Extending Two-Variable Logic on Trees*†

Bartosz Bednarczyk[1], Witold Charatonik[2], and Emanuel Kieroński[3]

1 **University of Wrocław, Wrocław, Poland**
bbednarczyk@stud.cs.uni.wroc.pl
2 **University of Wrocław, Wrocław, Poland**
wch@cs.uni.wroc.pl
3 **University of Wrocław, Wrocław, Poland**
kiero@cs.uni.wroc.pl

Abstract

The finite satisfiability problem for the two-variable fragment of first-order logic interpreted over trees was recently shown to be ExpSpace-complete. We consider two extensions of this logic. We show that adding either additional binary symbols or counting quantifiers to the logic does not affect the complexity of the finite satisfiability problem. However, combining the two extensions and adding both binary symbols and counting quantifiers leads to an explosion of this complexity. We also compare the expressive power of the two-variable fragment over trees with its extension with counting quantifiers. It turns out that the two logics are equally expressive, although counting quantifiers do add expressive power in the restricted case of unordered trees.

1998 ACM Subject Classification F.4 Mathematical Logic and Formal Languages

Keywords and phrases two-variable logic, trees, satisfiability, expressivity, counting quantifiers

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.11

1 Introduction

Two-variable logics. Two-variable logic, FO^2, is one of the most prominent decidable fragments of first-order logic. It is important in computer science because of its decidability and connections with other formalisms like modal, temporal and description logics or query languages. For example, it is known that FO^2 over words can express the same properties as unary temporal logic [10] and FO^2 over trees is precisely as expressive as the navigational core of XPath, a query language for XML documents [20]. The complexity of the satisfiability problem for FO^2 over words and trees, respectively, is studied in [10], and [2]. Namely, it is shown that its satisfiability problem over words is NExpTime-complete and over trees—ExpSpace-complete.

On the other hand, FO^2 cannot express that a structure is a word or a tree and it cannot express that a relation is transitive, an equivalence or an order. This led to extensive studies of FO^2 over various classes of structures, where some distinguished relational symbols are interpreted in a special way, e.g., as equivalences or linear orders. The finite satisfiability problem for FO^2 remains decidable over structures where one [17] or two relation symbols [18] are interpreted as equivalence relations; where one [21] or two relations are interpreted as linear orders [25, 27]; where two relations are interpreted as successors of two linear orders [19, 11, 8]; where one relation is interpreted as linear order, one as its successor

* Supported by the Polish National Science Centre grant No. 2016/21/B/ST6/01444.
† For missing proofs see [1].

and another one as equivalence [3]; where an equivalence closure can be applied to two binary predicates [16]; where deterministic transitive closure can be applied to one binary relation [6]. It is known that the finite satisfiability problem is undecidable for FO^2 with two transitive relations [14], with three equivalence relations [17], with one transitive and one equivalence relation [18], with three linear orders [15], with two linear orders and their two corresponding successors [19]. A summary of complexity results for extensions of FO^2 with binary predicates being the order relations can be found in [27].

In the context of extensions of FO^2 it is enough to consider relational signatures with symbols of arity at most two [12]. Some of the above mentioned decidability results, e.g., [2, 25, 19, 11, 3, 6], are obtained under the restriction that besides the distinguished binary symbols interpreted in a special way there are no other binary predicates in the signature; some, like [17, 18, 21, 8, 16, 27] are valid in the general setting. In undecidability results additional binary predicates are usually not necessary.

Another decidable extension of FO^2 is the two-variable fragment with *counting quantifiers*, C^2, where quantifiers of the form $\exists^{\leq k}$, $\exists^{=k}$, $\exists^{\geq k}$ are allowed. The finite satisfiability problem for C^2 was proved to be decidable and NExpTime-complete (both under unary and binary encoding of numbers in counting quantifiers) in [13, 22, 23]. There are also decidable extensions of C^2 with special interpretations of binary symbols: in [8] two relation symbols are interpreted as child relations in two forests (which subsumes the case of two successor relations on two linear orders), in [24] one symbol is interpreted as equivalence relation and in [7] one symbol is interpreted as linear order (and the case with two linear orders is undecidable).

Our contribution. In this paper we extend the main result from [2], namely ExpSpace-completeness of the satisfiability problem for FO^2 interpreted over finite trees without additional binary symbols. We consider two extensions of this logic. We show that adding either additional, uninterpreted binary symbols or counting quantifiers to the logic does not increase the complexity of the satisfiability problem. However, when we combine the two extensions and add both binary symbols and counting quantifiers then the complexity explodes and the problem is at least as hard as the emptiness problem for vector addition tree automata [9]. Since decidability of emptiness of vector addition tree automata is a long-standing open problem, showing decidability of C^2 over trees with additional binary symbols is unlikely in nearest future.

