples $(\sigma, d, \gamma, \tau)$ where $\sigma, \tau \in \Sigma_T$, $\gamma \in \Gamma_T$ and $d \in \Gamma_O$.

3 Warm-Up: One Linear Order and One Successor

Before proving the main results we study the complexity of the ESO^2-satisfiability problem on ordered $(S, <)$-structures. Decidability has been established in [10]. Combining Lemma 2.4 in [10] and a construction from [6, Corollary 9.2] yields at best a non-deterministic double-exponential upper bound. In this section we obtain an optimal non-deterministic exponential upper bound.

Theorem 3.1. ESO^2-*satisfiability on finite ordered $(S, <)$-structures is* NEXPTIME-*complete.*

This result should be compared with the known results that deciding EMSO^2 on ordered $(S, <)$-structure and deciding ESO^2 on ordered $<$-structures are NEXPTIME-complete [7, 21].

We present two different approaches for proving Theorem 3.1; a small model property based approach and an automata-based approach. Only the former approach leads to a non-deterministic exponential time algorithm. Later both approaches will be generalized in different directions to obtain decidability for larger fragments. The small model approach, combined with a technique from [24, 25], is used to obtain a decision algorithm for ESO^2 on $(S_1, <_1, <_2)$-structures in Section 4. The automata-based approach is used to solve the ESO^2-satisfiability problem on $(S_1, S_2, <_2)$-structures in Section 5.

To decide whether an ESO^2-sentence φ has a model which is a finite $(S, <)$-structure, φ is converted into a satisfiability equivalent constraint problem

C whose size is exponential in the size of φ using Lemma 2.2. We show, in Lemma 3.3, that if C has a finite $(S, <)$-solution then it has a $(S, <)$-solution of size at most N where N is polynomial in the size of C. Hence, a non-deterministic exponential time algorithm can guess a structure of size at most N and verify that it is indeed a solution of C. This proves Theorem 3.1. In the automata-based approach the satisfiability of C is checked by a finite state automaton, see Lemma 3.4.

The following notion will be useful in both approaches. Let w be a word. A position a of w is called (σ, τ, k)-*rich* if there are at least k σ-labeled positions (strictly) before a and at least k τ-labeled positions (strictly) after a. A position a of w is (σ, τ, k)-*poor* if there are at most k σ-labeled positions (strictly) before a and at most k τ labeled positions (strictly) after a. We stress that if a position is not (σ, τ, k)-rich than it is not necessarily (σ, τ, k)-poor, and vice versa.

Lemma 3.2. *If a word has no (σ, τ, k)-rich position then it has a $(\sigma, \tau, k+1)$-poor position.*

Proof. Let w be a word with no (σ, τ, k)-rich position. Towards a contradiction assume that w has no $(\sigma, \tau, k + 1)$-poor position. Then, by definition, for each position i of w there are at least $k + 1$ σ-labeled positions strictly left of i or at least $k + 1$ τ-labeled positions strictly right of i.

In particular there are at least $(k + 1)$ σ-labeled positions to the left of the last position of w. Let i be the $(k+1)$th σ-labeled position of w. Since i has only k σ-labeled positions to its left, it must have at least $k + 1$ τ-labeled positions to is right. Hence, i is (σ, τ, k)-rich; a contradiction. $\square$

As outlined above, the proof of Theorem 3.1 follows immediately from the following small model property and Lemma 2.2. The proof of the small model property is in the same spirit as the proof of Lemma 2.3 in [10].

Lemma 3.3. *If a constraint problem C has a finite $(S, <)$-solution then it has such a solution of size polynomial in $|C|$.*

Proof. Assume that $C \stackrel{\text{def}}{=} (C_\exists, C_\forall)$ is a constraint problem over a signature $T \cup \{S, <\}$. Let $\Sigma \stackrel{\text{def}}{=} \Sigma_T$ and $\Gamma \stackrel{\text{def}}{=} \Gamma_T$. Let $k \stackrel{\text{def}}{=} 3|\Gamma|$.

We show that if C has a finite $(S, <)$-solution $\mathfrak{A}$ with at least N elements then it also has a $(S, <)$-solution $\mathfrak{B}$ with $|\mathfrak{B}| < N$. The number N is polynomial in $|C|$ and will be specified later.

For each element a of $\mathfrak{A}$ denote by $W(a)$ a set containing witnesses for a for each existential constraint. An element $b \in W(a)$ is a *local witness* if $S(a, b)$, $a = b$, or $S(b, a)$. Otherwise it is a *non-local witness*.

We first construct, from $\mathfrak{A}$, a solution $\mathfrak{A}'$ of C whose domain and unary types coincide with the domain and unary types of $\mathfrak{A}$ but whose set of non-local witnesses is small. Afterwards we argue that a smaller solution $\mathfrak{B}$ can be obtained from $\mathfrak{A}'$ by removing some elements.

Towards constructing $\mathfrak{A}'$ let Z_1 be the set that contains, for every $\sigma \in \Sigma$, the first $k + 1$ σ-positions and the last $k + 1$ σ-positions of $\mathfrak{A}$ (if these exist). Further let Z_2 be a set that contains a witness for each position in Z_1 and each existential constraint, and let $Z \stackrel{\text{def}}{=} Z_1 \cup Z_2$. Observe that Z_1 and Z_2 are of size polynomial in $|\Sigma||\Gamma|$.

We reassign some of the binary T-types of $\mathfrak{A}$ in order to obtain $\mathfrak{A}'$. The goal is that every element a in $\mathfrak{A}'$ has a set $W'(a)$ of witnesses for $C_\exists$ such that (1)

$W(a)$ and $W'(a)$ coincide with respect to local witnesses, and (2) all non-local witnesses in $W'(a)$ are from Z. To this end let $\sigma, \tau \in \Sigma$ and $d \stackrel{\text{def}}{=} x < y \wedge \neg S(x,y)$. Denote $\bar{d} \stackrel{\text{def}}{=} y < x \wedge \neg S(y,x)$.

If there is a (σ, τ, k)-rich position u in $\mathfrak{A}$ then we reassign the binary types of all σ- and τ-positions a and b satisfying d by using a technique employed by Otto for constructing small models for satisfiable $\text{FO}^2(<)$-sentences [21]. For completeness we recall the construction. Let $A = A_1 \cup A_2 \cup A_3$ with disjoint A_i and $|A_i| = |\Gamma|$ contain the first k σ-labeled elements. Similarly let $B = B_1 \cup B_2 \cup B_3$ with disjoint B_i and $|B_i| = |\Gamma|$ contain the last k τ-labeled elements. Assume that $\Gamma \stackrel{\text{def}}{=} \{\gamma_1, \ldots, \gamma_m\}$. Then:

(A1) Witnesses for elements in $A \cup B$ are assigned as follows:

  (a) If $a \in A_i$ is a σ-labeled element that has a (d, γ_ℓ, τ)-witness $b \in W(a)$ in $\mathfrak{A}$ then the binary T-type of (a, b') is set to γ_ℓ where b' is the ℓth element of B_i. The element b' is the intended (d, γ_ℓ, τ)-witness of a in $\mathfrak{A}'$.

  (b) If $b \in B_i$ is a τ-labeled element that has a $(\bar{d}, \gamma_\ell, \sigma)$-witness $a \in W(b)$ in $\mathfrak{A}$ then the binary T-type of (b, a') is set to γ_ℓ where a' is the ℓth element of A_{i+1} (where $i+1$ is calculated modulo 3). The element a' is the intended $(\bar{d}, \gamma_\ell, \tau)$-witness of b in $\mathfrak{A}'$.

(A2) Witnesses for all other tuples of σ- and τ-labeled elements are assigned as follows:

  (a) If $a \notin A$ is a σ-labeled element that has a (d, γ_ℓ, τ)-witness $b \in W(a)$ in $\mathfrak{A}$ then the binary T-type of (a, b') in $\mathfrak{A}'$ is γ_ℓ where b' is the ℓth element of B_1. The element b' is the (d, γ_ℓ, τ)-witness of a in $\mathfrak{A}'$.

  (b) If $b \notin B$ is a τ-labeled element that has a $(\bar{d}, \gamma_\ell, \sigma)$-witness $a \in W(b)$ in $\mathfrak{A}$ then the binary T-type of (b, a') in $\mathfrak{A}'$ is γ_ℓ where a' is the ℓth element of A_1. The element a' is the (d, γ_ℓ, τ)-witness of b in $\mathfrak{A}'$.

(A3) If a tuple (a, b) with σ-labeled a, τ-labeled b and satisfying d has not been assigned a binary T-type in $\mathfrak{A}'$ so far, then it inherits its type from $\mathfrak{A}$.

Observe that no conflicts arise from (A1) and (A2). Furthermore, for all σ- and τ-labeled elements a there is a set $W'(a)$ satisfying (1) and (2). Moreover, no conflicts with universal constraints arise since no new types have been created. This concludes the case when there is a (σ, τ, k)-rich position in $\mathfrak{A}$.

