Two-variable Logic with Counting and a Linear Order

Witold Charatonik and Piotr Witkowski

Institute of Computer Science
University of Wroclaw, Poland
{wch,pwit}@cs.uni.wroc.pl

Abstract

We study the finite satisfiability problem for the two-variable fragment of the first-order logic extended with counting quantifiers (C^2) and interpreted over linearly ordered structures. We show that the problem is undecidable in the case of two linear orders (in presence of two other binary symbols). In the case of one linear order it is NExpTime-complete, even in presence of the successor relation. Surprisingly, the complexity of the problem explodes when we add one binary symbol more: C^2 with one linear order and its successor, in presence of other binary predicate symbols, is decidable, but it is as expressive (and as complex) as Vector Addition Systems.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Two-variable logic, counting quantifiers, linear order, satisfiability, complexity

Digital Object Identifier 10.4230/LIPIcs.CSL.2015.631

1 Introduction

Since 1930s, when Alonzo Church and Alan Turing proved that the satisfiability problem for first-order logic is undecidable, much effort was put to find decidable subclasses of this logic. One of the most prominent decidable cases is the two-variable fragment FO^2. FO^2 is particularly important in computer science because of its decidability and connections with other formalisms like modal, temporal or description logics or applications in XML or ontology reasoning. The satisfiability of FO^2 was proved to be decidable in [31, 23] and NExpTime-complete in [8].

All decidable fragments of first-order logic have a limited expressive power and a lot of effort is being put to extend them beyond the first-order logic while preserving decidability. Many extensions of FO^2, in particular with transitive closure or least fixed-point operators, quickly lead to undecidability [7, 11]. Extensions that go beyond the first order logic and enjoy decidable finite satisfiability problem include FO^2 over restricted classes of structures where one [15] or two relation symbols [16] are interpreted as equivalence relations; where one [25] or two relations are interpreted as linear orders [30]; where two relations are interpreted as successors of two linear orders [20, 6, 4]; where one relation is interpreted as linear order and another one as equivalence [1]; where one relation is transitive [33]; where an equivalence closure can be applied to two binary predicates [14]; where deterministic transitive closure can be applied to one binary relation [3]. It is known that the finite satisfiability problem is undecidable for FO^2 with two transitive relations [13], with three equivalence relations [15], with one transitive and one equivalence relation [16], with three linear orders [12], with two linear orders and their two corresponding successors [20]. A summary of complexity results for extensions of FO^2 with order relations can be found in [21].

24th EACSL Annual Conference on Computer Science Logic (CSL 2015).
Editor: Stephan Kreutzer; pp. 631–647

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The two-variable fragment with counting quantifiers (C^2) extends FO^2 by allowing counting quantifiers of the form $\exists^{<k}$, $\exists^{\leq k}$, $\exists^{=k}$, $\exists^{\geq k}$ and $\exists^{>k}$, for all natural numbers k. The two problems of satisfiability and finite satisfiability for C^2 (which are two different problems as C^2 does not have a finite model property) were proved to be decidable in [10]. Another solution of the satisfiability problem together with NExpTime-completeness result under unary encoding of numbers in counting quantifiers can be found in [26]. Pratt-Hartmann in [27] established NExpTime-completeness of both satisfiability and finite satisfiability under binary encoding of numbers in counting quantifiers. All these algorithms are quite sophisticated, a significant simplification can be found in [28]. There are not many known decidable extensions of C^2. In [4] it is shown that finite satisfiability for C^2 interpreted over structures where two binary relations are interpreted as forests of finite trees (which subsumes the case of two successor relations on two linear orders) is NExpTime-complete. [29] shows that the satisfiability and finite satisfiability problems for C^2 with one equivalence relation are both NExpTime-complete.

In this paper we study the extensions of C^2 with linear orders. We show that the finite satisfiability problem for C^2 with two linear orders, in presence other binary predicate symbols, is undecidable. For C^2 with one linear order, even if the successor of this linear order is present, in absence of other binary predicate symbols, it is decidable and NExpTime-complete. A surprising result is that when we add one more binary predicate symbol, the complexity of the problem explodes: C^2 with one linear order and its successor, in presence of other binary predicate symbols, is as expressive (and as complex) as multicounter automata.

Multicounter automata (MCA) is a very simple formalism equivalent to Petri Nets and vector addition systems (VAS) [24], which are used e. g., to describe distributed, concurrent systems and chemical/biological processes. One of the main reasoning tasks for VAS is to determine reachability of a given vector. It is known that this problem is decidable [17, 22, 18] and ExpSpace-hard [19], but precise complexity is not known, and after over 40 years of research it is even not known if the problem is elementary. We give a reduction from the emptiness problem of MCA (which is equivalent to the reachability for VAS) to finite satisfiability of C^2 with one linear order and its successor, in presence of one more binary predicate symbol. Although we show that C^2 with one linear order and its successor, in presence of arbitrary number of binary predicate symbols is decidable, it is very unlikely that it has an elementary decision algorithm since existence of such an algorithm implies existence of an elementary algorithm for VAS reachability.

Due to space limits we have omitted most proofs. The missing proofs can be found in the full version of the paper.

2 Preliminaries

We will consider finite satisfiability problems for the two-variable logic with counting (C^2 for short) over finite structures, where some binary symbols are interpreted as linear orders or successors of linear orders. W.l.o.g. we will be interested in largest antireflexive relations $<$ contained in linear orders $\leq$; for a linear order $\leq$ we write $a < b$ iff $a \leq b$ and $a \neq b$. We overload notation and name these relations $<$ linear orders. We use symbols $<$, $<_1$, $<_2$ to denote linear orders and $\dashv$, $\dashv_1$, $\dashv_2$ to denote their resp. successors. Given a finite vocabulary Σ we write $\mathcal{O}(\Sigma, <, \dashv)$ to denote the class of finite structures on vocabulary Σ, where $<$ and $\dashv$ have appropriate interpretation, and we adopt a similar notation for other classes of structures. Logics we consider will be denoted by $C^2[U, B, I]$, where U and B are vocabularies of resp. unary and binary symbols allowed in formulas, and $I \subseteq B$ is

the vocabulary of interpreted binary symbols. When $B = I$ we write $\mathrm{C}^2[U, I]$ instead of $\mathrm{C}^2[U, I, I]$.

Let Σ_u and Σ_b be countably infinite vocabularies of resp. unary and binary symbols and such that $\{<, +1, <_1, +1_1, <_2, +1_2, =\} \subseteq \Sigma_b$. Specifically, we will be interested in the following logics: $\mathrm{C}^2[\Sigma_u, \Sigma_b, \{<\}]$, $\mathrm{C}^2[\Sigma_u, \Sigma_b, \{<, +1\}]$ and $\mathrm{C}^2[\Sigma_u, \{<_1, s_1, <_2, s_2\}, \{<_1, <_2\}]$, where s_1 and s_2 are some binary symbols. By $\mathrm{C}^2[\Sigma_u, \Sigma_b, \{<\}]$ we mean the logic C^2 where vocabulary of every formula is a finite subset of $\Sigma_u \cup \Sigma_b$ and $<$ is interpreted as a linear order. Definition of $\mathrm{C}^2[\Sigma_u, \Sigma_b, \{<, +1\}]$ is similar, with the exception that $+1$ is interpreted as successor of $<$. The logic $\mathrm{C}^2[\Sigma_u, \{<_1, s_1, <_2, s_2\}, \{<_1, <_2\}]$ allows arbitrary number of unary and at most 4 binary symbols, two of them are interpreted as linear orders. Notice that we do not allow constant symbols in vocabularies, but this does not cause loss of generality since constants can be simulated by unary predicates and counting quantifiers.

3 Two linear orders

We start with an observation that the successor relation of a linear order can be expressed in $\mathrm{C}^2[\Sigma_u, \Sigma_b, \{<\}]$. More precisely, let s be a free binary symbol. The following lemma says that s can be defined to mean the successor of $<$ in $\mathrm{C}^2[\Sigma_u, \{<, s\}, \{<\}]$. Intuitively, it is enough to express that s is a subrelation of $<$ such that each node (with the exception of the least and the greatest one) has exactly one s-successor and exactly one s-predecessor.

▶ **Lemma 1.** *There exists a formula φ_s of $\mathrm{C}^2[\Sigma_u, \{<, s\}, \{<\}]$ such that for every finite structure $\mathcal{M}$, we have $\mathcal{M} \models \varphi_s$ if and only if $s^{\mathcal{M}}$ is the successor relation of $<^{\mathcal{M}}$.*

▶ **Corollary 2.** *Finite satisfiability of $\mathrm{C}^2[\Sigma_u, \Sigma_b, \{<, +1\}]$ is reducible in constant time to finite satisfiability of $\mathrm{C}^2[\Sigma_u, \Sigma_b, \{<\}]$.*

▶ **Corollary 3.** *Finite satisfiability of $\mathrm{C}^2[\Sigma_u, \{<_1, +1_1, <_2, +1_2\}, \{<_1, +1_1, <_2, +1_2\}]$ is reducible in constant time to finite satisfiability of $\mathrm{C}^2[\Sigma_u, \{<_1, s_1, <_2, s_2\}, \{<_1, <_2\}]$, where s_1 and s_2 are some binary symbols.*

Since $\mathrm{FO}^2[\Sigma_u, \{<_1, +1_1, <_2, +1_2\}, \{<_1, +1_1, <_2, +1_2\}]$, i. e., the two-variable logic with two linear orders and their corresponding successors, is undecidable [20], we have the following conclusion.