Let us recall that the situation is similar to the case of finite words: FO^2 with a linear order and the induced successor relation remains NExpTime-complete when extended either with additional binary relations [27] or with counting quantifiers [7]. Combining both additional ingredients gives a logic which is equivalent to the emptiness problem of multicounter automata [7], a problem which is known to be decidable, but for which no algorithm of elementary complexity is known.

We additionally compare the expressive power of the two-variable fragment over trees with its extension with counting quantifiers. It is not difficult to see that FO^2 over unordered trees cannot count and thus C^2 is strictly more expressive in this case. However, the presence of order in form of sibling relations gives FO^2 the ability of counting and makes the two logics equally expressive.

2 Preliminaries

2.1 Logics, trees and atomic types

We work with signatures of the form $\tau = \tau_0 \cup \tau_{nav} \cup \tau_{com}$, where τ_0 is a set of unary symbols, and $\tau_{nav} \subseteq \{\downarrow, \downarrow^+, \rightarrow, \rightarrow^+\}$ and τ_{com} are sets of binary symbols called, respectively, *navigational* binary symbols and *common* binary symbols. Over such signatures we consider two fragments of first-order logic: FO^2, i.e., the restriction of first-order logic in which only variables x and y are available, and its extension with *counting quantifiers*, C^2, in which quantifiers of the form $\exists^{\geq n}$, $\exists^{\leq n}$, for $n \in \mathbb{N}$ are allowed. We assume that the reader is familiar with their standard semantics. When measuring the length of formulas we assume binary encodings of numbers n in superscripts of quantifiers.

We write $\mathrm{FO}^2[\tau_{bin}]$ or $\mathrm{C}^2[\tau_{bin}]$ where $\tau_{bin} \subseteq \tau_{nav} \cup \tau_{com}$ to denote that the only binary symbols that are allowed in signatures are from τ_{bin}. We will mostly work with two logics: $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$, for τ_{com} being an arbitrary set of common binary symbols, and $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$, i.e., the fragment with counting quantifiers with no common binary symbols.

We are interested in finite unranked, ordered tree structures, in which the interpretation of the symbols from τ_{nav} is fixed: $\downarrow$ is interpreted as the child relation, $\rightarrow$ as the right sibling relation, and $\downarrow^+$ and $\rightarrow^+$ as their respective transitive closures. We read $u \downarrow w$ as "w is a *child* of u" and $u \rightarrow w$ as "w is the *right sibling* of u". We will also use other standard terminology like *ancestor*, *descendant*, *preceding-sibling*, *following-sibling*, etc. The interpretation of symbols from τ_{com} (if present) is not restricted.

We use $x \not\sim y$ to abbreviate the formula stating that x and y are in *free position*, i.e., that they are related by none of the navigational binary predicates available in the signature. Let us call the formulas specifying the relative position of a pair of elements in a tree with respect to binary navigational predicates *order formulas*. There are ten possible order formulas: $x{\downarrow}y$, $y{\downarrow}x$, $x{\downarrow^+}y \wedge \neg(x{\downarrow}y)$, $y{\downarrow^+}x \wedge \neg(y{\downarrow}x)$, $x{\rightarrow}y$, $y{\rightarrow}x$, $x{\rightarrow^+}y \wedge \neg(x{\rightarrow}y)$, $y{\rightarrow^+}x \wedge \neg(y{\rightarrow}x)$, $x \not\sim y$, $x{=}y$. They are denoted, respectively, as: $\theta_{\downarrow}$, $\theta_{\uparrow}$, $\theta_{\downarrow\downarrow^+}$, $\theta_{\uparrow\uparrow^+}$, $\theta_{\rightarrow}$, $\theta_{\leftarrow}$, $\theta_{\Rightarrow^+}$, $\theta_{\Leftarrow^+}$, $\theta_{\not\sim}$, $\theta_{=}$. Let Θ be the set of these ten formulas.

We use symbol $\mathfrak{T}$ (possibly with sub- or superscripts) to denote tree structures. For a given tree $\mathfrak{T}$ we denote by T its universe. A *tree frame* is a tree over a signature containing no unary predicates and no common binary predicates. We will sometimes say that a tree frame $\mathfrak{T}_f$ is the tree frame *of* $\mathfrak{T}$, or that $\mathfrak{T}$ is *based* on $\mathfrak{T}_f$ if $\mathfrak{T}_f$ is obtained from $\mathfrak{T}$ by dropping the interpretation of all unary and common binary symbols. We say that a formula φ is satisfiable over a tree frame if it has a model based on this tree frame.