If there is no (σ, τ, k)-rich position u in $\mathfrak{A}$ then Conditions (1) and (2) are already satisfied for all σ- and τ-labeled positions a and b satisfying $d = x < y \wedge S(x,y)$. To see this we argue as follows. By Lemma 3.2 there is a $(\sigma, \tau, k+1)$-poor position v. Now let a be a σ-labeled position. If $a \leq v$ then all (d, γ, τ)-witnesses $b \in W(a)$ are contained in Z by construction (as a is one of the $k+1$ leftmost σ-labeled positions). If $a > v$ then all (d, γ, τ)-witnesses of a are among the last $k+1$ τ-labeled positions which are also contained in Z. The argument for τ-labeled positions is symmetric.

The structure $\mathfrak{A}'$ thus constructed satisfies (1) and (2).

It remains to construct $\mathfrak{B}$. If $|\mathfrak{A}'| > N$ for $N \geq c(|\Sigma||\Gamma|)^4$ for an appropriate constant c then there are positions $a_1, a_2 \notin Z$ with successors b_1 and b_2 such that (i) there is no position $z \in Z$ with $a_1 < z < a_2$, (ii) a_1 and a_2 have the

same unary T-type, and (iii) (a_1, b_1) and (a_2, b_2) have the same binary T-type. The solution $\mathfrak{B}$ is obtained from $\mathfrak{A}'$ by removing all elements between a_1 and a_2 (including a_2), and assigning to (a_1, b_2) the binary type of (a_1, b_1) in $\mathfrak{A}'$. Then $\mathfrak{B}$ satisfies all universal constraints of C since no new types have been created. Further $\mathfrak{B}$ satisfies all existential constraints of C since elements inherit their local and non-local witnesses from $\mathfrak{A}'$, and $\mathfrak{A}'$ only uses elements from Z as non-local witnesses. $\qquad\square$

Now we present the automata-based approach. As discussed above, this approach only yields an exponential space algorithm, yet it will later be used as the basis for a decision algorithm for a larger fragment.

Lemma 3.4. *For every constraint problem C there is a non-deterministic finite state automaton $\mathcal{A}$ such that C has a finite $(S, <)$-solution if and only if $L(\mathcal{A})$ is non-empty.*

Here $L(\mathcal{A})$ denotes the language recognized by $\mathcal{A}$.

Proof. Assume that $C \stackrel{\text{def}}{=} (C_\exists, C_\forall)$ is a constraint problem over a signature $T \cup \{S, <\}$. Let $\Sigma \stackrel{\text{def}}{=} \Sigma_T$ and $\Gamma \stackrel{\text{def}}{=} \Gamma_T$.

Without loss of generality we assume that the witnesses requested by the existential constraints of C do not contradict universal constraints. More precisely, for an existential constraint (σ, E) and every $(d, \gamma, \tau) \in E$ we assume that there is no universal constraint $(\sigma, d, \gamma, \tau)$.

We construct a finite state automaton $\mathcal{A}$ such that $\mathcal{A}$ accepts a word over Σ if and only if C has a solution. Intuitively the automaton $\mathcal{A}$ interprets words as $(S, <)$-structures with no binary relations from T. In order to accept a word w, it has to verify that binary types can be assigned to all pairs of positions in a way consistent with C.

The main difficulty in verifying the existence of an assignment of binary types is to ensure that the types of tuples (a, b) and (b, a) are consistent. More precisely, if (a, b) has to be typed $\gamma \in \Gamma$ due to some existential constraint and if (b, a) has to be typed $\gamma' \in \Gamma$ due to some other existential constraint, then γ and γ' have to be compatible, that is $\gamma' = \bar{\gamma}$.

In the following we describe the construction of $\mathcal{A}$, afterwards we argue that the construction is correct. For an easier exposition of the automaton $\mathcal{A}$, we often assume that positions are labeled with some extra information. It can be easily verified that those labels could also be guessed by the automaton (or, alternatively, they could be contained in an extended alphabet).

(E) We assume that for every existential constraint (σ, E), all σ-labeled positions a are labeled with a fresh label $(\sigma, d, \gamma, \tau)$ such that $(d, \gamma, \tau) \in E$. The intention is that the (σ, E)-witness of a satisfies (d, γ, τ).

The automaton has to verify that binary T-types can be assigned such that the witnesses declared in (E) exist and all pairs of positions satisfy the universal constraints. To this end the automaton handles positions that are $<$-close to each other and positions that are $<$-remote from each other in a different way.

Dealing with positions that are close to each other is simple. Each position a has at most two positions that are $<$-close to it: there might be a position b_1 with $S(a, b_1)$ and a position b_2 with $S(b_2, a)$. The positions b_1 and b_2 might not exist (if a is the first or last position with respect to $<$). For all a the automaton can guess and verify the binary types for (a, b_1) and (a, b_2).

10

(L1) (Local types) We assume that each position a is labeled by up to two labels $\gamma_1, \gamma_2 \in \Gamma$. The intention is that γ_1 is the binary T-type of (a, b_1) (if the position b_1 exists), and likewise for γ_2.

(L2) (Consistency of local types) The automaton verifies that the labels are consistent, that is, e.g., if a and b are positions with $S(a, b)$ then the label γ_1 of a (i.e. the type guessed for (a, b)) is compatible with the type γ_2 of b (i.e. the type guessed for (b, a)).

(L3) (Local witnesses) For every σ-labeled position a that is labeled by $(\sigma, d, \gamma, \tau)$ due to (E), the automaton verifies that if $d = S(x, y)$ then the label γ_1 of a is γ and that its successor is labeled with τ. Likewise for $d = S(y, x)$.

(L4) (Local universal constraints) For all σ- and τ-labeled positions a and b with $S(a, b)$ the automaton verifies that if a is labeled γ_1 then there is no universal constraint $(\sigma, S(x, y), \gamma_1, \tau)$. Likewise for $S(b, a)$.

Verifying the existence of a type assignment for positions that are remote from each other is more intricate. The automaton $\mathcal{A}$ has to check that there is at least one binary T-type consistent with the universal constraints that can be assigned to every pair of far-away positions and that witnesses for the existential constraints can be assigned consistently according to (E).

The former condition can be verified easily (and analogously to (L4)):

(U) For all σ- and τ-labeled positions a and b satisfying $d = x < y \wedge \neg S(x, y)$ the automaton verifies that there is a γ such that there is no universal constraint $(\sigma, d, \gamma, \tau)$. Similarly for $\bar{d}$.

Testing that existential witnesses can be assigned to remote positions is more involved. We discuss how $\mathcal{A}$ verifies that binary types for all σ- and τ-labeled positions a and b satisfying $d \stackrel{\text{def}}{=} x < y \wedge \neg S(x, y)$ can be assigned; the other case is symmetric. Let $\bar{d} \stackrel{\text{def}}{=} d(y, x)$. To verify that binary types can be assigned to such σ- and τ-labeled positions, the automaton $\mathcal{A}$ guesses whether there is a (σ, τ, k)-rich position u with $k \stackrel{\text{def}}{=} 3(|\Gamma| + 3)$.

If such a position u exists then the automaton tests that every σ-labeled position has as many τ-labeled positions to its right as is required by (E), and that every τ-labeled position has sufficiently many σ-labeled positions to its left. If this is the case then binary types can be assigned using an assignment technique similar to Otto's technique [21] for reducing the size of models for $\text{FO}^2(<)$ sentences (see the correctness proof below).

If there is no (σ, τ, k)-rich position u then, by Lemma 3.2 there is a $(\sigma, \tau, k+1)$-poor position v. The automaton exploits this structure to guess and verify the binary types in this case.

More precisely, for all σ and τ the automaton does the following:

(E1) It guesses whether there is a (σ, τ, k)-rich position u. The correctness of the guess can be easily verified by $\mathcal{A}$.

(E2) If there is such a position u then $\mathcal{A}$ verifies the following:

  (a) If a σ-labeled position a is labeled by $(\sigma, d, \gamma_1, \tau), \ldots, (\sigma, d, \gamma_m, \tau)$ in (E) then there are m τ-labeled positions to the right of a.

(b) Symmetrically, if a τ-labeled position a is labeled by $(\tau, \bar{d}, \gamma_1, \sigma)$, ..., $(\tau, d, \gamma_m, \sigma)$ in (E) then there are m σ-labeled positions to the right of a.