▶ **Corollary 4.** *Finite satisfiability problem of $\mathrm{C}^2[\Sigma_u, \Sigma_b, \{<_1, <_2\}]$ is undecidable. This remains true even for $\mathrm{C}^2[\Sigma_u, \{<_1, s_1, <_2, s_2\}, \{<_1, <_2\}]$, where s_1 and s_2 are distinct binary symbols.*

4 $\mathrm{C}^2[\Sigma_u, \{<, +1\}, \{<, +1\}]$ is NExpTime-complete

We will show that finite satisfiability problem for $\mathrm{C}^2[\Sigma_u, \{<, +1\}, \{<, +1\}]$ is NExpTime-complete. Since the lower bound follows from the complexity of FO^2 with only unary predicates [5, Theorem 11], we will concentrate on proving the upper bound. The proof presented here is similar to a corresponding result [2] on FO^2 on finite trees.

We assume that the input $\mathrm{C}^2[\Sigma_u, \{<, +1\}, \{<, +1\}]$ formula φ is in a normal form

$$\varphi = \forall x \forall y.\chi(x, y) \wedge \bigwedge_{h=1}^{m} \forall x \exists^{\preccurlyeq_h C_h} y.\chi_h(x, y),$$

where $\chi, \chi_1, \ldots, \chi_m$ are quantifier-free formulas with arbitrary unary predicates and binary predicates $<, +1$, symbols $\preccurlyeq_h \in \{\leq, \geq\}$ for $h = \{1, \ldots, m\}$, and $C_1, \ldots, C_m$ are positive

integers encoded in binary. For the rest of this section the constant c is fixed and it equals $\max\{C_h \mid h \in \{1,\ldots,m\}\}$. It is well known [9, Theorem 2.2] that by adding additional unary predicates each C^2 formula φ can be transformed in polynomial time to an equisatisfiable formula in normal form.

Observe that $\mathrm{C}^2[\Sigma_u, \{<, +1\}, \{<, +1\}]$ may be seen as a fragment of the weak monadic second-order logic with one successor WS1S, where unary relations are simulated by second-order existential quantifiers and counting quantifiers by first-order ones (e.g., a formula of the form $\exists^{\leq k} x.\chi(x)$ can be replaced by an equivalent formula with $k+1$ universal quantifiers). However, this view leads to formulas with three alternations of quantifiers that can be checked for satisfiability in 4ExpTime, which is not a desired complexity bound.

Because an element of a model of φ may require up to c witnesses for satisfaction, we will be interested in multisets counting these witnesses. Let $\mathbb{N}_c = \{n \in \mathbb{N} \mid n \leq c\} \cup \{\infty\}$. For $k, k' \in \mathbb{N}_c$ define $\mathrm{cut}_c(k) = k$ if $k \leq c$ and $\mathrm{cut}_c(k) = \infty$ if $k > c$. Define $k \oplus_c k' = \mathrm{cut}_c(k + k')$. A c-multiset of elements from a given set A is any function $f : A \to \mathbb{N}_c$. For a given element a in A, by $\{a\}$ we denote the multiset defined by $\{a\}(x) = 1$ if $x = a$ and $\{a\}(x) = 0$ for $x \neq a$. The union of two multisets f and g is a function denoted $f \cup g$ such that $(f \cup g)(x) = f(x) \oplus_c g(x)$. Empty multiset denoted $\emptyset$ is the constant function equal 0 for all arguments.

Let us call maximal consistent formulas specifying the relative position of a pair of nodes in a structure in $\mathcal{O}(\Sigma, <, +1)$ *order formulas*. There are five possible order formulas: $x{=}y \wedge \neg +1(y,x) \wedge \neg +1(x,y) \wedge y{\not<}x \wedge x{\not<}y$, $x{\neq}y \wedge +1(y,x) \wedge \neg +1(x,y) \wedge y{<}x \wedge x{\not<}y$, $x{\neq}y \wedge +1(x,y) \wedge \neg +1(y,x) \wedge x{<}y \wedge y{\not<}x$, $x{\neq}y \wedge \neg +1(x,y) \wedge \neg +1(y,x) \wedge x{<}y \wedge y{\not<}x$, and $x{\neq}y \wedge \neg +1(x,y) \wedge \neg +1(y,x) \wedge y{<}x \wedge x{\not<}y$. They are denoted, respectively, as: $\theta_=$, θ_{-1}, θ_{+1}, $\theta_<$, $\theta_>$. Let Θ be the set of these five formulas.

A *1-type* over the signature Σ is a maximal consistent conjunction of atomic and negated atomic formulas over Σ involving only the variable x. The set of all 1-types over Σ will be denoted $\Pi(\Sigma)$. The family of all multisets of 1-types over the signature Σ is denoted $\mathbb{N}_c^{\Pi(\Sigma)}$.

▶ **Definition 5** (Full type over Σ w.r.t. c)**.** A *full type* over Σ w.r.t. c is a function $\sigma : \Theta \to \mathbb{N}_c^{\Pi(\Sigma)}$, such that $\sigma(\theta_{-1})$ and $\sigma(\theta_{+1})$ are singletons or empty, and $\sigma(\theta_=)$ is a singleton.

▶ **Definition 6** (Full type in $\mathcal{A}$ w.r.t. c)**.** Let $\mathcal{A}$ be a structure over a vocabulary Σ and let a be an element of $\mathcal{A}$. A *full type* of a in $\mathcal{A}$, denoted $\mathrm{ft}^{\mathcal{A}}(a)$ is a function $\sigma : \Theta \to \mathbb{N}_c^{\Pi(\Sigma)}$ such that

- $\sigma(\theta_=)$ is the singleton of the 1-type of a in $\mathcal{A}$,
- $\sigma(\theta_{-1})$ is the singleton of the 1-type of the predecessor of a (if a has a predecessor) or empty multiset (if a has no predecessor),
- $\sigma(\theta_{+1})$ is the singleton of the 1-type of the successor of a (if a has a successor) or empty multiset (if a has no successor),
- $\sigma(\theta_<)$ is the c-multiset of 1-types of elements strictly smaller than a in $\mathcal{A}$, excluding the predecessor (if it exists), and
- $\sigma(\theta_>)$ is the c-multiset of 1-types of elements strictly greater than a in $\mathcal{A}$, excluding the successor (if it exists).

A structure $\mathcal{A}$ is said to realise a full type σ if $\mathrm{ft}^{\mathcal{A}}(a) = \sigma$ for some $a \in \mathcal{A}$.

In the following, we often identify a full type σ, which is a function, with the tuple $\langle \sigma(\theta_{-1}), \sigma(\theta_=), \sigma(\theta_{+1}), \sigma(\theta_<), \sigma(\theta_>)\rangle$. We define Σ-c-graph as the graph $\langle V, E\rangle$ where the set V of nodes is the set of full-types over Σ w.r.t. c and the set E of edges is defined as follows.

$$\langle\langle \Pi_{-1}, \{\pi\}, \{\pi_{+1}\}, \Pi_<, \Pi_>\rangle, \langle \{\pi\}, \{\pi_{+1}\}, \Pi'_{+1}, \Pi'_<, \Pi'_>\rangle\rangle \in E \quad \text{iff} \quad \Pi'_< = \Pi_< \cup \Pi_{-1} \text{ and } \Pi_> = \Pi'_> \cup \Pi'_{+1}$$

Let σ be a full type such that $\sigma(\theta_=) = \{\pi\}$ and let $\forall x \exists^{\leqslant_h C_h} y.\chi_h(x,y)$ be a conjunct in φ. The following five functions are used to count witnesses w.r.t. this conjunct for elements of full type σ.

$$W^{\chi_h}_{=}(\sigma) = \begin{cases} 1 & \text{if } \pi(x) \models \chi_h(x,x) \\ 0 & \text{otherwise} \end{cases}$$

$$W^{\chi_h}_{-1}(\sigma) = \begin{cases} 1 & \text{if } \sigma(\theta_{-1}) = \{\pi'\} \text{ and } \pi(x) \wedge \pi'(y) \wedge \theta_{-1}(x,y) \models \chi_h(x,y) \\ 0 & \text{otherwise} \end{cases}$$

$$W^{\chi_h}_{+1}(\sigma) = \begin{cases} 1 & \text{if } \sigma(\theta_{+1}) = \{\pi'\} \text{ and } \pi(x) \wedge \pi'(y) \wedge \theta_{+1}(x,y) \models \chi_h(x,y) \\ 0 & \text{otherwise} \end{cases}$$

$$W^{\chi_h}_{<}(\sigma) = \mathrm{cut}_c\Big(\sum_{\pi':\pi(x)\wedge\pi'(y)\wedge\theta_<(x,y)\models\chi_h(x,y)} (\sigma(\theta_<))(\pi') \Big)$$

$$W^{\chi_h}_{>}(\sigma) = \mathrm{cut}_c\Big(\sum_{\pi':\pi(x)\wedge\pi'(y)\wedge\theta_>(x,y)\models\chi_h(x,y)} (\sigma(\theta_>))(\pi') \Big)$$

Note that in the definition above $(\sigma(\theta_>))(\pi')$ is simply the number of occurrences of the 1-type π' in the multiset $\sigma(\theta_>)$.

▶ **Definition 7** (Compatible full types)**.** We say that a full type σ such that $\sigma(\theta_=) = \{\pi\}$ is compatible with formula φ if the following conditions are satisfied.

- $\pi(x) \models \chi(x,x)$,
- $\pi(x) \wedge \pi'(y) \wedge \theta(x,y) \models \chi$ for all $\theta \in \{\theta_{-1}, \theta_{+1}, \theta_<, \theta_>\}$ and all $\pi' \in \sigma(\theta)$, and
- for each conjunct $\forall x \exists^{\leqslant_h C_h} y.\chi_h(x,y)$ of φ we have

$$W^{\chi_h}_{=}(\sigma) + W^{\chi_h}_{+1}(\sigma) + W^{\chi_h}_{-1}(\sigma) + W^{\chi_h}_{>}(\sigma) + W^{\chi_h}_{<}(\sigma) \leqslant_h C_h$$

It is quite obvious that whenever $\mathcal{A} \models \varphi$, all full types realised in $\mathcal{A}$ are compatible with φ. It is not difficult to see that the converse is also true, as the following lemma says.