Given a tree $\mathfrak{T}$, we say that a node $v \in T$ is a *minimal* node (having some fixed property) if there is no $w \in T$ (having this property) such that $\mathfrak{T} \models w{\downarrow^+}v$. A $\downarrow$-path ($\rightarrow$-path) is a sequence of nodes $v_1, \ldots, v_k$ such that $\mathfrak{T} \models v_i{\downarrow}v_{i+1}$ ($\mathfrak{T} \models v_i{\rightarrow}v_{i+1}$), for $i = 1, \ldots, k-1$. Given a $\downarrow$-path ($\rightarrow$-path) P we say that distinct nodes $v_1, \ldots, v_l$ (having some fixed property) are the l *smallest* elements (having this property) on P if for any other $v \in P$ (having this property) we have $\mathfrak{T} \models v_i{\downarrow^+}v$ ($\mathfrak{T} \models v_i{\rightarrow^+}v$) for $i = 1, \ldots, l$. Analogously we define *maximal* and *greatest* elements.

An (atomic) 1-*type* is a maximal satisfiable set of atoms or negated atoms with free variable x. Similarly, an (atomic) 2-*type* is a maximal satisfiable set of atoms or negated atoms with free variables x, y. Note that the numbers of atomic 1- and 2-types are bounded exponentially in the size of the signature. We often identify a type with the conjunction of all its elements. If we work with a signature with empty τ_{com} then 1-types correspond to subsets of τ_0. We denote by $\boldsymbol{\alpha}_\varphi$ the set of 1-types over the signature consisting of symbols appearing in φ.

For a given τ-tree $\mathfrak{T}$, and a node $v \in T$ we say that v *realizes* a 1-type α if α is the unique 1-type such that $\mathfrak{T} \models \alpha[v]$. We denote by $\mathrm{tp}^{\mathfrak{T}}(v)$ the 1-type realized by v. Similarly, for distinct $u, v \in T$, we denote by $\mathrm{tp}^{\mathfrak{T}}(u, v)$ the unique 2-type *realized* by the pair u, v, i.e. the type β such that $\mathfrak{T} \models \beta[u, v]$.

2.2 Normal forms

As usual when working with satisfiability of two-variable logics we employ a Scott-type normal form [26]. We start with its adaptation for the case of $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$.

▶ **Definition 1.** We say that an $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ formula φ is in *normal form* if

$$\varphi = \forall xy \chi(x, y) \wedge \bigwedge_{i=1}^{m} \forall x(\lambda_i(x) \Rightarrow \exists y(\theta_i(x, y) \wedge \chi_i(x, y))),$$

where $\lambda_i(x)$ is an atomic formula $A(x)$ for some unary symbol A, $\chi(x, y)$ and $\chi_i(x, y)$ are quantifier-free, $\chi_i(x, y)$ do not use symbols from τ_{nav}, and $\theta_i(x, y)$ is an order formula.

We remark that the equality symbol may be used in χ, e.g., we can force a model to contain at most one node satisfying A: $\forall xy(A(x) \wedge A(y) \Rightarrow x{=}y)$. The following lemma can be proved in a standard fashion (cf. e.g., [2]).

▶ **Lemma 2.** *Let φ be an $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ formula over a signature τ. There exists a polynomially computable $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ normal form formula φ' over signature τ' consisting of τ and some additional unary symbols, such that φ and φ' are satisfiable over the same tree frames.*

Consider a conjunct $\varphi_i = \forall x(\lambda_i(x) \Rightarrow \exists y(\theta_i(x, y) \wedge \chi_i(x, y)))$ of an $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ normal form formula φ. Let $\mathfrak{T} \models \varphi$, and let $v \in T$ be an element such that $\mathfrak{T} \models \lambda_i[v]$. Then an element $w \in T$ such that $\mathfrak{T} \models \theta_i[v, w] \wedge \chi_i[v, w]$ is called a *witness* for v and φ_i. We call w an *upper* witness if $\theta_i(v, w) \models w{\downarrow^+}v$, a *lower* witness if $\theta_i(v, w) \models v{\downarrow^+}w$, a *sibling* witness if $\theta_i(v, w) \models v{\rightarrow^+}w \vee w{\rightarrow^+}v$, and a *free* witness if $\theta_i(v, w) \models v \not\sim w$. We also sometimes simply speak about $\rightarrow^+$-witnesses, $\uparrow$-witnesses, etc.