(E3) If there is no (σ, τ, k)-rich position then $\mathcal{A}$ guesses a $(\sigma, \tau, k+1)$-poor position v. Then:

(a) We assume that positions are labeled by the following extra information (using fresh labels depending on σ and τ):

- The ith σ-labeled position a to the left of v is labeled by σ_i. The ith τ-labeled position a to the right of v is labeled by τ_i. (As v is $(\sigma, \tau, k+1)$-poor, there are at most $k+1$ such σ_i and τ_i.)
- The intended witnesses for σ_i- and τ_i-labeled positions are labeled as follows:
    - If the σ_i-labeled position a is labeled by $(\sigma, \gamma, d, \tau)$ in (E), then there is a τ-labeled position b which is also labeled with $(\sigma_i, \gamma, d, \tau)$ such that (a, b) satisfies d.
    - Likewise, if the τ_i-labeled position a is labeled by $(\tau, \gamma, \bar{d}, \sigma)$ in (E), then there is a σ-labeled position b which is also labeled with $(\tau_i, \gamma, \bar{d}, \sigma)$ such that (a, b) satisfies $\bar{d}$.
- Positions with intended witnesses from σ_i- and τ_i-labeled positions are labeled as follows:
    - Each position a to the left of v that is labeled by $(\tau, \gamma, \bar{d}, \sigma)$ in (E) is also labeled with $(\tau, \gamma, \bar{d}, \sigma_i)$ for some i such that the tuple (a, b) satisfies $\bar{d}$ where b is the σ_i-labeled position b.
    - Likewise, each $(\sigma, \gamma, d, \tau)$-labeled position a to the right of v is labeled with $(\sigma, \gamma, d, \tau_i)$ for some i such that the tuple (a, b) satisfies d where b is the σ_i-labeled position b.

(b) The automaton verifies that the labels are consistent, that is:

- No τ-labeled position to the left of v is labeled with $(\sigma_i, \gamma, d, \tau)$ and with $(\tau, \gamma', \bar{d}, \sigma_i)$ where γ and γ' are not reverse types.
- Likewise, no σ-labeled position to the right of v is labeled with $(\tau_i, \bar{\gamma}', d, \sigma)$ and with $(\sigma, \gamma, d, \tau_i)$ where γ and γ' are not reverse types.

We emphasize again that all labels assigned in (E3) can also be guessed by $\mathcal{A}$.

We argue that the construction of $\mathcal{A}$ is correct. Clearly, if the set C of constraints is satisfiable by a $(S, <)$-structure then the corresponding word is accepted by $\mathcal{A}$ (the labels in (E), (L1), (E2) and (E3a) are assigned according to the solution of C).

Now, assume that $\mathcal{A}$ accepts a word w and let ρ be an accepting run. We argue that a $(S, <)$-structure $\mathfrak{A}$ that satisfies C can be obtained from w. The elements of $\mathfrak{A}$ are the positions of w ordered as in w and the unary type of an element v is its label in w.

We describe how to assign binary types. To this end we first assign, for every position a, types witnessing the existential constraints. Afterwards all so far non-typed pairs are typed by some type admissible by the universal constraints.

For each pair (a, b) of nodes with $S(a, b)$, the edge (a, b) in T is typed with the type guessed in (L1). No binary type conflicts arise from this due to (L2).

For non-local pairs of nodes, binary types of witnesses are assigned simultaneously for all σ- and τ-labeled nodes. In the following assume that a and b are σ- and τ-labeled nodes and that (a, b) satisfies $d \stackrel{\text{def}}{=} x < y \wedge \neg S(x, y)$,

If there is a position u as in (E1) then the witnesses can be assigned as in [21]. For the sake of completeness we recall the assignment strategy. Let A be the first k σ-labeled nodes and assume that $A = A_1 \cup A_2 \cup A_3$ with disjoint A_i and $|A_i| = |\Gamma|$. Likewise let $B = B_1 \cup B_2 \cup B_3$ be the last k τ-labeled nodes with disjoint B_i and $|B_i| = |\Gamma|$. Further assume that $\Gamma \stackrel{\text{def}}{=} \{\gamma_1, \ldots, \gamma_m\}$.

Then the witness types are assigned as follows:

(A1) The witnesses for elements A and B are assigned as follows:

    (a) If a position $a \in A_i$ is labeled by $(\sigma, d, \gamma_\ell, \tau)$ in ρ then its γ_ℓ-witness is the ℓth element b of $B_i \setminus N(a)$ where $N(a)$ contains a and all elements that are close to a.

    (b) If a position $b \in B_i$ is labeled by $(\tau, d, \gamma_\ell, \sigma)$ in ρ then its γ_ℓ-witness is the ℓth element of $A_{i+1} \setminus N(b)$ where $i + 1$ is calculated modulo 3.

(A2) The witnesses for elements not in A and B are assigned as follows:

    (a) If a σ-labeled element $a \notin A$ is also labeled with $(\sigma, d, \gamma_\ell, \tau)$ then its γ_ℓ-witness is the ℓ-last element of $B \setminus N(a)$ (which is to the right of a due to (E2)).

    (b) If a τ-labeled element $a \notin B$ is also labeled with $(\tau, d, \gamma_\ell, \sigma)$ then its γ_ℓ-witness is the ℓth element in $A \setminus N(a)$.

The type assignments from (A1) ensure that all elements in A and B have their witnesses. The type assignments from (A2) ensure that all other elements have their witnesses. Observe that for $a \neq b$ at most one of the pairs (a, b) and (b, a) is assigned a type in (A1) and (A2); this determines the type of the other pair.

If there is no position u as in (E1) then the automaton guessed a position v in run ρ. The witness types are assigned according to the labels assumed in (E3a), that is:

(A'1) The witnesses for the at most $k + 1$ σ-labeled elements to the left of v and the at most $k + 1$ τ-labeled elements to the right of v are assigned as follows:

    • If a is a σ-labeled position to the left of v that is labeled with $(\sigma, d, \gamma, \tau)$ and σ_i in ρ and if b is the position labeled with $(\sigma_i, d, \gamma, \tau)$ in ρ then (a, b) is labeled γ.

    • Likewise, if a is a τ-labeled position to the right of v that is also labeled with $(\tau, \bar{d}, \gamma, \sigma)$ and τ_i in ρ and if b is the position labeled with $(\tau_i, d, \bar{\gamma}, \tau)$ in ρ then (a, b) is labeled $\bar{\gamma}$.

(A'2) The witnesses for the other elements are assigned as follows:

    • If a is a τ-labeled position to the left of v that is labeled with $(\tau, \bar{d}, \gamma, \sigma_i)$ in ρ then (a, b) is labeled with γ where b is the position labeled with σ_i in ρ.

- Likewise, if a is a σ-labeled position to the right of v that is labeled with $(\sigma, \bar{d}, \gamma, \tau_i)$ in ρ then (x, y) is labeled with $\bar{\gamma}$ where b is the position labeled with τ_i in ρ.

Note that the assignments are consistent due to (E3c).

Thus, so far types have been assigned such that all elements have witnesses, ensuring that the existential constraints are satisfied. All remaining so far non-typed edges are typed by some type admissible by the universal constraints. Such types exist due to condition (U). □

4 Two Linear Orders and One Successor

In this section we will show that ESO^2-satisfiability problem on finite $(S_1, <_1, <_2)$-structures is decidable. It is known that ESO^2 is decidable on $<$-structures [21] and that EMSO^2 is decidable on finite $(<_1, <_2)$-structures. We combine the approaches used for those two results as well as the technique introduced in Lemma 3.3 to obtain a nondeterministic double-exponential upper bound. We conjecture that this bound can be improved to exponential space by generalizing the methods from [25].

Theorem 4.1. ESO^2-*Satisfiability on finite ordered* $(S_1, <_1, <_2)$-*structures is in* 2-NExpTime.

The result immediately follows from the following small solution property and Lemma 2.2.

Lemma 4.2. *If a constraint problem C has a finite $(S_1, <_1, <_2)$-solution then it has such a solution of size exponential in $|C|$.*

Proof. We follow the proof outline employed for EMSO^2 in [24]; yet for dealing with binary relations the individual steps have to be generalized. For consistency with the proofs in [24] we prove the statement of the lemma for $(<_1, S_2, <_2)$-constraint problems. For the description below, recall that structures with two linear orders can be viewed as point sets in the plane (cf. Section 2.1).

In order to establish the exponential solution property for $(<_1, S_2, <_2)$-constraint problems, we show that smaller solutions can be constructed from large solutions. To this end we assign a *profile* $Pro(c)$ to each element c of a solution $\mathfrak{A}$ in such a way that if the profiles $Pro(c_1)$ and $Pro(c_2)$ of two elements c_1 and c_2 of $\mathfrak{A}$ with $c_1 <_2^{\mathfrak{A}} c_2$ coincide then a solution $\mathfrak{B}$ with fewer elements can be constructed from $\mathfrak{A}$ by deleting all elements a with $c_1 <_2^{\mathfrak{A}} a \leq_2^{\mathfrak{A}} c_2$ and shifting the elements b with $c_2 <_2^{\mathfrak{A}} b$ along the $<_1^{\mathfrak{A}}$-dimension. Deleting elements might, of course, also eliminate some witnesses of remaining elements. The profiles of elements will be defined such that new witnesses can be assigned.

We make this more precise now. Assume that C is a $(<_1, S_2, <_2)$-constraint problem with signature $T \cup \{<_1, S_2, <_2\}$ and let $\Sigma \overset{\text{def}}{=} \Sigma_T$ and $\Gamma \overset{\text{def}}{=} \Gamma_T$. Let $k \overset{\text{def}}{=} 3|\Gamma|$. Further let $\mathfrak{A}$ be a solution of C with domain D. For every element a of $\mathfrak{A}$, we also fix a set $W(a)$ of elements witnessing that the existential constraints of C are satisfied for a.