▶ **Lemma 8.** *For any ordered structure $\mathcal{A}$ and any $\mathrm{C}^2[\Sigma_u, \{<, +1\}, \{<, +1\}]$ formula φ in normal form, if all full types realised in $\mathcal{A}$ are compatible with φ then $\mathcal{A} \models \varphi$.*

We define Σ-c-φ-graph as the subgraph of Σ-c-graph consisting of nodes compatible with φ. The nodes of the form $\langle \emptyset, \ldots, \ldots, \emptyset, \ldots \rangle$ are called *source* nodes; the nodes of the form $\langle \ldots, \ldots, \emptyset, \ldots, \emptyset \rangle$ are called *target* nodes. Intuitively, a source node corresponds to a full type of the least element in some model of φ while a target node corresponds to the greatest element in some model.

▶ **Lemma 9.** *Let φ be a $\mathrm{C}^2[\Sigma_u, \{<, +1\}, \{<, +1\}]$ formula in normal form over vocabulary Σ. Formula φ is finitely satisfiable if and only if there exists a path from a source node to a target node in Σ-c-φ-graph.*

Lemma 9 leads us directly to the main theorem of this section. To check satisfiability of a formula in $\mathrm{C}^2[\Sigma_u, \{<, +1\}, \{<, +1\}]$ it is enough to guess an appropriate path in Σ-c-φ-graph. Moreover, it is enough to use only exponentially many different full types in the guessed path.

▶ **Theorem 10.** *The finite satisfiability problem for $\mathrm{C}^2[\Sigma_u, \{<, +1\}, \{<, +1\}]$ is* NExpTime*-complete.*

Proof. The lower bound follows from the complexity of FO^2 with only unary predicates. For the upper bound, an algorithm for deciding finite satisfiability of $\mathrm{C}^2[\Sigma_u, \{<, +1\}, \{<, +1\}]$ works as follows. We take a $\mathrm{C}^2[\Sigma_u, \{<, +1\}, \{<, +1\}]$ formula φ and convert it to an equisatisfiable normal form (in polynomial time) if necessary. Then we guess a path from a source node to a target node in Σ-c-φ-graph where Σ is the vocabulary of φ. This requires in particular verification of the fact that all nodes are compatible with φ. All this can be accomplished in time polynomial in the size of the graph. This size is potentially doubly exponential in $|\varphi|$: the number of all 1-types over Σ is exponential in $|\varphi|$, so the number of sets of 1-types, and, in consequence, the number of full types, is doubly exponential. The potential 2NExpTime complexity of the algorithm can be lowered to NExpTime using the observation that the $\theta_<$ and $\theta_>$ components of full types behave in a monotone way along any path connecting any source node with any target node. The $\theta_<$ component may only increase and $\theta_>$ only decrease along any such path. Since a multiset may increase only exponentially many times (and only exponentially many times it may decrease) there are only exponentially many such multisets occurring along the path. Therefore it is enough to guess only exponentially many different full types. ◀

5 Hardness of $\mathrm{C}^2[\Sigma_u, \{<, +1, s\}, \{<, +1\}]$

We will show that finite satisfiability problem for $\mathrm{C}^2[\Sigma_u, \{<, +1, s\}, \{<, +1\}]$, where s is a binary relation, is at least as hard as non-emptiness of multicounter automata. Below, for a given MCA M we construct a $\mathrm{C}^2[\Sigma_u, \{<, +1, s\}, \{<, +1\}]$ formula φ_M which has a finite model if and only if M is non-empty.

Multicounter automata

We adopt a notion of *multicounter automata* (MCA for short) similar to one in [32], but with empty input alphabet and simplified counter manipulation. Intuitively, a MCA is a finite state automaton without input but equipped with a finite set of counters which can be incremented and decremented, but not tested for zero. More formally, a multicounter automaton M is a tuple $\langle Q, C, R, \delta, q_I, F\rangle$, where the set Q of states, the initial state $q_I \in Q$ and the set $F \subseteq Q$ of final states are as in usual finite state automata, C is a finite set (the *counters*) and R is a subset of C. The transition relation δ is a subset of

$$Q \times \{inc(c), dec(c), skip \mid c \in C\} \times Q.$$

An MCA is called *reduced* if it does not have skip transitions and $R = C$ (in this case we just omit R component of tuple M).

A *configuration* of a multicounter automaton M is a pair $\langle p, \vec{n}\rangle$ where p is a state and $\vec{n} \in \mathbb{N}^C$ gives a value $\vec{n}(c)$ for each counter c in C. Transitions with $inc(c)$ and *skip* can always be applied, whereas transitions with $dec(c)$ can only be applied to configurations with $\vec{n}(c) > 0$. Applying a transition $\langle p, inc(c), q\rangle$ to a configuration $\langle p, \vec{n}\rangle$ yields a configuration $\langle q, \vec{n}_0\rangle$ where $\vec{n}_0$ is obtained from $\vec{n}$ by incrementing its c-th component and keeping values of all other components unchanged. Analogously, applying (an applicable) transition $\langle p, dec(c), q\rangle$ to a configuration $\langle p, \vec{n}\rangle$ yields a configuration $\langle q, \vec{n}_0\rangle$ where $\vec{n}_0$ is obtained from $\vec{n}$ by decrementing its c-th component. Transitions with *skip* do not change value of any counter in C. A *run* is an interleaving sequence of configurations and transitions $conf_1, trans_1, \ldots, trans_{k-1}, conf_k$ such that $trans_i$ applied to $conf_i$ gives $conf_{i+1}$, for $1 \leq i < k$. A run is *accepting*, if it starts in configuration $\langle q_I, \vec{0}\rangle$ and ends in some configuration $\langle q_F, \vec{n}_F\rangle$ with $q_F \in F$ and $\vec{n}_F(c) = 0$ for every $c \in R$. The emptiness problem for multicounter automata is the question whether

a given automaton M has an accepting run. It is well known that this problem (for both MCA and reduced MCA) is decidable, as it is polynomial-time equivalent to reachability problem in Vector Addition Systems/Petri Nets[17, 22].

▶ **Definition 11.** Let $M = \langle Q, C, \delta, q_I, F\rangle$ be a reduced MCA. Let $\Sigma = \{q \mid q \in Q\} \cup \{inc_c, dec_c \mid c \in C\} \cup \{\min, \max, <, +1, s\}$ where predicates $q, inc_c, dec_c, \min$ and max are unary and $<, +1$ and s are binary. Define φ_M as the conjunction of the following Σ-formulas.

1. $\exists^{=1}x.\, \min(x) \wedge \exists^{=1}x.\, \max(x)$
2. $\forall x \forall y.\, (\min(x) \to (x < y \vee x = y)) \wedge (\max(x) \to (y < x \vee y = x))$
3. $\forall x.\, \Big(\bigvee_{q\in Q} q(x)\Big) \wedge \bigwedge_{q\in Q} \Big(q(x) \to \bigwedge_{q'\in Q\setminus\{q\}} \neg q'(x)\Big)$
4. $\forall x.\, (\min(x) \to q_I(x)) \wedge \Big(\max(x) \to \bigvee_{q_F\in F} q_F(x)\Big)$
5. $\forall x \forall y.\, +1(x,y) \to$
   $\bigvee_{\langle q, inc(c), q'\rangle \in \delta} (q(x) \wedge inc_c(x) \wedge q'(x)) \;\vee\; \bigvee_{\langle q, dec(c), q'\rangle \in \delta} (q(x) \wedge dec_c(x) \wedge q'(x))$
6. $\forall x.\, (\neg \max(x)) \to \bigvee_{c\in C} (inc_c(x) \vee dec_c(x))$
7. $\forall x.\, \bigwedge_{c\in C} \Big(inc_c(x) \to \neg dec_c(x) \wedge \bigwedge_{c'\in C\setminus\{c\}} (\neg dec_{c'}(x) \wedge \neg inc_{c'}(x))\Big)$
8. $\forall x.\, \bigwedge_{c\in C} \Big(dec_c(x) \to \neg inc_c(x) \wedge \bigwedge_{c'\in C\setminus\{c\}} (\neg inc_{c'}(x) \wedge \neg dec_{c'}(x))\Big)$
9. $\forall x.\, (\max(x)) \to \bigwedge_{c\in C} (\neg inc_c(x) \wedge \neg dec_c(x))$
10. $\forall x \forall y. s(x,y) \to \bigvee_{c\in C} (inc_c(x) \wedge dec_c(y))$
11. $\forall x \forall y.\, (s(x,y) \to x < y)$
12. $\forall x.\, \big(\max(x) \vee \exists^{=1} y.\, (s(x,y) \vee s(y,x))\big)$

We will interpret φ_M as a $\mathrm{C}^2[\Sigma_u, \{<, +1, s\}, \{<, +1\}]$ formula. Models of φ_M encode accepting runs of MCA M. The first two conjuncts of φ_M define the meaning of the auxiliary predicates min and max; they hold for the least (resp. the greatest) element of a model. Each element of the model corresponds to precisely one state $q \in Q$, as encoded by conjunct 3. Thus the model is just a sequence of states. The first of them must be the starting state q_I and the last must be a final state $q_F \in F$, as defined by conjunct 4. Every two consecutive elements of the model form a transition. A state in which the transition is fired is marked by predicate of the form inc_c or dec_c denoting a counter to increment or decrement; this is specified by conjunct 5. Every state, with the exception of the last one, must be labelled by precisely one predicate of the form inc_c or dec_c, as expressed by conjuncts 6–8. The last element is not labelled by any of these predicates (conjunct 9), as no transition is fired there. Since values of all counters in starting and final state is 0 and no counter may fall below 0, each incrementation of a counter c must be followed by its decrementation, and conversely, each decrementation of c must be preceded by its incrementation. We use the relation s to match these increments and decrements, as stated in conjunct 10. Conjunct 11 states that decrementation of a counter indeed follows its incrementation. Since each state, except the final one, is a starting state of some transition, it either corresponds to incrementation or decrementation of some counter. Therefore it emits or accepts precisely one edge labelled s, as stated by conjunct 12 of φ_M. Formally, we have the following lemma and a corollary that results from it.