For C^2 we use a similar but slightly different normal form. One obvious difference is that it uses counting quantifiers, the other is that its $\forall\exists$-conjuncts do not need to contain the θ_i-components, specifying the position of the required witnesses. Refining the normal form by incorporating those components is possible but seems to require an exponential blow-up.

▶ **Definition 3.** We say that a formula $\varphi \in \mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ is in *normal form*, if:

$$\varphi = \forall x \forall y\ \chi(x, y) \wedge \bigwedge_{i=1}^{m} \left(\forall x\ \exists^{\bowtie_i C_i} y\ \chi_i(x, y)\right),$$

where $\bowtie_i \in \{\leq, \geq\}$, each C_i is a natural number, and $\chi(x, y)$ and all $\chi_i(x, y)$ are quantifier-free.

▶ **Lemma 4** ([13])**.** *Let φ be a formula from $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ over a signature τ. There exists a polynomially computable $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ normal form formula φ' over signature τ' consisting of τ and some additional unary symbols, such that φ and φ' are satisfiable over the same tree frames.*

As in the case of $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ we speak about *witnesses*. Given a normal form $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ formula φ and a tree $\mathfrak{T} \models \varphi$, we say that a node $w \in T$ is a witness for $v \in T$ and a conjunct $\forall x\ \exists^{\bowtie_i C_i} y\ \chi_i(x, y)$ of φ if $\mathfrak{T} \models \chi_i[v, w]$. If additionally $\mathfrak{T} \models w{\downarrow^+}v$ then w is an *upper* witness, if $\mathfrak{T} \models v{\downarrow^+}w$ then w is a *lower* witness, and so on.

In Section 3, when a normal form formula φ is considered we always assume that it is as in Definition 1. In particular we allow ourselves, without explicitly recalling the shape of φ, to refer to its parameter m and components $\chi, \chi_i, \lambda_i, \theta_i$. Analogously, in Section 4 we assume that any normal form φ is as in Definition 3.

3 FO^2 on trees with additional binary relations

In this section we show that the complexity of the satisfiability problem for $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ [2] is retained when the logic is extended with additional, uninterpreted binary relations. Thus we combine here the logic from [2] with the logic from [12]. It means that we need not only to ensure that an element can see realizations of appropriate 1-types in appropriate positions, but also that it is related to them by uninterpreted binary relations in a specific way. In our approach we combine the cutting arguments from [2] with the careful strategy of ensuring witnesses, similar to that from [12] or [27].

▶ **Theorem 5.** *The satisfiability problem for* $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ *over finite trees is* ExpSpace*-complete.*

The lower bound is inherited from $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$. For the upper bound we show that any satisfiable formula φ has a model of depth and degree bounded exponentially in $|\varphi|$. Then we show an auxiliary result allowing us to restrict attention to models in which there is a small number of elements that serve as free witnesses for all elements of the tree. We finally design an alternating exponential time procedure searching for such small models.

3.1 Small model property

Let $\mathfrak{f}$ be a fixed function, which for a given normal form $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ formula φ returns $96m^3|\boldsymbol{\alpha}_\varphi|^3$. Recall that m is the number of $\forall\exists$-conjuncts of φ and $\boldsymbol{\alpha}_\varphi$ is the set of 1-types over the signature of φ. We will use $\mathfrak{f}$ to estimate the length of paths and the degree of nodes in models. Note that for a given φ the value returned by $\mathfrak{f}$ is exponentially bounded in $|\varphi|$. It should be mentioned that by a more careful analysis one could obtain slightly better bounds (still exponential in $|\varphi|$), but $\mathfrak{f}$ is sufficient for our purposes and allows for a reasonably simple presentation. The following small model property is crucial for obtaining an ExpSpace-upper bound on the complexity of the satisfiability problem. It can be seen as an extension of Theorem 3.3 from [2], where a similar result was proved for FO^2 over trees without additional binary relations.

▶ **Theorem 6** (Small model theorem)**.** *Let φ be a satisfiable normal form formula from* $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$*. Then φ has a model in which the length of every $\downarrow$-path and the degree of each node are bounded exponentially in $|\varphi|$ by $\mathfrak{f}(\varphi)$.*

The proof is split into two lemmas. In the first one we show how to shorten the $\downarrow$-paths and in the second how to reduce the degree of nodes, i.e., to shorten $\rightarrow$-paths.