We extend the notion of profiles introduced in [24] to structures with arbitrary binary relations. The profile of an element c shall capture all relevant information about elements below c that have witnesses above c, and vice versa. Storing

information for all such witnesses in the profiles is not possible, yet it turns out that less information is sufficient for being able to construct smaller solutions.

For defining the profile of c, we first specify a set $P(c)$ of elements that are important for c. Then, roughly speaking, the profile will be defined as the substructure of $\mathfrak{A}$ induced by $P(c)$. The set $P(c)$ contains c, its $<_2^{\mathfrak{A}}$-successor, a set of special elements $A(c)$ as well as some of the witnesses $A_W(c)$ of elements in $A(c)$. The last two sets will be defined next. The set $A(c)$ is the union of the sets $A_{\sigma,\min,\downarrow}(c)$, $A_{\sigma,\max,\downarrow}(c)$, $A_{\sigma,\min,\uparrow}(c)$, and $A_{\sigma,\max,\uparrow}(c)$, for all $\sigma \in \Sigma$, defined as follows:

1. The set $A_{\sigma,\min,\downarrow}(c)$ contains the $k+1$ leftmost σ-labeled points that are below c (if they exist).

2. The set $A_{\sigma,\min,\uparrow}(c)$ contains the $k+1$ leftmost σ-labeled points that are above c (if they exist), but excluding the $<_2^{\mathfrak{A}}$-successor of c.

3. The set $A_{\sigma,\max,\downarrow}(c)$ contains the $k+1$ rightmost σ-labeled points that are below c (if they exist).

4. The set $A_{\sigma,\max,\uparrow}(c)$ contains the $k+1$ rightmost σ-labeled points that are above c (if they exist), but excluding the $<_2^{\mathfrak{A}}$-successor of c.

Intuitively $A_W(c)$ contains the relevant witnesses of elements in $A(c)$. It is the union of the sets $W_{\sigma,\min,\downarrow}^{\tau}(c)$, $W_{\sigma,\min,\uparrow}^{\tau}(c)$, $W_{\sigma,\max,\downarrow}^{\tau}(c)$, and $W_{\sigma,\max,\uparrow}^{\tau}(c)$, for all unary types $\sigma, \tau \in \Sigma$, defined as follows:

1. The set $W_{\sigma,\min,\downarrow}^{\tau}(c)$ contains, for all $a \in A_{\sigma,\min,\downarrow}(c)$, all witnesses $b \in W(a)$ that are to the right of a and above c.

2. The set $W_{\sigma,\min,\uparrow}^{\tau}(c)$ contains, for all $a \in A_{\sigma,\min,\uparrow}(c)$, all witnesses $b \in W(a)$ that are to the right of a and below c.

3. The set $W_{\sigma,\max,\downarrow}^{\tau}(c)$ contains, for all $a \in A_{\sigma,\min,\downarrow}(c)$, all witnesses $b \in W(a)$ that are to the left of a and above c.

4. The set $W_{\sigma,\max,\uparrow}^{\tau}(c)$ contains, for all $a \in A_{\sigma,\min,\uparrow}(c)$, all witnesses $b \in W(a)$ that are to the left of a and below c.

The set of important points for c is $P(c) \stackrel{\text{def}}{=} \{c, s(c)\} \cup A(c) \cup A_W(c)$ where $s(c)$ is the unique element satisfying $S_2^{\mathfrak{A}}(c, s(c))$. The profile $Pro(c)$ of c is the structure $(\mathcal{P}(c), c)$ where $\mathcal{P}(c)$ is the substructure of $\mathfrak{A}$ induced by $P(c)$. Observe that relation $S_2^{\mathcal{P}(c)}$ is not necessarily a successor relation.

Now we show that if two profiles $Pro(c_1)$ and $Pro(c_2)$ of elements $c_1 \neq c_2$ are isomorphic then a solution $\mathfrak{B}$ with fewer elements can be constructed from $\mathfrak{A}$. The domain of $\mathfrak{B}$ is the set $D' \subseteq D$ which contains all elements a with $a \leq_2^{\mathfrak{A}} c_1$ or $c_2 <_2^{\mathfrak{A}} a$ (assuming without loss of generality that $c_1 <_2^{\mathfrak{A}} c_2$). We now describe how the relations $<_1^{\mathfrak{B}}$, $<_2^{\mathfrak{B}}$ and $S_2^{\mathfrak{B}}$ as well as the interpretations of symbols from T are constructed for $\mathfrak{B}$.

Towards defining $<_1^{\mathfrak{B}}$, $<_2^{\mathfrak{B}}$ and $S_2^{\mathfrak{B}}$ in $\mathfrak{B}$, we fix an embedding θ that maps every element u of $\mathfrak{A}$ to a point $\theta(u) \in \mathbb{Q} \times \mathbb{Q}$ such that $\theta(u_1).\mathsf{x} < \theta(u_2).\mathsf{x}$ if and only if $u_1 <_1^{\mathfrak{A}} u_2$, and $\theta(u_1).\mathsf{y} < \theta(u_2).\mathsf{y}$ if and only if $u_1 <_2^{\mathfrak{A}} u_2$. Here $<$ is the usual linear order on the rational numbers, and $p.\mathsf{x}$ and $p.\mathsf{y}$ denote the x- and y-component of a point $p \in \mathbb{Q} \times \mathbb{Q}$.

From θ we define an embedding θ' of the elements of $\mathfrak{B}$ into $\mathbb{Q} \times \mathbb{Q}$ which will be used to obtain the order relations on $\mathfrak{B}$. Intuitively θ' keeps the positions of elements below c_1 but shifts elements above c_2 along the x-direction in order to make those points consistent with the profile of c_1. The embedding is defined as follows:

(P1) For all elements $u \in D'$ with $u \leq_2^{\mathfrak{A}} c_1$, define $\theta'(u) \stackrel{\text{def}}{=} \theta(u)$.

(P2) For all elements $u \in D'$ with $c_2 <_2^{\mathfrak{A}} u$ the embedding $\theta'(u)$ is defined as follows. Assume that $c_1^1, \ldots, c_1^n$ are the elements of $P(c_1)$ ordered by $<_1^{\mathfrak{A}}$ and $c_2^1, \ldots, c_2^n$ are the elements of $P(c_2)$ ordered by $<_1^{\mathfrak{A}}$. Observe that $c_2^1 \leq u \leq c_2^n$ by the definition of $P(c_2)$. Assume that $\theta(c_2^i).\mathsf{x} \leq \theta(u).\mathsf{x} \leq \theta(c_2^{i+1}).\mathsf{x}$. Then the position of u in the new orders is obtained by shifting the element u such that its x-coordinate is between the x-coordinates of the elements c_1^i and c_1^{i+1}. More precisely $\theta'(u).\mathsf{y} \stackrel{\text{def}}{=} \theta(u).\mathsf{y}$ and $\theta'(u).\mathsf{x}$ is defined as follows. If $u = c_2^i$ for some i then $\theta'(u).\mathsf{x} = \theta(c_1^i).\mathsf{x}$. Otherwise,

$$\theta(c_1^i).\mathsf{x} + \frac{\theta(u).\mathsf{x} - \theta(c_2^i).\mathsf{x}}{\theta(c_2^{i+1}).\mathsf{x} - \theta(c_2^i).\mathsf{x}} (\theta(c_1^{i+1}).\mathsf{x} - \theta(c_1^i).\mathsf{x})$$

It might happen, that the embedding θ' maps an element u to a point in $\mathbb{Q} \times \mathbb{Q}$ whose x-coordinate is already used by another element v. This can be dealt with in the same way as in [24, Lemma 8]. The x-coordinates are assigned sequentially: first the ones for (P1), then the remaining elements of $c_2^1, \ldots, c_2^n$, and finally all remaining elements for (P2). No conflicts can arise for the first two cases. If the designated x-coordinate of an element u assigned in the third case is already used by an element v then the x-coordinate of u is shifted by a very small distance. Namely, if w is the element whose x-coordinate is the largest with $\theta'(w).\mathsf{x} < \theta'(v).\mathsf{x}$ assigned so far then the new x-coordinate of u is shifted in between $\theta'(w).\mathsf{x}$ and $\theta'(v).\mathsf{x}$. Note that such a w always exists due to the first two cases.

In $\mathfrak{B}$, the symbol $<_1$ is interpreted by the linear order $<_1^{\mathfrak{B}}$ such that $a <_1^{\mathfrak{B}} b$ if and only if $\theta'(a).x < \theta'(b).\mathsf{x}$. The interpretation of $S_1^{\mathfrak{B}}$ is induced by $<_1^{\mathfrak{B}}$. The interpretations of $<_2$ and S_2 are defined accordingly, but with respect to the y-coordinates. Observe that $<_2^{\mathfrak{B}}$ is induced by $<_2^{\mathfrak{A}}$ and that for all tuples (a, b) with either $a \leq_2^{\mathfrak{A}} c_1$ and $b \leq_2^{\mathfrak{A}} c_1$, or $c_1 <_2^{\mathfrak{A}} a$ and $c_1 <_2^{\mathfrak{A}} b$, the relative order of the elements in $\mathfrak{B}$ is the same.