▶ **Lemma 12.** *Let $M = \langle Q, C, \delta, q_I, F\rangle$ be a reduced multicounter automaton and let φ_M be a $\mathrm{C}^2[\Sigma_u, \{<, +1, s\}, \{<, +1\}]$ formula constructed in Definition 11. Formula φ_M is finitely satisfiable if and only if M is non-empty.*

▶ **Corollary 13.** *Finite satisfiability problem for $\mathrm{C}^2[\Sigma_u, \{<, +1, s\}, \{<, +1\}]$ is at least as hard as emptiness problem for multicounter automata.*

6 Satisfiability of $\mathrm{C}^2[\Sigma_u, \Sigma_b, \{<, +1\}]$

In this section we show that the finite satisfiability problem of $\mathrm{C}^2[\Sigma_u, \Sigma_b, \{<, +1\}]$ is decidable.

Fix a finite signature Σ satisfying $\Sigma \subseteq \Sigma_u \cup \Sigma_b$. A *2-type* is a maximal consistent conjunction of atomic and negated atomic formulas over Σ involving only the variables x and y and satisfying three additional restrictions: first, it contains $\neg x = y$; second, whenever it contains $+1(x, y)$ or $+1(y, x)$, it also contains respectively $x < y$ or $y < x$; and third, it contains either $x < y$ or $y < x$, but not both. We will identify a 1-type π (a 2-type τ) with the set of positive atomic formulas occurring in π (in τ). Each 2-type $\tau(x, y)$ uniquely determines two 1-types of x and y, respectively, that we denote $\mathrm{tp}_1(\tau)$ and $\mathrm{tp}_2(\tau)$. For a 2-type τ the 2-type obtained by swapping the variables x and y is denoted τ^{-1}. Symbol $\mathcal{T}(\Sigma)$ denotes the set of of 2-types over Σ.

For a structure $\mathcal{A}$ over the signature Σ and an element $e \in \mathcal{A}$, $\mathrm{tp}^{\mathcal{A}}(e)$ denotes the unique 1-type $\pi \in \Pi(\Sigma)$ such that $\mathcal{A} \models \pi(e)$. Similarly, for $e_1, e_2 \in \mathcal{A}$, $\mathrm{tp}^{\mathcal{A}}(e_1, e_2)$ is the unique 2-type $\tau \in \mathcal{T}(\Sigma)$ such that $\mathcal{A} \models \tau(e_1, e_2)$. If $\mathcal{A} \models \tau(e_1, e_2)$, we say that e_1 *emits* the type τ and e_2 *accepts* it and that τ *originates* in e_1. A 1-type π (resp. 2-type τ) is *realised* in $\mathcal{A}$ if $\pi = \mathrm{tp}^{\mathcal{A}}(e)$ (resp. $\tau = \mathrm{tp}^{\mathcal{A}}(e_1, e_2)$) for some $e \in \mathcal{A}$ (resp. $e_1, e_2 \in \mathcal{A}$, with $e_1 \neq e_2$). Symbols $\Pi(\mathcal{A})$ and $\mathcal{T}(\mathcal{A})$ denote respectively the set of 1-types and the set of 2-types over Σ realised in $\mathcal{A}$. A 1-type $\kappa \in \Pi(\Sigma)$ that has only one realisation in a structure $\mathcal{A}$ is said to be a *king 1-type* in $\mathcal{A}$. If an element e of $\mathcal{A}$ realises a king 1-type then it is said to be a king in $\mathcal{A}$. Any structure may have multiple kings.

If Σ is a relational signature and $\overline{f} = f_1, \ldots, f_m$ is a sequence of distinct binary predicates in Σ, then the pair $\langle \Sigma, \overline{f} \rangle$ is called a *classified signature*. Let $\langle \Sigma, \bar{f} \rangle$ be a classified signature and let $\tau(x, y)$ be a 2-type over Σ. We say that τ is a *message type* over $\langle \Sigma, \bar{f} \rangle$ if $f(x, y) \in \tau(x, y)$ for some distinguished predicate f in $\bar{f}$. Given a structure $\mathcal{A}$ over a signature $\langle \Sigma, \bar{f} \rangle$ and an element $a \in \mathcal{A}$, we want to capture message types connecting a to other elements of $\mathcal{A}$ and all 2-types connecting a to kings of $\mathcal{A}$. We first define the set of all these 2-types. If K is a set of king 1-types from $\mathcal{A}$, then denote by $\tau(K, \Sigma, \bar{f})$ the set of all 2-types μ, such that μ is a message type over $\langle \Sigma, \bar{f} \rangle$ or $\mathrm{tp}_2(\mu) \in K$. A 2-type from $\tau(K, \Sigma, \bar{f})$ is called an *essential type*. If τ is an essential type (resp. a message type) such that τ^{-1} is also an essential type (resp. a message type) then we say that τ is an *invertible essential type* (resp. *invertible message type*). On the other hand, if τ is a 2-type such that neither τ nor τ^{-1} is an essential type, then we say that τ is a *silent type*.

Given a structure $\mathcal{A}$ over a classified signature $\langle \Sigma, \overline{f} \rangle$ and a message type τ, if $\mathcal{A} \models \tau(e_1, e_2)$ then e_2 is called a *witness for* e_1 in $\mathcal{A}$. It follows that if τ is an invertible message type then also e_1 is a witness for e_2. It is because $\mathcal{A} \models \tau^{-1}(e_2, e_1)$ and τ^{-1} is an (invertible) message type.

Since we consider predicates of arity at most 2, a structure $\mathcal{A}$ can be seen as a complete directed graph, where nodes are labelled by 1-types and edges are labelled by 2-types. Thus to define such a structure it is enough to define 1-types of its elements and 2-types of all pairs of elements provided that the projections of 2-types onto 1-types coincide with these 1-types and that for each pair $\langle e_1, e_2 \rangle$ of elements connected by a 2-type μ the pair $\langle e_2, e_1 \rangle$ is connected by the 2-type μ^{-1}.

Normal form of C^2 formulas

For a natural number n denote by $\underline{n}$ the set $\{1, \ldots, n\}$. We will assume that input $\mathrm{C}^2[\Sigma_u, \Sigma_b, \{<, +1\}]$ formula φ is in a normal form

$$\varphi = \quad \forall x \forall y.(\alpha(x, y) \vee x = y) \wedge \quad \bigwedge_{h \in \underline{m}} \forall x \exists^{=1} y.(f_h(x, y) \wedge x \neq y) \tag{1}$$

where α is a quantifier-free formula with unary and binary predicate symbols and $f_1, \ldots, f_m$ are distinguished binary predicates. By a routine adaptation of transformation in [9] we may convert each C^2 formula to an exponentially larger C^2 formula φ' in normal form, such that φ and φ' are equisatisfiable (on structures of cardinality > 1). From now on we also assume that the classified vocabulary of φ is $\langle \Sigma, \bar{f} \rangle$, where $\bar{f} = f_1, \ldots f_m$ and Σ is a finite subset of $\Sigma_u \cup \Sigma_b$.

▶ Remark. In Section 4 we use a notion of normal forms for C^2 formulas with only polynomial blowup. Here, for simplicity of presentation, we decided to employ the one with exponential blowup. Since there is no elementary upper bound on the complexity of the problem to which we reduce our logic, the construction in the present section would not benefit from the usage of a more succinct normal form.

Normal structures

Now, to simplify the reasoning, we restrict the class of models that we consider.

▶ **Definition 14.** A finite structure $\mathcal{A} \in \mathcal{O}(\Sigma, <, \dashv)$ over a classified signature $\langle \Sigma, \bar{f} \rangle$ is *normal* if

1. both the smallest and the largest elements w.r.t. $<^{\mathcal{A}}$ are kings in $\mathcal{A}$,
2. for every two non-king elements $e_1, e_2 \in \mathcal{A}$ satisfying $e_1 <^{\mathcal{A}} e_2$ there exist two elements $e_1', e_2' \in \mathcal{A}$ such that $e_1' <^{\mathcal{A}} e_2'$, $\mathrm{tp}^{\mathcal{A}}(e_1) = \mathrm{tp}^{\mathcal{A}}(e_1')$, $\mathrm{tp}^{\mathcal{A}}(e_2) = \mathrm{tp}^{\mathcal{A}}(e_2')$, and $\mathrm{tp}^{\mathcal{A}}(e_1', e_2')$ is a silent 2-type,
3. for every node $e \in \mathcal{A}$ and $f \in \bar{f}$ we have $|\{e' \in \mathcal{A} \mid \mathcal{A} \models f(e, e')\}| = 1$, and
4. for every $e_1, e_2 \in \mathcal{A}$ if $\dashv^{\mathcal{A}}(e_1, e_2)$ then $\mathrm{tp}^{\mathcal{A}}(e_1, e_2)$ is an invertible essential type.

The following lemma says that when dealing with models of $C^2[\Sigma_u, \Sigma_b, \{<, \dashv\}]$ formulas, we may restrict to normal structures.