The unary type of an element a in $\mathfrak{B}$ is the same as in $\mathfrak{A}$. It remains to assign binary T-types for tuples of elements of $\mathfrak{B}$. The binary type of tuples (a, b) with either $a \leq_2^{\mathfrak{B}} c_1$ and $b \leq_2^{\mathfrak{B}} c_1$ or $c_1 <_2^{\mathfrak{B}} a$ and $c_1 <_2^{\mathfrak{B}} b$ is inherited from $\mathfrak{A}$. The tuple (c_1, b) with $(c_1, b) \in S_2^{\mathfrak{B}}$ inherits the type of the tuple (c_1, u) with $(c_1, u) \in S_2^{\mathfrak{A}}$ in $\mathfrak{A}$.

We now explain how the binary types of all tuples (a, b) with $a \leq_2^{\mathfrak{B}} c_1 <_2^{\mathfrak{B}} b$ and $(a, b) \notin S_2^{\mathfrak{B}}$ are assigned. Our focus is on the case when $a <_1^{\mathfrak{B}} b$; the case $b <_1^{\mathfrak{B}} a$ is symmetric. In the following let $d \stackrel{\text{def}}{=} x <_1 y \wedge x <_2 y \wedge \neg S_2(x, y)$. We simultaneously assign all binary T-types to all σ-labeled a and τ-labeled b such that (a, b) satisfies d.

Depending on the structure of the points in $A(c_1)$ we distinguish two cases. To this end let $a_1, \ldots, a_n$ be the elements of $A(c_1)$ ordered by $<_1^{\mathfrak{A}}$. Denote by $w(c_1)$ the sequence $(\sigma_1, d_1), \ldots, (\sigma_n, d_n)$ where σ_i is the unary type of a_i and $d_i = \uparrow$ if $c_1 <_2^{\mathfrak{A}} a_i$, $d_i = S$ if $S(c_1, a_i)$, $d_i = \cdot$ if $c_1 = a_i$ and $d_i = \downarrow$ if $a_i <_2^{\mathfrak{A}} c_1$.

If there is a $((\sigma,\downarrow),(\tau,\uparrow),k)$-rich position u in $w(c_1)$ then we assign the binary T-types of all tuples (a,b) satisfying d in $\mathfrak{B}$ using the technique employed by Otto and also used in Lemma 3.3. Let $A = A_1 \cup A_2 \cup A_3$ with disjoint A_i and $|A_i| = |\Gamma|$ be the set that contains the first k elements of $A_{\sigma,\min,\downarrow}(c_1)$ in $\mathfrak{B}$. Similarly let $B = B_1 \cup B_2 \cup B_3$ with disjoint B_i and $|B_i| = |\Gamma|$ contain the last k elements of $A_{\tau,\max,\uparrow}(c_2)$ in $\mathfrak{B}$. Then the binary types are assigned as in Lemma 3.3:

(A1) Witnesses for elements in $A \cup B$ are assigned as follows:

  (a) If, in $\mathfrak{A}$, an element $a \in A_i$ is σ-labeled and has a (d,γ_ℓ,τ)-witness $b \in W(a)$ then the binary T-type of (a,b) in $\mathfrak{B}$ is γ_ℓ where b is the ℓth element of B_i. The element b is the (d,γ_ℓ,τ)-witness of a in $\mathfrak{B}$.

  (b) If, in $\mathfrak{A}$, an element $b \in B_i$ is τ-labeled and has $(\bar{d},\gamma_\ell,\sigma)$-witness $a \in W(b)$ then the binary T-type of (b,a) in $\mathfrak{B}$ is γ_ℓ where a is the ℓth element of A_{i+1} (where $i+1$ is calculated modulo 3). The element a is the (d,γ_ℓ,τ)-witness of b in $\mathfrak{B}$.

(A2) Witnesses for all other tuples of σ- and τ-labeled elements are assigned as follows:

  (a) If, in $\mathfrak{A}$, an element $b \notin B$ is σ-labeled and has a (d,γ_ℓ,τ)-witness $b \in W(a)$ then the binary T-type of (a,b) in $\mathfrak{B}$ is γ_ℓ where b is the ℓth element of B_1. The element b is the (d,γ_ℓ,τ)-witness of a in $\mathfrak{B}$.

  (b) If, in $\mathfrak{A}$, an element $b \notin B$ is a τ-labeled and has a $(\bar{d},\gamma_\ell,\sigma)$-witness $a \in W(b)$ then the binary type of (b,a) in $\mathfrak{B}$ is γ_ℓ where a is the ℓth element of A_1. The element a is the (d,γ_ℓ,τ)-witness of b in $\mathfrak{B}$.

(A3) If a tuple $(a,b) \in D' \times D'$ such that, in $\mathfrak{A}$, the element a is σ-labeled, b is τ-labeled and (a,b) satisfies d, and if (a,b) has not been assigned a binary T-type so far, then a type is assigned as follows. Since $Pro(c_1) \cong Pro(c_2)$, there is a τ-labeled element $b' \in D$ such that (a,b') satisfies d in $\mathfrak{A}$. The tuple (a,b) inherits its binary type from (a,b') in $\mathfrak{B}$.

This concludes the binary type assignments in the case when there is a $((\sigma,\downarrow),(\tau,\uparrow),k)$-rich position u in $w(c)$. Note that this case is settled without using the fact that the profiles contain the elements added by A_W.

If there is no $((\sigma,\downarrow),(\tau,\uparrow),k)$-rich position u in $w(c_1)$ then we assign the binary T-types to σ-labeled a and τ-labeled b satisfying d as explained below.

We observe that in $\mathfrak{A}$ the $(d,\cdot,\tau)$-witnesses for all σ-labeled elements $a <_2^{\mathfrak{A}} c_1$ are in $W_{\sigma,\min,\downarrow}^\tau(c_1) \cup A_{\tau,\max,\uparrow}(c_1)$. To see this we argue as in Lemma 3.3. By Lemma 3.2 there is a $((\sigma,\downarrow),(\tau,\uparrow),k+1)$-poor position in $w(c_1)$. Let v be the minimal such position. Now let $a <_2^{\mathfrak{A}} c_1$ be a σ-labeled position. If $a \leq_1^{\mathfrak{A}} v$ then all $(d,\cdot,\tau)$-witnesses $b \in W(a)$ are contained in $W_{\sigma,\min,\downarrow}^\tau(c_1)$ by construction (as a is one of the $k+1 <_1^{\mathfrak{A}}$-smallest σ-labeled elements below c_1). If $a >_1^{\mathfrak{A}} v$ then all $(d,\cdot,\tau)$-witnesses of a are among the elements $A_{\tau,\max,\uparrow}(c_1)$ (as there are only at most $k+1$ τ-labeled elements above c_1 and to the right of a). Similarly all $(\bar{d},\cdot,\sigma)$-witnesses for all τ-labeled elements $b >_2^{\mathfrak{A}} c_2$ are in $A_{\sigma,\min,\downarrow}(c_2) \cup W_{\tau,\max,\uparrow}(c_2)$.

Let π be an isomorphism of $(\mathcal{P}(c_1),c_1)$ and $(\mathcal{P}(c_2),c_2)$. We use the above observation to assign binary T-types as follows:

(B1) Witnesses for σ-labeled a with $a <_2^{\mathfrak{B}} c_1$ are assigned as follows. If $b \in W(a)$ is a (d, γ, τ)-witness of a in $\mathfrak{A}$ then $\pi(b)$ is the (d, γ, τ)-witness of a in $\mathfrak{B}$. That is, the binary type of $(a, \pi(b))$ in $\mathfrak{B}$ is γ.

(B2) Witnesses for τ-labeled b with $b >_2^{\mathfrak{B}} c_2$ are assigned as follows. If $a \in W(b)$ is a $(\bar{d}, \gamma, \sigma)$-witness of b in $\mathfrak{A}$ then $\pi(a)$ is the (d, γ, τ)-witness of b in $\mathfrak{B}$. That is, the binary type of $(b, \pi(a))$ in $\mathfrak{B}$ is γ.

(B3) If $(a, b) \in D' \times D'$ has not been assigned a binary T-type so far, then a type is assigned as follows. Since $Pro(c_1) \cong Pro(c_2)$, there is a τ-labeled element $b' \in D$ such that (a, b') satisfies d in $\mathfrak{A}$. The tuple (a, b) inherits its binary type in $\mathfrak{B}$ from (a, b).

This concludes the binary type assignments in the case when there is no $((\sigma, \downarrow), (\tau, \uparrow), k)$-rich position u in $w(c)$.

We shortly argue why the construction is correct. The assignments in (A1) and (A2) as well as (B1) and (B2) ensure that no conflicting types are assigned to (a, b) and (b, a). No conflicts with universal constraints arise by the assignments from (A1)-(A3) and (B1)-(B3), as no new types are introduced due to the choice of θ'. Finally, after the assignments (A1)-(A2) and (B1)-(B2), each element a has a (d, γ, τ)-witness in $\mathfrak{B}$ if it has a (d, γ, τ)-witness in $\mathfrak{A}$. Similarly for $(\bar{d}, \gamma, \sigma)$-witnesses.

$\square$

5   Two Successors and One Linear Order

In this section we show that ESO^2 is decidable on finite $(S_1, S_2, <_2)$-structures. This has only been known for EMSO^2 so far [18].

Theorem 5.1. ESO^2-satisfiability on finite ordered $(S_1, S_2, <_2)$-structures is decidable.

More precisely it is decidable as fast as emptiness for multicounter automata.

The theorem is proved using the automata-based approach along the same lines as in Lemma 3.4. To this end we first define *linearly ordered data automata* (short: LODA), a restriction of *ordered data automata* which were introduced in [18]. Then we show that each constraint problem can be translated into a LODA such that the constraint problem has a finite $(S_1, S_2, <_2)$-solution if and only if the automaton accepts some linearly ordered data word. Theorem 5.1 then follows from Lemma 2.2 and the decidability of the emptiness problem for LODA (see Theorem 5.3).

Ordered data automata have been introduced for studying EMSO^2 on $(S_1, S_2, \prec_2)$-structures where $\prec_2$ is a preorder relation. For convenience we simplify the automaton model in order to study ESO^2 on plain $(S_1, S_2, <_2)$-structures.

A *linearly ordered data word* is a word $w = (\sigma_1, d_1) \dots (\sigma_n, d_n)$ with (σ_i, d_i) from $\Sigma \times \mathbb{N}$ such that $\{d_1, \dots, d_n\}$ is a contiguous interval in $\mathbb{N}$ and $d_i \neq d_j$ for all $i \neq j$. Each linearly ordered data word represents a $(S_1, <_1, S_2, <_2)$-structures extended by unary relations interpreting symbols from a signature T in a canonical way. The linear order $<_1$ and its successor are represented by the positional order, while $<_2$ and S_2 are encoded by the order of the data values
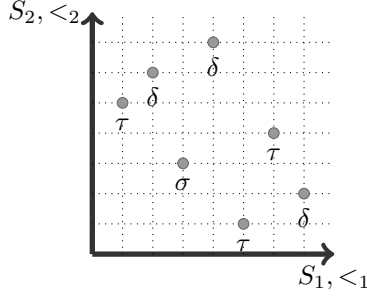
Figure 2: A $(S_1, <_1, S_2, <_2)$-structure represented as a point set in the two-dimensional plane. Here all other binary relations are disregarded. A linearly ordered data word corresponding to the structure is $((\tau, 5), (\delta, 6), (\sigma, 3), (\delta, 7), (\tau, 1), (\tau, 4), (\delta, 2))$.

$d_1, \ldots, d_n$. The unary symbols are encoded by the unary types Σ over T. See Figure 2 for an illustration.

The *resorting* of w is the string $\sigma_{i_1} \ldots \sigma_{i_n}$ such that the data values $d_{i_1}, \ldots, d_{i_n}$ are sorted in ascending order. The *string projection* of w is the string $\sigma_1 \ldots \sigma_n$. The *marked string projection* of w is its string projection annotated by information about the relationship of data values of adjacent positions. We make this more precise. The *marking* $m_i = (m, m')$ of position i is a tuple from $\Sigma_M = \{-\infty, -1, +1, \infty, -\}^2$ and is defined as follows. If $i = 1$ (or $i = n$) then $m = -$ (or $m' = -$), otherwise:

$$
m = \begin{cases}
-\infty & \text{if } d_{i-1} < d_i - 1 \\
-1 & \text{if } d_{i-1} = d_i - 1 \\
+1 & \text{if } d_{i-1} = d_i + 1 \\
\infty & \text{if } d_{i-1} > d_i + 1
\end{cases}
$$

Similarly m' is defined using d_{i+1} instead of d_{i-1}. The marked string projection of w is the string $(\sigma_1, m_1) \ldots (\sigma_n, m_n)$ over the alphabet $\Sigma \times \Sigma_M$. The string projection of the linearly ordered data word $(\tau, 5)$ $(\delta, 6)$ $(\sigma, 3)$ $(\delta, 7)$ $(\tau, 1)$ $(\tau, 4)$ $(\delta, 2)$ from Figure 2 is $\tau\delta\sigma\delta\tau\tau\delta$. Its marked string projection is $(\tau, (-, +1))$ $(\delta, (-1, -\infty))$ $(\sigma, (+\infty, +\infty))$ $(\delta, (-\infty, -\infty))$ $(\tau, (+\infty, +\infty))$ $(\tau, (-\infty, -\infty))$ $(\delta, (+\infty, -))$, and its resorting is $\tau\delta\sigma\tau\tau\delta\delta$.

A *linearly ordered data automaton (short: LODA)* $\mathcal{A}$ over alphabet Σ is a tuple $(\mathcal{B}, \mathcal{C})$ where $\mathcal{B}$ is a (non-deterministic) finite state string transducer with input alphabet $\Sigma \times \Sigma_M$ (where Σ_M is the set of markings) and some output alphabet Σ'; and $\mathcal{C}$ is a finite state automaton over Σ'.

A LODA $\mathcal{A} = (\mathcal{B}, \mathcal{C})$ works as follows. First, for a given linearly ordered data word w, the transducer $\mathcal{B}$ reads the marked string projection of w. A run ρ_B of the transducer defines a unique new labeling of each position. Let w' be the linearly ordered data word thus obtained from w. The finite state automaton $\mathcal{C}$ runs over the resorting of w' yielding a run ρ_C. The run $\rho_{\mathcal{A}} = (\rho_B, \rho_C)$ of $\mathcal{A}$ is accepting, if both ρ_B and ρ_C are accepting. The automaton $\mathcal{A}$ accepts w if there is an accepting run of $\mathcal{A}$ on w. The set of linearly ordered data words accepted by $\mathcal{A}$ is denoted by $\mathcal{L}(\mathcal{A})$. We refer to [18] for more details.

19

Example 5.2. Consider the data language L over $\Sigma = \{\sigma, \tau, \rho\}$ that contains all linearly ordered data words with a unique σ-position x with a τ-position y to its right such that the data value of y is the successor of the data value of x (see Figure 2 for a linearly ordered data word in L). A LODA $\mathcal{A} = (\mathcal{B}, \mathcal{C})$ can recognize L as follows. For a linearly ordered data word, the automaton $\mathcal{B}$ checks that there is a unique σ-position x, guesses the τ-position y to the right of x and colors this position y with a fresh label τ' (using the transduction). The automaton $\mathcal{C}$ checks that the τ'-position is the data successor of the σ-position.

The following theorem follows from Theorem 7 and Corollary 11 in [18].

Theorem 5.3. *Emptiness of LODA is decidable.*

More precisely it is decidable as fast as emptiness for multicounter automata.

Next we show that the satisfiability problem for ESO^2 on finite $(S_1, S_2, <_2)$-structures reduces to the non-emptiness problem for LODAs.

Lemma 5.4. *For every constraint problem C there is a LODA $\mathcal{A}$ such that C has a finite $(S_1, S_2, <_2)$-solution if and only if $L(\mathcal{A})$ is non-empty.*

Proof sketch. Assume that $C \stackrel{\text{def}}{=} (C_\exists, C_\forall)$ is a constraint problem over a signature $T \cup \{S_1, S_2, <_2\}$ and let $\Sigma \stackrel{\text{def}}{=} \Sigma_T$ and $\Gamma \stackrel{\text{def}}{=} \Gamma_T$. As in the proof of Lemma 3.3 we assume that non of the possible witnesses requested by an existential constraint contradicts a universal constraint.

We construct a LODA $\mathcal{A} = (\mathcal{B}, \mathcal{C})$ such that $\mathcal{A}$ accepts an ordered data word over Σ if and only if C has a finite $(S_1, S_2, <_2)$-solution. Intuitively the automaton $\mathcal{A}$ interprets linearly ordered data words as extensions of ordered $(S_1, S_2, <_2)$-structures by unary relations but with no binary relations. In order to accept a word, it has to verify that binary T-types can be assigned to all pairs of positions in a way consistent with C.

The LODA $\mathcal{A}$ is very similar to the finite state automaton constructed in Lemma 3.4. Like the automaton in Lemma 3.4, the LODA $\mathcal{A}$ assumes that every position is labeled with its required witnesses (E).

(E) We assume that for every existential constraint (σ, E), all σ-labeled positions a are labeled with a fresh label $(\sigma, d, \gamma, \tau)$ such that $(d, \gamma, \tau) \in E$. The intention is that the (σ, E)-witness of a satisfies (d, γ, τ).

Dealing with positions that are close to each other can be done in a similar same way as in Lemma 3.4, except that now *close* means either close with respect to S_1 or close with respect to S_2 (or both). The existence of an assignment of binary T-types for positions that are close with respect to S_1 is verified by $\mathcal{B}$, and it is verified by $\mathcal{C}$ for positions close with respect to S_2.