▶ **Lemma 15.** *Let φ be a $C^2[\Sigma_u, \Sigma_b, \{<, \dashv\}]$ formula in normal form over a vocabulary $\langle \Sigma, \bar{f} \rangle$. If φ is finitely satisfiable then there exists a vocabulary $\langle \Sigma', \bar{f}' \rangle$ such that $\Sigma \subseteq \Sigma'$ and $\bar{f} \subseteq \bar{f}'$ and a finite normal $\langle \Sigma', \bar{f}' \rangle$-structure $\mathcal{B}$ such that $\mathcal{B} \models \varphi$. Moreover, $|\Sigma'|$ is polynomial in $|\Sigma|$ and $|\bar{f}'| = |\bar{f}| + 2$.*

Star types

Given a structure $\mathcal{A}$ over a signature $\langle \Sigma, \bar{f} \rangle$ and an element $a \in \mathcal{A}$, we want to capture essential 2-types emitted from a to other elements of $\mathcal{A}$. For this reason we introduce star types.

▶ **Definition 16** (Star type in $\mathcal{A}$)**.** Let $\mathcal{A}$ be a normal structure over $\langle \Sigma, \bar{f} \rangle$, and let a be an element of $\mathcal{A}$. Let $K = \{\kappa \mid \kappa \text{ is a king type in } \mathcal{A}\}$. A *star type* of a in $\mathcal{A}$, denoted $\mathrm{st}^{\mathcal{A}}(a)$ is a pair $\sigma = \langle \pi, \mathcal{T} \rangle$ where $\pi = \mathrm{tp}^{\mathcal{A}}(a)$ and $\mathcal{T}$ is the set of essential types originating in a:

$$\mathcal{T} = \{\mu \in \tau(K, \Sigma, \bar{f}) \mid \mathrm{tp}^{\mathcal{A}}(a, b) = \mu \text{ for some } b \in \mathcal{A}\}.$$

We denote the type π by $\pi(\sigma)$. We say that a 2-type μ *occurs* in σ, written $\mu \in \sigma$, if $\mu \in \mathcal{T}$. We write $\sigma - \mu$ for the star-type $\sigma' = \langle \pi, \mathcal{T} \setminus \{\mu\} \rangle$. When S is a set of 2-types we write $\sigma \setminus S$ to denote the star type $\langle \pi, \mathcal{T} \setminus S \rangle$.

Observe that in the definition above σ satisfies the conditions

1. for all $\mu_1, \mu_2 \in \sigma$ if $f(x, y) \in \mu_1$ and $f(x, y) \in \mu_2$ for some $f \in \bar{f}$ then $\mu_1 = \mu_2$,
2. $\mu \in \sigma$ implies $\mathrm{tp}_1(\mu) = \pi$ for all $\mu \in \tau(K, \Sigma, \bar{f})$,

3. $|\{\mu \in \sigma \mid \mathrm{tp}_2(\mu) = \kappa\}| = 1$ for all $\kappa \in K$ such that $\kappa \neq \pi$,
4. $|\{\mu \in \sigma \mid \mathrm{tp}_2(\mu) = \pi\}| = 0$ if $\pi \in K$.

The first of these conditions is obvious, as normal structures emit precisely one edge τ with $f(x,y) \in \tau$ for $f \in \bar{f}$. The second one says that all 2-types originating in a have the same 1-type of the origin, namely the 1-type of a. The third one says that for all kings k (in $\mathcal{A}$) the element a is connected with k by exactly one 2-type (provided that $k \neq a$). The last condition says that if a is a king then it is not connected with itself by any 2-type (recall that 2-types connect different elements).

▶ **Definition 17.** A *star type* over the set of 2-types $\tau(K, \Sigma, \bar{f})$ is any pair of the form $\langle \pi, \mathcal{T} \rangle$ satisfying conditions 1–4 above. A structure $\mathcal{A}$ is said to realise a star type σ if $\mathrm{st}^{\mathcal{A}}(a) = \sigma$ for some $a \in \mathcal{A}$.

For a given set of star types ST, by $\pi(\mathrm{ST})$ we denote the set of 1-types $\{\pi(\sigma) \mid \sigma \in \mathrm{ST}\}$, by $\tau(\mathrm{ST})$ — the set of 2-types occurring in star types from ST, that is the set $\{\mu \mid \exists \sigma \in \mathrm{ST}. \mu \in \sigma\}$, and by *partial*(ST) the set $\{\langle \pi, \mathcal{T}' \rangle \mid \langle \pi, \mathcal{T} \rangle \in \mathrm{ST}$ for some $\mathcal{T}$ satisfying $\mathcal{T}' \subseteq \mathcal{T}\}$. Elements of *partial*(ST) are called *partial star types*. A partial star type is said to be *empty* if it is of the form $\langle \pi, \emptyset \rangle$.

Frames

We now introduce finite and small structures called frames. Frames will be used in deciding the finite satisfiability problems for $\mathrm{C}^2[\Sigma_u, \Sigma_b, \{<, +1\}]$: together with multicounter automata they provide a description of finite models of a given formula.

▶ **Definition 18** (Frame)**.** Let $\langle \Sigma, \bar{f} \rangle$ be a classified signature, K be a set of 1-types over Σ, let ST be a set of star types over $\tau(K, \Sigma, \bar{f})$ and let Ξ be a set of silent 2-types over $\langle \Sigma, \bar{f} \rangle$. A tuple $\langle K, \mathrm{ST}, \Xi, \Sigma, \bar{f} \rangle$ is called *a frame* if the following conditions are satisfied

1. for each 2-type $\tau \in \tau(\mathrm{ST}) \cup \Xi$ if $+1(x,y) \in \tau$ or $+1(y,x) \in \tau$ then τ is an invertible essential type,
2. there exists exactly one star type $\sigma_{first} \in \mathrm{ST}$ such that for every $\tau \in \sigma_{first}$ we have $+1(y,x) \notin \tau$,
3. there exists exactly one star type $\sigma_{last} \in \mathrm{ST}$ such that for every $\tau \in \sigma_{last}$ we have $+1(x,y) \notin \tau$,
4. for each $\kappa \in K$ there exists exactly one $\sigma \in \mathrm{ST}$ such that $\pi(\sigma) = \kappa$, and
5. for each star type $\sigma \in \mathrm{ST}$ and each 2-type μ, if $\mu \in \sigma$ then $\mathrm{tp}_2(\mu) \in \pi(\mathrm{ST})$.

Frames are intended to describe local configurations in normal structures $\mathcal{A}$. The set K contains all king 1-types of $\mathcal{A}$, ST — all star types of $\mathcal{A}$ and the set Ξ — all silent 2-types realised in $\mathcal{A}$. Condition 1 says that every node in $\mathcal{A}$ is connected to its successor and predecessor by invertible essential types. Conditions 2 and 3 say that there are unique star types for the first and the last node in $\mathcal{A}$. Conditions 1–3 follow from the assumption that $\mathcal{A}$ is normal. Condition 4 says that each king has exactly one star type. Condition 5 ensures that if a neighbour of a node in a structure has some 1-type π, then there exists a star type $\sigma \in \mathrm{ST}$ such that $\pi \in \pi(\mathrm{ST})$. The above two conditions hold in every relational structure.

Intuitively, we want to check finite satisfiability of a C^2 formula φ by guessing a right frame. "Right" means here that two conditions must be satisfied. First, the frame should be locally consistent with φ. This means that every 2-type occurring in the frame entails the subformulas of φ of the form $\forall x \forall y \ldots$, and that the number of witnesses in every star type is correct. This is formalised in the following definition. Second, the frame should be

globally consistent in the sense that there exists a structure that conforms to this frame — this is formalised in Definition 20.

▶ **Definition 19** ($\mathcal{F} \models \varphi$)**.** Consider a frame $\mathcal{F} = \langle K, \mathrm{ST}, \Xi, \Sigma, \bar{f}\rangle$ and a $\mathrm{C}^2[\Sigma_u, \Sigma_b, \{<, +1\}]$ formula φ in normal form (1) over $\langle \Sigma, \bar{f}\rangle$. We say that $\mathcal{F}$ *satisfies* φ, in symbols $\mathcal{F} \models \varphi$, if

- for each 2-type $\mu \in \Xi \cup \tau(\mathrm{ST})$, the formula α is a consequence of μ and of μ^{-1}, that is $\models \mu \to \alpha$ and $\models \mu^{-1} \to \alpha$, where μ is seen as conjunction of literals, and
- for each $\sigma \in \mathrm{ST}$ and $h \in \underline{m}$ we have $|\{\mu \in \sigma \mid f_h(x, y) \in \mu\}| = 1$.

▶ **Definition 20.** Let $\langle K, \mathrm{ST}, \Xi, \Sigma, \bar{f}, c\rangle$ be a frame, and $\mathcal{A}$ structure over $\langle \Sigma, \bar{f}\rangle$. We say that $\mathcal{A}$ *fits to the frame* $\mathcal{F}$ if

- the set of king 1-types realised in structure $\mathcal{A}$ is K, and
- the set of all silent types realised in $\mathcal{A}$ is a subset of Ξ, and
- the set of star types of $\mathcal{A}$ is a subset of ST, in symbols $\mathrm{st}^{\mathcal{A}}(\mathcal{A}) \subseteq \mathrm{ST}$.

The following proposition reduces the finite satisfiability problem of $\mathrm{C}^2[\Sigma_u, \Sigma_b, \{<, +1\}]$ to the problem of existence of a structure in $\mathcal{O}(\Sigma, <, +1)$ that fits to a given frame.

▶ **Proposition 21.** *Let φ be a $\mathrm{C}^2[\Sigma_u, \Sigma_b, \{<, +1\}]$ formula in normal form over a vocabulary $\langle \Sigma, \bar{f}\rangle$, where $\{<, +1\} \subseteq \Sigma$. Let $\mathcal{A}$ be a structure in $\mathcal{O}(\Sigma, <, +1)$.*

1. *If $\mathcal{A}$ is normal and $\mathcal{A} \models \varphi$ then there exists a frame $\mathcal{F}$, such that $\mathcal{A}$ fits to $\mathcal{F}$ and $\mathcal{F} \models \varphi$.*
2. *If there exists a frame $\mathcal{F}$ such that $\mathcal{A}$ fits to $\mathcal{F}$ and $\mathcal{F} \models \varphi$ then $\mathcal{A} \models \varphi$.*

Proof. For the proof of the first statement, assume that $\mathcal{A}$ is normal and $\mathcal{A} \models \varphi$. Let K be the set of king 1-types realised in $\mathcal{A}$, let ST be the set of star types of $\mathcal{A}$ and let Ξ be the set of all silent types realised in $\mathcal{A}$. The facts that tuple $\mathcal{F} = \langle K, \mathrm{ST}, \Xi, \Sigma, \bar{f}, c\rangle$ forms a frame, $\mathcal{F} \models \varphi$ and $\mathcal{A}$ fits to $\mathcal{F}$ are immediate, once Definitions 18, 19 and 20 are spelled.

For the proof of the second statement, let $\mathcal{F}$ be a frame such that $\mathcal{F} \models \varphi$ and let $\mathcal{A}$ be a structure such that $\mathcal{A}$ fits to $\mathcal{F}$. Since φ is in normal form, it is of the form (1). Let μ be any 2-type realised in $\mathcal{A}$. Then either μ is a silent type or it occurs in some star type realised in $\mathcal{A}$. In any case, by Definition 20 we have that $\mu \in \Xi \cup \tau(\mathrm{ST})$. By Definition 19 it follows that $\models \mu \to \alpha$. So $\mathcal{A} \models \forall x \forall y.(\alpha \vee x = y)$. Since $\mathcal{A}$ fits to $\mathcal{F}$ and $\mathcal{F} \models \varphi$, it also follows that for each star type σ realised in $\mathcal{A}$ and each h such that $1 \leq h \leq m$ we have $|\{\mu \in \sigma \mid f_h(x, y) \in \mu\}| = 1$, and thus $\mathcal{A} \models \bigwedge_{h \in \underline{m}} \forall x \exists^{=1} y.(f_h(x, y) \wedge x \neq y)$. Hence $\mathcal{A} \models \varphi$ as required. ◀

High-level multicounter automata

In the rest of this section we will want to decide for a given frame $\mathcal{F}$ if there exists a structure in $\mathcal{O}(\Sigma, <, +1)$ that fits to this frame. This will be done by a reduction to emptiness problem for Multicounter Automata.

We now introduce a syntactic extension to multicounter automata that we call *High-level MultiCounter Automata* (**HMCA**). The idea is to specify transitions of an automaton as programs in a higher-level imperative language with conditionals, loops and arrays, which leads to clearer exposition of reachability problems. A transition in a High-Level MCA is a sequence Δ of actions; each action in turn may update and test finite-domain variables, and conduct conditional or loop instructions depending on results of these tests. A transition may also increment or decrement, but not test the value of, counters, which are the only infinite-domain variables of the automaton.

Formally, an HMCA is a tuple $H = \langle \mathrm{V}_{fin}, \mathrm{Vec}_{\mathbb{N}}, \mathrm{Type}, \Delta, \rho_I, P_F, E\rangle$. Set V_{fin} consists of variables v to be interpreted in the finite domain $\mathrm{Type}(v)$. We may think of V_{fin} as a declaration of finite-domain variables of the program. Set $\mathrm{Vec}_{\mathbb{N}}$ corresponds to a declaration

of arrays, it consists of variables $\overrightarrow{vec}$ to be interpreted as vectors of natural numbers indexed by elements of some finite set $\mathbb{A}$, where $\mathrm{Type}(\overrightarrow{vec}) = \mathbb{A} \to \mathbb{N}$. We will refer to the index set $\mathbb{A}$ as the domain $\mathrm{Dom}(\overrightarrow{vec})$. The Type function assigns to every variable in $\mathrm{V}_{fin} \cup \mathrm{Vec}_{\mathbb{N}}$ its type. The sequence of actions Δ is the actual program built from actions defined below. The starting state of H is ρ_I, the set of accepting states is P_F. Set E is a subset of $\{\langle \vec{v}, a\rangle \mid \vec{v} \in \mathrm{Vec}_{\mathbb{N}},\ a \in \mathrm{Dom}(\vec{v})\}$, and is used in the acceptance condition explained later.

We now define a set of actions α that constitute transitions in **HMCA**. The simplest action is an *assignment* of the form $v := \mathit{Expr}$, where v is a variable of some domain $\mathbb{A} = \mathrm{Type}(v)$, and *Expr* is an expression built from variables from V_{fin}, constants from appropriate domains and operators. An *operator* is any effectively computable function, e. g., $\cup, \cap, \setminus$ are operators of domain $(2^{\mathbb{B}})^2 \to 2^{\mathbb{B}}$ for any domain $\mathbb{B}$; function $\pi : \mathrm{ST}(\mathcal{K}, \Sigma, \bar{f}) \to \Pi(\Sigma)$ from Definition 16 is also an operator, provided that our finite domain contains $\mathrm{ST}(\mathcal{K}, \Sigma, \bar{f})$ and $\Pi(\Sigma)$. We silently extend Type function to constants by letting $\mathrm{Type}(a) = \mathbb{A}$ if $a \in \mathbb{A}$, and to expressions, e. g., $\mathrm{Type}(s_1 \cup s_2) = 2^{\mathbb{B}}$ if $\mathrm{Type}(s_1) = 2^{\mathbb{B}}$ and $\mathrm{Type}(s_2) = 2^{\mathbb{B}}$. We require that assignments $v := \mathit{Expr}$ are well typed, i. e., that $\mathrm{Type}(v) = \mathrm{Type}(\mathit{Expr})$. An *atomic test* is of the form $\mathit{Expr}_1 = \mathit{Expr}_2$ or $\mathit{Expr}_3 \in \mathit{Expr}_4$, where Expr_1, Expr_2, Expr_3, Expr_4 are expressions such that $\mathrm{Type}(\mathit{Expr}_1) = \mathrm{Type}(\mathit{Expr}_2)$ and $\mathrm{Type}(\mathit{Expr}_4) = 2^{\mathrm{Type}(\mathit{Expr}_3)}$. A *test* is an arbitrary Boolean combination of *atomic tests*. Notice that tests do not use counters. A *non-deterministic assignment* action is of the form **guess** $v \in \mathit{Expr}$ **with** *Test*, where $\mathrm{Type}(\mathit{Expr}) = 2^{\mathrm{Type}(v)}$ and the variable v may occur in *Test*. A conditional action is of the form **if** *Test* **then** α^* **else** α'^* **endif** or **if** *Test* **then** α^* **endif**. A loop action is of the form **while** *Test* **do** α^* **endwhile**. An *incrementing action* (resp. *decrementing action*) is of the form $\mathbf{inc}(\vec{f}[\mathit{Expr}])$ (resp. $\mathbf{dec}(\vec{f}[\mathit{Expr}])$), where *Expr* evaluates to an index of the array $\vec{f}$, that is, $\vec{f} \in \mathrm{Vec}_{\mathbb{N}}$, $\mathrm{Type}(\vec{f}) = \mathbb{A} \to \mathbb{N}$ and $\mathrm{Type}(\mathit{Expr}) = \mathbb{A}$. The remaining action, **Reject**, simply rejects current computation.

Expressions and tests are evaluated in context of variable valuations. A *variable valuation* (also called a *state*) is any function ρ that assigns to every finite-domain variable v a value $[\![v]\!]_\rho \in \mathrm{Type}(v)$.We write $[\![v]\!]_\rho = \rho(v)$ for $v \in \mathbb{V}_{fin}$, $[\![\mathit{Expr}_1 \bowtie \mathit{Expr}_2]\!]_\rho = [\![\mathit{Expr}_1]\!]_\rho \bowtie [\![\mathit{Expr}_2]\!]_\rho$, where $\bowtie \in \{\cup, \cap, \setminus\}$ and $[\![f(\mathit{Expr}_1, \ldots, \mathit{Expr}_k)]\!]_\rho = f([\![\mathit{Expr}_1]\!]_\rho, \ldots, [\![\mathit{Expr}_k]\!]_\rho)$, where f is an operator. In a similar way we define semantics of tests.

A *counter valuation* is any function ϑ that assigns (a sequence of) natural numbers to (arrays of) counters. A *configuration* of **HMCA** H is a pair $\langle \rho, \vartheta\rangle$ where ρ is an variable valuation (i. e., a state) and ϑ is a counter valuation. Actions transform configurations. Most actions work only on variable valuations; the exceptions are incrementing and decrementing of counters. With the exception of the decrementing action, the semantics of actions is self-explanatory; $\mathit{dec}(c)$ decrements the counter c if it is strictly positive and otherwise (if it is 0) it rejects the current computation.

A *run* of an HMCA H is a sequence of configurations $\langle \rho_1, \vartheta_1\rangle, \ldots \langle \rho_k, \vartheta_k\rangle$ such that $\langle \rho_{i+1}, \vartheta_{i+1}\rangle$ is obtained after executing transition Δ in configuration $\langle \rho_i, \vartheta_i\rangle$, for $i \in \{1, \ldots, k-1\}$. A run is *accepting*, if it starts in an initial configuration $\langle \rho_I, \vartheta_0\rangle$ with ρ_I being initial state and ϑ_0 assigning 0 to all counters, and it ends in some configuration $\langle \rho_F, \vartheta_F\rangle$ with ρ_F being a final state and ϑ_F assigning 0 to all counters specified in the set E of final counters: $\vartheta_F(\vec{f})(a) = 0$ for every $\langle f, a\rangle \in E$. The emptiness problem for high-level multicounter automata is the question whether a given automaton H has an accepting run.

In the full version of the paper we give formal syntax and semantics to high-level multicounter automata and we prove that HMCA can be compiled to multicounter automata. Intuitively, the control structures and finite-domain variables (including tests for zero on finite-domain variables) can be hidden in states of the constructed MCA. Formally, we have the following proposition.