More precisely, the automaton guesses the types of pairs of elements that are S_1- or S_2-close to each other. Each element a has at most four elements that are close to it: there might be an element b_1 with $S_1(a, b_1)$, an element b_2 with $S_1(b_2, a)$ and similarly elements b_3 with $S_2(a, b_3)$ and b_4 with $S_2(b_4, a)$. Some of those elements might not exist (if a is the first or last element with respect to $<_1$ or $<_2$) or might coincide (e.g. b_1 might coincide with either b_3 or b_4). For each of the elements $b_1, \ldots, b_4$ the automaton can guess and verify the binary T-type.

(L1) (Local types) We assume that each element a is labeled by up to four labels $\gamma_1, \ldots, \gamma_4 \in \Gamma$. The intention is that γ_i is the binary type of (a, b_i) (if the element b_i exists).
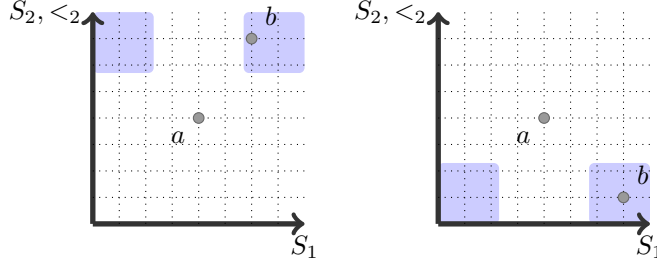
Figure 3: Illustration of remote elements in Lemma 5.4. On the left, the element a and all potential elements b in the highlighted area satisfy $d(x,y) \stackrel{\text{def}}{=} \neg S_1(x,y) \wedge \neg S_1(y,x) \wedge x <_2 y \wedge \neg S_2(x,y)$. On the right, the symmetric constraint $\neg S_1(x,y) \wedge \neg S_1(y,x) \wedge y <_2 x \wedge \neg S_2(y,x)$ is illustrated.

(L2) (Consistency of local types) The automaton verifies that the labels are consistent, that is, e.g., that if a and b are elements with $S_1(a,b)$ then the label γ_1 of a (i.e. the type guessed for (a,b)) is compatible with the label γ_2 of b (i.e. the type guessed for (b,a)) and that if $S_1(a,b)$ and $S_2(a,b)$ then $\gamma_1 = \gamma_3$.

(L3) (Local witnesses) For every σ-labeled element a that is labeled by $(\sigma, d, \gamma, \tau)$ due to (E), the automaton verifies that if $d = S_1(x,y)$ then the label γ_1 of a is γ and that its successor is labeled with τ. Likewise for $S_1(y,x), S_2(x,y)$, and $S_2(y,x)$).

(L4) (Local universal constraints) For all σ- and τ-labeled positions a and b with $S_1(a,b)$ the automaton verifies that if a is labeled γ_1 then there is no universal constraint $(\sigma, S(x,y), \gamma_1, \tau)$. Likewise for $S_1(b,a)$, $S_2(a,b)$, and $S_2(b,a)$.

For verifying the existence of an assignment of binary types for positions that are remote from each other, the LODA $\mathcal{A}$ has to do slightly more than the finite state automaton from Lemma 3.4. There is, essentially, only one $(S_1, S_2, <_2)$-type of remote positions which we will denote by $d \stackrel{\text{def}}{=} \neg S_1(x,y) \wedge \neg S_1(y,x) \wedge x <_2 y \wedge \neg S_2(x,y)$ (see Figure 3).

Checking that for all positions a and b at least one binary T-type is consistent with the universal constraints can be done as follows:

(U1) Assume that each position a is labeled by a set $L_{<_2}(a)$ containing all $\tau \in \Sigma$ such that there is a τ-labeled position b_1 such that (a, b_1) satisfies d and by a set $L_{>_2}(a)$ containing all $\tau \in \Sigma$ such that there is a τ-labeled position b_2 such that (b_2, a) satisfies d. Those labels can be easily verified by $\mathcal{B}$ and $\mathcal{C}$. (E.g. if $\tau \in L_{<_2}(a)$ for a position a and there are $i \in \{0,1,2\}$ many τ-labeled S_1-close elements b such that (a,b) satisfies $x <_2 y \wedge \neg S_2(x,y)$ then $\mathcal{C}$ verifies that there are $i+1$ τ-labeled elements b' such that (a,b') satisfies $x <_2 y \wedge \neg S_2(x,y)$. Then (a,b') satisfies d for one of those b'. Similarly for $\tau \notin L_{<_2}(a)$ and $L_{>_2}(a)$.)

(U2) For all σ-labeled a the automaton verifies that for every $\tau \in L_{<_2}(a)$ there

is a γ such that there is no universal constraint $(\sigma, d, \gamma, \tau)$ in C. Similarly for $L_{>_2}(a)$.

Testing that existential witnesses can be assigned to far-away positions is analogous to Lemma 3.4. The intuition is that pairs (a, b) of elements satisfying d behave like pairs (a', b') satisfying $x < y \wedge \neg S(x, y)$ in Lemma 3.4. A little care is needed for ensuring that elements are indeed remote from each other.

We shortly discuss some details of how $\mathcal{A}$ verifies that binary T-types for all σ- and τ-labeled elements a and b satisfying d can be assigned. To this end, the automaton $\mathcal{A}$ guesses whether there is an element u such that σ occurs more than $k \stackrel{\text{def}}{=} 3(|\Gamma| + 5)$ times above u and τ occurs more than k times below u. (Recall that a position a is above u if $a >_2 u$.)

If such a position u exists then the automaton tests that every σ-labeled element a has as many τ-labeled elements b such that (a, b) satisfies d as is required by (E), and that every τ-labeled element a' has sufficiently many σ-labeled positions b' such that (a', b') satisfies $\bar{d}$. If this is the case then binary types can be assigned as in Lemma 3.4 using Otto's assignment technique.

For testing that there are, say m many, τ-labeled elements b such that (a, b) satisfies d the automaton $\mathcal{B}$ labels a with the number $i \in \{0, 1, 2\}$ of τ-labeled positions that are S_1-close to a and S_2-remote from a. The automaton $\mathcal{C}$ then tests that there are at least $m + i$ many τ-labeled elements b such that (a, b) satisfies $x <_2 y \wedge S_2(x, y)$.

If there is no such position u then there is a position v such that there are at most $k + 1$ σ-labeled positions below v and at most $k + 1$ τ-labeled positions above v by Lemma 3.2. The automaton $\mathcal{A}$ exploits this structure exactly as in Lemma 3.4 by labeling those up to $k + 1$ many positions distinctly and guessing and verifying their witnesses.

For completeness we describe $\mathcal{A}$ in more detail, even though it is very similar to the automaton in Lemma 3.4. The automaton $\mathcal{A}$ does the following:

(E1) It guesses whether there is such a position u. The correctness of the guess can be easily verified by $\mathcal{C}$.

(E2) If there is such a position u then $\mathcal{A}$ verifies the following for all $\sigma, \tau \in \Sigma$:

  (a) If a σ-labeled position a is labeled by $(\sigma, d, \gamma_1, \tau), \ldots, (\sigma, d, \gamma_m, \tau)$ then there are m τ-labeled positions above a that are neither S_1- nor S_2-close to a.

  (b) Symmetrically, if a τ-labeled position a is labeled by $(\tau, \bar{d}, \gamma_1, \sigma), \ldots, (\tau, d, \gamma_m, \sigma)$ then there are m σ-labeled positions below a that are neither S_1- nor S_2-close to a.

(E3) If there is no such position u then $\mathcal{A}$ guesses the position v. The correctness of the guess can be easily verified by $\mathcal{C}$. Then:

  (a) We assume that positions are labeled by the following extra information (using fresh labels depending on σ and τ):

    • The ith σ-labeled position a below v is labeled by σ_i. The ith τ-labeled position a above v is labeled by τ_i. (By the choice of v there are at most $k + 1$ such σ_i and τ_i.)

    • The intended witnesses for σ_i- and τ_i-labeled positions are labeled as follows:

- If the σ_i-labeled position a is labeled by $(\sigma, \gamma, d, \tau)$ in (E), then there is a τ-labeled position b labeled with $(\sigma_i, \gamma, d, \tau)$ such that (a, b) satisfies d.
- Likewise, if the τ_i-labeled position a is labeled by $(\tau, \gamma, \bar{d}, \sigma)$ in (E), then there is a σ-labeled position b that is labeled with $(\tau_i, \gamma, \bar{d}, \sigma)$ such that (a, b) satisfies $\bar{d}$.

- Positions with intended witnesses from σ_i- and τ_i-labeled positions are labeled as follows:
  - Each position a below v that is labeled by $(\tau, \gamma, \bar{d}, \sigma)$ in (E) is also labeled with $(\tau, \gamma, \bar{d}, \sigma_i)$ for some i such that (a, b) satisfies $\bar{d}(x, y)$ where b is the σ_i-labeled position.
  - Likewise, each $(\sigma, \gamma, d, \tau)$-labeled position a above u is labeled with $(\sigma, \gamma, d, \tau_i)$ for some i such that (a, b) satisfies d where b is the σ_i-labeled position.