▶ **Proposition 22.** *Emptiness problem for* ***HMCA*** *is reducible to emptiness problem of multicounter automata, and is therefore decidable.*

Figure 1 shows a high-level multicounter automaton $H_{\mathcal{F}}$ that for a given frame $\mathcal{F}$ checks whether there exists a normal structure that fits to $\mathcal{F}$. The automaton guesses one by one the sequence of elements of the structure as they appear in the order $<$. The constructed HMCA keeps track of the set of nodes visited (i. e., guessed) so far; their 1-types are stored in variable Visited; variable Required stores the set of king types that still must be constructed. These two variables are updates in lines 4–7.

A crucial notion in the construction of the automaton is the difference type of a node w.r.t. to another node in a structure. Fig 2 shows an example of a difference type.

▶ **Definition 23** (Difference type of e_1 w.r.t. e in a structure $\mathcal{A}$)**.** Let $\mathcal{A} \in \mathcal{O}(\Sigma, <, \dashv\!\!\dashv)$ and $e_1, e \in \mathcal{A}$ be elements satisfying $e_1 <^{\mathcal{A}} e$. Let σ be the star type of e_1 and let $\{\tau_i\}_{i=1}^{k}$ be essential types emitted from e_1 and accepted by nodes $<^{\mathcal{A}}$ than e. A partial star type $\sigma \setminus \{\tau_i\}_{i=1}^{k}$ is called a *difference type of* e_1 *w.r.t.* e *in* $\mathcal{A}$.

▶ **Definition 24.** Let $A \in \mathcal{O}(\Sigma, <, \dashv\!\!\dashv)$ and $e \in \mathcal{A}$. The *cut* at point e in $\mathcal{A}$ is a vector $\overrightarrow{\mathsf{Cut}}_e$ of natural numbers indexed by star types on $\tau(K, \Sigma, \bar{f})$ such that

$$\overrightarrow{\mathsf{Cut}}_e[\sigma] = |\{e_1 \in \mathcal{A} \mid \text{difference type of } e_1 \text{ w.r.t. } e \text{ in } \mathcal{A} \text{ is } \sigma\}|.$$

Intuitively, the cut vector $\overrightarrow{\mathsf{Cut}}_e$ informs us, for each 2-type τ, how many edges of type τ emitted by nodes smaller than e must be accepted by nodes greater or equal to e. Additionally, it informs which of these edges have common origin, i. e., they belong to a star type of the same node. This information is used when we define 2-types connecting e with smaller nodes. When we define a 2-type connecting e with a node e' smaller than e, we have to subtract this 2-type from the current cut. At the same time we have to remember that for each pair of nodes there is only one 2-type connecting them, so when we subtract two 2-types from a cut, we have to be sure that their origins are different. This is why the cut vector is indexed by difference types and not by 2-types.

For a star type σ define $\sigma^{<} = \{\tau \in \sigma \mid (y < x) \in \tau\}$ and $\sigma^{>} = \{\tau \in \sigma \mid (x < y) \in \tau\}$. Intuitively $\sigma^{<}$ (resp. $\sigma^{>}$) denotes the subset of σ containing essential types emitted to smaller (resp. larger) nodes. For a partial star type σ from ST define $\tau_{\dashv\!\!\dashv}(\sigma)$ as the only 2-type τ such that $\dashv\!\!\dashv(x, y) \in \tau$, or the special value $\perp$ if σ is the star type of last node of a structure. Similarly, define $\mathit{first}(\sigma)$ to be an arbitrary 2-type τ such that $\tau \in \sigma$. The value of $\overrightarrow{\mathsf{Cut}}$ is updated in two loops in lines 9–27. During the computation some counters from $\overrightarrow{\mathsf{Cut}}$ are decremented, and some counters from $\overrightarrow{\mathsf{Processed}}$ — incremented. Decrementation of a counter corresponds to establishing a 2-type between e and some e' smaller than e (this is done in the loop in lines 9–19), or between e' and e (loop in lines 21–27). In order not to establish multiple 2-types between the same pair of nodes, we remove the difference type of e' w.r.t. e from $\overrightarrow{\mathsf{Cut}}$ and store it in $\overrightarrow{\mathsf{Processed}}$. When the second loop (lines 21–27) finishes its execution the initial value of $\overrightarrow{\mathsf{Cut}}$ vector for next node is the sum of the values of updated vectors $\overrightarrow{\mathsf{Cut}}$ and $\overrightarrow{\mathsf{Processed}}$ in line 27.

In lines 29–34 we guess the star type of the next node, or the special value $\perp$ in case the maximal element of the structure is already guessed. To keep the constructed structure normal (and to be able to define silent 2-types between non-king nodes) we have to satisfy condition 2 in Definition 14. Therefore, while guessing a consecutive node, we must check that it does not violate this condition. Therefore we guess (the star type of) the consecutive

Type $\mathrm{Type}(\sigma_c) = \mathrm{ST} \cup \{\bot\}$, $\mathrm{Type}(\sigma)$, $\mathrm{Type}(\sigma_u)$ and $\mathrm{Type}(\sigma_g)$ is *partial*(ST), $\mathrm{Type}(\tau) = \tau(\mathrm{ST})$ and $\mathrm{Type}(\tau_{\text{Ш}}) = \tau(\mathrm{ST}) \cup \{\bot\}$. $\mathrm{Type}(\mathrm{Required}) = 2^K$ and $\mathrm{Type}(\mathrm{Visited}) = 2^{\pi(ST)}$. $\mathrm{Type}(\overrightarrow{\mathsf{Cut}})$ and $\mathrm{Type}(\overrightarrow{\mathsf{Processed}})$ is *partial*(ST) $\to \mathbb{N}$.

Initial Configuration Initial state is ρ_I such that $\rho_I(\sigma_c) = \sigma_{first}$, $\rho_I(\mathrm{Required}) = K$, $\rho_I(\mathrm{Visited}) = \emptyset$ and the value of ρ_I on remaining variables is arbitrary (but fixed). Initial counter valuation assigns 0 to all counters.

Accepting Configurations The set of accepting states P_F consists of all states ρ_F satisfying $K \subseteq \rho_F(\mathrm{Visited})$. Accepting counter valuations are defined by the set $E = \{\langle \overrightarrow{\mathsf{Cut}}, \sigma \rangle \mid \sigma \in \mathit{partial}(\mathrm{ST}) \text{ is non-empty}\}$.

Transition Δ

if $\sigma_c = \bot$ **then**
  Reject
endif
if $\pi(\sigma_c) \in \mathrm{Required}$ **then**
  $\mathrm{Required} := \mathrm{Required} \setminus \{\pi(\sigma_c)\}$
endif
$\mathrm{Visited} := \mathrm{Visited} \cup \{\pi(\sigma_c)\}$
$\sigma := {\sigma_c}^{<}$
while $\sigma \neq \emptyset$ **do**
  $\tau := \mathit{first}(\sigma)$
  $\sigma := \sigma - \tau$
  if τ is an invertible essential type **then**
    guess $\sigma_u \in \mathit{partial}(\mathrm{ST})$ **with** $\tau^{-1} \in {\sigma_u}^{>}$
  else
    guess $\sigma_u \in \mathit{partial}(\mathrm{ST})$ **with** $\pi(\sigma_u) = \mathrm{tp}_2(\tau)$
  endif
  dec$(\overrightarrow{\mathsf{Cut}}[{\sigma_u}^{>}])$
  inc$(\overrightarrow{\mathsf{Processed}}[{\sigma_u}^{>} - \tau^{-1}])$
endwhile
guess *anotherIteration* $\in \{\mathbf{true}, \mathbf{false}\}$
while *anotherIteration* **do**
  guess $\sigma_g \in \mathit{partial}(\mathrm{ST})$
  guess $\tau \in {\sigma_g}^{>}$ **with** $\mathrm{tp}_2(\tau) = \pi(\sigma_c)$ **and** (τ is non-invertible essential type)
  dec$(\overrightarrow{\mathsf{Cut}}[{\sigma_g}^{>}])$
  inc$(\overrightarrow{\mathsf{Processed}}[{\sigma_g}^{>} - \tau])$
  guess *anotherIteration* $\in \{\mathbf{true}, \mathbf{false}\}$
endwhile
inc$(\overrightarrow{\mathsf{Cut}}[{\sigma_c}^{>}])$
$\tau_{\text{Ш}} := \tau_{\text{Ш}}({\sigma_c}^{>})$
if $\tau_{\text{Ш}} = \bot$ **then**
  $\sigma_c := \bot$
else
  guess $\sigma_c \in \mathrm{Allowed}(\mathrm{Visited})$ **with** $(\tau_{\text{Ш}})^{-1} \in \sigma_c$
endif
guess *anotherIteration* $\in \{\mathbf{true}, \mathbf{false}\}$
while *anotherIteration* **do**
  guess $\sigma_g \in \mathit{partial}(\mathrm{ST})$
  dec$(\overrightarrow{\mathsf{Processed}}[{\sigma_g}^{>}])$
  inc$(\overrightarrow{\mathsf{Cut}}[{\sigma_g}^{>}])$
  guess *anotherIteration* $\in \{\mathbf{true}, \mathbf{false}\}$
endwhile

Figure 1 A high-level multicounter automaton $H_{\mathcal{F}}$ corresponding to a frame $\mathcal{F}$. Here the set of finite-domain variables is $\{\sigma_c, \mathrm{Visited}, \mathrm{Required}, \sigma, \tau, \sigma_u, \sigma_g, \tau_{\text{Ш}}, \mathit{anotherIteration}\}$, and there are two arrays of counters $\overrightarrow{\mathsf{Cut}}$ and $\overrightarrow{\mathsf{Processed}}$.

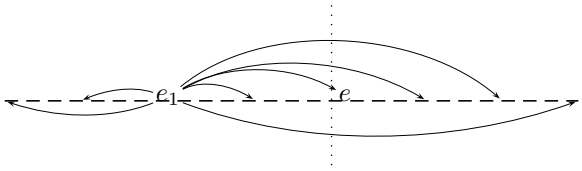

Figure 2 Difference type of e_1 w.r.t. e contains the arrows that cross the dotted line.

node from the set

$$\begin{aligned}&\text{Allowed}(\text{Visited}) = \{\sigma \in \text{ST} \mid \pi(\sigma) \in K \setminus \text{Visited}\} \cup \\ &\quad \{\sigma \in \text{ST} \mid \forall \pi \in \text{Visited}\,(\pi \notin K \Rightarrow \exists \tau \in \Xi.\text{tp}_1(\tau) = \pi \wedge (x < y) \in \tau \wedge \text{tp}_2(\tau) = \pi(\sigma))\}.\end{aligned}$$

Finally, loop in lines 35–41 may move the content of vector $\overrightarrow{\mathsf{Processed}}$ to $\overrightarrow{\mathsf{Cut}}$. We may assume that the entire content is actually moved. Formally, the correspondence between a frame $\mathcal{F}$ and HMCA $H_{\mathcal{F}}$ is captured by the following proposition, which directly leads to the main theorem of this section.

▶ **Proposition 25.** *Let $\mathcal{F} = \langle K, \text{ST}, \Xi, \Sigma, \bar{f}\rangle$ be a frame. The automaton $H_{\mathcal{F}}$ is non-empty if and only if there exists a structure $\mathcal{M} \in \mathcal{O}(\Sigma, <, +1)$ that fits to $\mathcal{F}$.*

▶ **Theorem 26.** *The finite satisfiability problem for $\text{C}^2[\Sigma_u, \Sigma_b, \{<, +1\}]$ is decidable.*

Proof. We may assume that the input $\text{C}^2[\Sigma_u, \Sigma_b, \{<, +1\}]$ formula φ is in normal form (otherwise it can be brought to the normal form). A non-deterministic decision procedure for the finite satisfiability problem guesses a frame $\mathcal{F}$ such that $\mathcal{F} \models \varphi$ and checks if HMCA $H_{\mathcal{F}}$ is non-empty. If so, then by Proposition 25 we obtain a structure $\mathcal{M} \in \mathcal{O}(\Sigma, <, +1)$ that fits to $\mathcal{F}$. Because $\mathcal{M}$ fits to $\mathcal{F}$ and $\mathcal{F} \models \varphi$, by Proposition 21 we conclude that $\mathcal{M} \models \varphi$. This shows that φ is finitely satisfiable, so our procedure is sound. On the other hand, if φ is finitely satisfiable then, by Lemma 15, it has a model $\mathcal{M}$ which is a normal structure. Again, by Proposition 21 there exists a frame $\mathcal{F}$ such that $\mathcal{M}$ fits to $\mathcal{F}$ and $\mathcal{F} \models \varphi$. By Proposition 25 we conclude that $H_{\mathcal{F}}$ is non-empty, so the procedure is complete.

Note that the size of $\mathcal{F}$ is at most doubly exponential in the size of formula's vocabulary $\langle \Sigma, \bar{f}\rangle$, so there are finitely many frames that can be guessed, and that the emptiness problem of HMCA $H_{\mathcal{F}}$ is decidable, as stated in Proposition 22. ◀

7 Conclusion

We have shown several complexity results for finite satisfiability of two-variable logics with counting quantifiers and linear orders. In particular we proved NExpTime-completeness of the problem for $\text{C}^2[\Sigma_u, \{<, +1\}, \{<, +1\}]$, VAS-completeness for $\text{C}^2[\Sigma_u, \Sigma_b, \{<\}]$ and undecidability for $\text{C}^2[\Sigma_u, \{<_1, s_1, <_2, s_2\}, \{<_1, <_2\}]$. There are still some unsolved cases, including $\text{C}^2[\Sigma_u, \{<_1, <_2\}, \{<_1, <_2\}]$ and $\text{C}^2[\Sigma_u, \{<_1, s_1, <_2\}, \{<_1, <_2\}]$.

There are lots of open problems in the area. One of them is general satisfiability. None of the logics considered here has finite model property. Our techniques rely on finiteness of the underlying structure, so they cannot be directly applied to general satisfiability on possibly infinite structures. Among possible directions for future work one can choose combination of C^2 with other interpreted binary relations like preorders [21] or transitive relations [33]. Another possibility is to consider C^2 with closure operations on some relations, like equivalence closure [14] or deterministic transitive closure [3].

References

1 Mikolaj Bojanczyk, Anca Muscholl, Thomas Schwentick, Luc Segoufin, and Claire David. Two-variable logic on words with data. In *LICS*, pages 7–16. IEEE Computer Society, 2006.

2 Witold Charatonik, Emanuel Kieronski, and Filip Mazowiecki. Satisfiability of the two-variable fragment of first-order logic over trees. *CoRR*, abs/1304.7204, 2013.

3 Witold Charatonik, Emanuel Kieroński, and Filip Mazowiecki. Decidability of weak logics with deterministic transitive closure. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS'14, Vienna, Austria, July 14–18, 2014*, page 29, 2014.

4 Witold Charatonik and Piotr Witkowski. Two-variable logic with counting and trees. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 73–82, 2013.

5 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.

6 Diego Figueira. Satisfiability for two-variable logic with two successor relations on finite linear orders. *Computing Research Repository*, abs/1204.2495, 2012.

7 E. Grädel, M. Otto, and E. Rosen. Undecidability results on two-variable logics. *Archive for Mathematical Logic*, 38:213–354, 1999.

8 Erich Grädel, Phokion Kolaitis, and Moshe 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, Martin Otto, and Eric Rosen. Two-variable logic with counting is decidable. In *LICS*, pages 306–317. IEEE Computer Society, 1997.

11 N. Immerman, A. Rabinovich, T. Reps, M.Sagiv, and G. Yorsh. The boundary between decidability and undecidability for transitive-closure logics. In *Proceedings of the 18th Annual Conference of the European Association for Computer Science Logic (CSL'04)*, pages 160–174, 2004.

12 Emanuel Kieroński. Decidability issues for two-variable logics with several linear orders. In Marc Bezem, editor, *CSL*, volume 12 of *LIPIcs*, pages 337–351. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2011.

13 Emanuel Kieroński and Jakub Michaliszyn. Two-variable universal logic with transitive closure. In Patrick Cégielski and Arnaud Durand, editors, *CSL*, volume 16 of *LIPIcs*, pages 396–410. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012.

14 Emanuel Kieroński, Jakub Michaliszyn, Ian Pratt-Hartmann, and Lidia Tendera. Two-variable first-order logic with equivalence closure. In *LICS*, pages 431–440. IEEE, 2012.

15 Emanuel Kieroński and Martin Otto. Small substructures and decidability issues for first-order logic with two variables. In *LICS*, pages 448–457. IEEE Computer Society, 2005.

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

17 S. Rao Kosaraju. Decidability of reachability in vector addition systems (preliminary version). In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 267–281, 1982.

18 Jérôme Leroux. The general vector addition system reachability problem by presburger inductive invariants. In *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science*, pages 4–13, 2009.

19 R. J. Lipton. The reachability problem requires exponential space. 62, New Haven, Connecticut: Yale University, Department of Computer Science, Research, Jan, 1976.

20 Amaldev Manuel. Two variables and two successors. In Petr Hlinený and Antonín Kucera, editors, *MFCS*, volume 6281 of *Lecture Notes in Computer Science*, pages 513–524. Springer, 2010.

21 Amaldev Manuel and Thomas Zeume. Two-variable logic on 2-dimensional structures. In Simona Ronchi Della Rocca, editor, *Computer Science Logic 2013 (CSL 2013), CSL 2013, September 2-5, 2013, Torino, Italy*, volume 23 of *LIPIcs*, pages 484–499. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2013.

22 Ernst W. Mayr. An algorithm for the general Petri net reachability problem. *SIAM J. Comput.*, 13(3):441–460, 1984.

23 Michael Mortimer. On languages with two variables. *Mathematical Logic Quarterly*, 21(1):135–140, 1975.

24 B. O. Nash. Reachability problems in vector addition systems. *The American Mathematical Monthly*, 80(3):pp. 292–295, 1973.

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

26 Leszek Pacholski, Wieslaw Szwast, and Lidia Tendera. Complexity of two-variable logic with counting. In *LICS*, pages 318–327. IEEE, 1997.

27 Ian Pratt-Hartmann. Complexity of the two-variable fragment with counting quantifiers. *Journal of Logic, Language and Information*, 14(3):369–395, 2005.

28 Ian Pratt-Hartmann. The two-variable fragment with counting revisited. In Anuj Dawar and Ruy J. G. B. de Queiroz, editors, *WoLLIC*, volume 6188 of *Lecture Notes in Computer Science*, pages 42–54. Springer, 2010.

29 Ian Pratt-Hartmann. Logics with counting and equivalence. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS'14, Vienna, Austria, July 14–18, 2014*, page 76, 2014.

30 Thomas Schwentick and Thomas Zeume. Two-variable logic with two order relations – (extended abstract). In Anuj Dawar and Helmut Veith, editors, *CSL*, volume 6247 of *Lecture Notes in Computer Science*, pages 499–513. Springer, 2010.

31 Dana Scott. A decision method for validity of sentences in two variables. *Journal of Symbolic Logic*, 27:477, 1962.

32 Luc Segoufin. Automata and logics for words and trees over an infinite alphabet. In *Proceedings of CSL 2006*, volume 4207 of *Lecture Notes in Computer Science*, pages 41–57. Springer, 2006.

33 Wieslaw Szwast and Lidia Tendera. FO^2 with one transitive relation is decidable. In *30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013)*, volume 20 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 317–328, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.