(b) The automaton verifies that the labels are consistent, that is:

- No τ-labeled position below v is labeled with $(\sigma_i, \gamma, d, \tau)$ and with $(\tau, \gamma', \bar{d}, \sigma_i)$ where γ and γ' are not reverse types.
- Likewise, no σ-labeled position above v is labeled with $(\tau_i, \bar{\gamma}', d, \sigma)$ and with $(\sigma, \gamma, d, \tau_i)$ where γ and γ' are not reverse types.

The correctness is proved completely analogous to Lemma 3.4. $\qquad\square$

6 Order- and Successor-invariance

In this Section we discuss order-invariance for two-variable formulas. A first-order sentence φ over a signature $T \cup \{<\}$ is *order-invariant* (*<-invariant* for short) if for each T-structure $\mathcal{A}$ and all linear orders $<_1$ and $<_2$ on the domain of $\mathcal{A}$,

$$(\mathcal{A}, <_1) \models \varphi \iff (\mathcal{A}, <_2) \models \varphi.$$

A class of finite T-structures $\mathcal{A}$ is *<-invariantly first-order definable* if there is an *<*-invariant $T \cup \{<\}$-sentence φ such that, for each finite T-structure $\mathfrak{A}$,

$$\mathfrak{A} \in \mathcal{C} \text{ if and only } (\mathfrak{A}, <) \models \varphi$$

for each linear order $<$ on the domain of $\mathfrak{A}$.

It is not immediately obvious that allowing this restricted use of an order extends the expressive power of FO. A well-known example due to Gurevich (cf. eg. [16]) shows that there is indeed a $<$-invariantly definable class of structures which is not FO-definable without an order. The example of Gurevich indicates that order-invariance is, at least potentially, useful in formulating queries. For using invariance in this context, it is essential to be able to verify that an FO-sentence is indeed invariant. Unfortunately, a simple reduction of the finite satisfiability problem shows that $<$-invariance of FO-sentences is undecidable.

The results from the previous sections as well as the discussion in the introduction imply that $<$-invariance of FO^2-sentences is decidable.

Theorem 6.1. *Order-invariance of* FO^2 *is in* 2-NExpTime.

Successor-invariance (*S-invariance*, for short) is defined analogously to order-invariance where instead of the linear order the formulas may use a successor relation (see e.g. [22]). By combining the approach discussed in the introduction with Theorem 2 of [3], one obtains the following result.

Theorem 6.2. *Successor-invariance of* FO^2 *is in* NExpTime.

Our approach for deciding $<$-invariance and S-invariance does not immediately transfer to $(S, <)$-invariance where both the order and its induced successor can be used in formulas, since even $EMSO^2$ is undecidable on finite $(S_1, <_1, S_2, <_2)$-structures [17] However, in the proof of Theorem 6.1, it would suffice to prove decidability for formulas of the form $\varphi(S_1, <_1) \wedge \neg\varphi(S_2, <_2)$. This seems conceivable, but so far we were unable to obtain a proof using our techniques.

The following example shows that invariance is useful even for two-variable logic.

Example 6.3. FO^2 is often extended by *counting quantifiers* of the shape $\exists^{\geq k}$, where $\exists^{\geq k} x \, \varphi$ states that there are at least k satisfying assignments to x. We define *monadic counting quantifiers* which are defined in the same way but φ is restricted to use at most one free variable x. We observe that these quantifiers are $<$-invariantly definable by two-variable formulas, since

$$
\begin{aligned}
\exists^{\geq 0} x \, \varphi(x) &\equiv x = x, \\
\exists^{\geq k+1} x \, \varphi(x) &\equiv \exists x \, \varphi(x) \wedge \exists^{\geq k} y \; y < x \wedge \varphi(y).
\end{aligned}
$$

It is easy to see using a pebble game argument that the monadic counting quantifier $\exists^{\leq k}$ is not two-variable definable for each $k \geq 3$.

7 Conclusion and Future Work

We have shown that $<$-invariance and S-invariance of FO^2-sentences is decidable by establishing decidability of ESO^2-satisfiability on finite $(S_1, <_1, <_2)$- and $(S_1, S_2, <_2)$-structures. Several interesting questions remain open:

1. Is ESO^2 decidable on (S_1, S_2, S_3)-structures?

2. Where is the border of decidability of ESO^2 on general, not necessarily finite, ordered structures?

3. Is $(S, <)$-invariance of FO^2 decidable?

4. Is every $<$-invariantly FO^2-definable property also FO-definable without the use of a linear order?

For the second question preliminary results have been obtained for certain order types.

Acknowledgements

References

[1] Michael A. Benedikt and Luc Segoufin. Towards a characterization of order-invariant queries over tame graphs. *J. Symbolic Logic*, 74(1):168–186, 03 2009.

[2] Mikolaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data words. *ACM Trans. Comput. Logic*, 12(4):27:1–27:26, July 2011.

[3] Witold Charatonik and Piotr Witkowski. Two-variable logic with counting and trees. In *LICS'13*, pages 73–82, 2013.

[4] Witold Charatonik and Piotr Witkowski. Two-variable logic with counting and a linear order. In *CSL 2015*, volume 41 of *LIPIcs*, pages 631–647, 2015.

[5] Kord Eickmeyer, Michael Elberfeld, and Frederik Harwath. Expressivity and succinctness of order-invariant logics on depth-bounded structures. In *MFCS 2014*, pages 256–266. Springer, 2014.

[6] Thomas Eiter, Yuri Gurevich, and Georg Gottlob. Existential second-order logic over strings. *J. ACM*, 47(1):77–131, 2000.

[7] Kousha Etessami, Moshe Y. Vardi, and Thomas Wilke. First-order logic with two variables and unary temporal logic. *Inf. Comput.*, 179(2):279–295, 2002.

[8] Erich Grädel, Phokion G. Kolaitis, and Moshe Y. Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3(1):53–69, 1997.

[9] Erich Grädel and Martin Otto. On logics with two variables. *Theor. Comput. Sci.*, 224(1-2):73–113, 1999.

[10] Erich Grädel and Eric Rosen. Two-variable descriptions of regularity. In *LICS '99*, pages 14–23, 1999.

[11] Martin Grohe and Thomas Schwentick. Locality of order-invariant first-order formulas. *ACM Trans. Comput. Log.*, 1(1):112–130, 2000.

[12] Emanuel Kieronski. Decidability issues for two-variable logics with several linear orders. In *CSL 2011*, volume 12 of *LIPIcs*, pages 337–351, 2011.

[13] Emanuel Kieronski, Jakub Michaliszyn, Ian Pratt-Hartmann, and Lidia Tendera. Two-variable first-order logic with equivalence closure. *SIAM J. Comput.*, 43(3):1012–1063, 2014.

[14] Emanuel Kieronski and Martin Otto. Small substructures and decidability issues for first-order logic with two variables. *J. Symb. Log.*, 77(3):729–765, 2012.

[15] Emanuel Kieronski and Lidia Tendera. On finite satisfiability of two-variable first-order logic with equivalence relations. In *LICS 2009*, pages 123–132. IEEE Computer Society, 2009.

[16] L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[17] Amaldev Manuel. Two variables and two successors. In *MFCS 2010*, volume 6281, pages 513–524. Springer, 2010.

[18] Amaldev Manuel and Thomas Zeume. Two-variable logic on 2-dimensional structures. In *CSL 2013*, volume 23 of *LIPIcs*, pages 484–499, 2013.

[19] Maarten Marx. First order paths in ordered trees. In *ICDT 2005*, volume 3363, pages 114–128. Springer, 2005.

[20] Michael Mortimer. On languages with two variables. *Zeitschr. f. math. Logik u. Grundlagen d. Math.*, 21:135–140, 1975.

[21] Martin Otto. Two variable first-order logic over ordered domains. *J. Symb. Log.*, 66(2):685–702, 2001.

[22] Benjamin Rossman. Successor-invariance in the finite. In *LICS 2003*, pages 148–157. IEEE Computer Society, 2003.

[23] Nicole Schweikardt. A short tutorial on order-invariant first-order logic. In *CSR 2013*, volume 7913, pages 112–126. Springer, 2013.

[24] Thomas Schwentick and Thomas Zeume. Two-variable logic with two order relations. In *CSL 2015*, volume 6247, pages 499–513, 2010.

[25] Thomas Schwentick and Thomas Zeume. Two-variable logic with two order relations. *Logical Methods in Computer Science*, 8(1), 2012.

[26] Dana Scott. A decision method for validity of sentences in two variables. *Journal of Symbolic Logic*, 27(377):74, 1962.

[27] Wieslaw Szwast and Lidia Tendera. FO^2 with one transitive relation is decidable. In *STACS 2013*, volume 20 of *LIPIcs*, pages 317–328, 2013.