▶ **Lemma 7.** *Let φ be a normal form* $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ *formula and $\mathfrak{T}$ its model. Then there exists a tree model $\mathfrak{T}'$ for φ whose every $\downarrow$-path has length at most $\mathfrak{f}(\varphi)$.*

Proof. Assume that $\mathfrak{T}$ contains a $\downarrow$-path $P = (v_1, v_2, \ldots, v_n)$ longer than $\mathfrak{f}(\varphi)$. We show that then it is possible to remove some nodes from $\mathfrak{T}$ and obtain a smaller model $\mathfrak{T}_0$. For a node $u \in T$ we define its *projection onto* P as the greatest node $v \in P$, such that $\mathfrak{T} \models v \downarrow^+ u$.

We first distinguish a set W of some relevant elements of $\mathfrak{T}$. W will consist of four disjoint sets W_0, W_1, W_2, W_3. For each 1-type α we mark:

(i) m greatest and m smallest realizations of α on P (or all realizations of α on P if there are less than m of them);

(ii) m realizations of α outside P having greatest projections onto P and m realizations of α outside P having smallest projections onto P (or all realizations of α outside P if there are less than m of them).

Let W_0 be the set consisting of all the marked elements. Let W_1 be a minimal (in the sense of $\subseteq$) set of nodes of $\mathfrak{T}$ such that all the elements from W_0 have all the required witnesses in $W_0 \cup W_1$. Similarly, let W_2 be a minimal set of nodes of $\mathfrak{T}$ such that all the elements from W_1 have all the required witnesses in $W_0 \cup W_1 \cup W_2$. Finally, let W_3 be the set of those projections onto P of elements of $W_0 \cup W_1 \cup W_2$ which are not in $W_0 \cup W_1 \cup W_2$. Let $W := W_0 \cup W_1 \cup W_2 \cup W_3$. To estimate the size of W, observe that $|W_0| \leq 4m|\boldsymbol{\alpha}_\varphi|$, $|W_1| \leq m|W_0|$, $|W_2| \leq m|W_1|$ and $|W_3| \leq |W_0 \cup W_1 \cup W_2|$. Thus $|W| \leq 24m^3|\boldsymbol{\alpha}_\varphi|$.

An *interval* of P *of length* s is a sequence of nodes of the form $(v_i, v_{i+1}, \ldots, v_{i+s-1})$ for some i. We claim that P contains an interval I of length at least $2|\boldsymbol{\alpha}_\varphi|^2 + 2$ having no elements in W. For assume to the contrary that there there is no such interval. Note that the extremal points of P (which are the root and a leaf of $\mathfrak{T}$) are members of W. Hence the points of $W \cap P$ determine at most $|W| - 1$ maximal (possibly empty) intervals not containing elements of W. It follows that $|P| \leq (|W| - 1)(2|\boldsymbol{\alpha}_\varphi|^2 + 1) + |W| < |W|(2|\boldsymbol{\alpha}_\varphi|^2 + 2)$, which by routine calculations gives $|P| < 96m^3|\boldsymbol{\alpha}_\varphi|^3$, a contradiction.

Using the pigeonhole principle we can easily see that in I there are two disjoint pairs of nodes v_k, v_{k+1} and v_l, v_{l+1}, for some $k < l$ such that $\mathrm{tp}^{\mathfrak{T}}(v_{l+i}) = \mathrm{tp}^{\mathfrak{T}}(v_{k+i})$, for $i = 0, 1$. We build a tree $\mathfrak{T}_0$ by replacing in $\mathfrak{T}$ the subtree rooted at v_{k+1} by the subtree rooted at v_{l+1}, setting $\mathrm{tp}^{\mathfrak{T}_0}(v_k, v_{l+1}) := \mathrm{tp}^{\mathfrak{T}}(v_k, v_{k+1})$ and for each v being a sibling of v_{k+1} in $\mathfrak{T}$ setting $\mathrm{tp}^{\mathfrak{T}_0}(v, v_{l+1}) := \mathrm{tp}^{\mathfrak{T}}(v, v_{k+1})$ (all the remaining 2-types are retained from $\mathfrak{T}$). In effect, all the subtrees rooted at elements of P between v_{k+1} and v_l are removed from $\mathfrak{T}$. Note that all elements of W survive our surgery. This guarantees that the elements of $W_0 \cup W_1$ retain all their witnesses. However, some nodes v from $T_0 \setminus (W_0 \cup W_1)$ could lose their witnesses. We can now reconstruct them using the nodes from W_0. This is done by distinguishing several cases. Here we analyse just one of them.

Case 1: $v = v_k$. All the siblings, ancestors and elements in free position to v_k from $\mathfrak{T}$ are retained in $\mathfrak{T}_0$. Thus v_k retains all its sibling, ancestor and free witnesses. There is also no problem with $\downarrow$-witnesses, as v_k retains all its children except v_{k+1}, and v_{k+1} is replaced by v_{l+1} having the same 1-type and connected to v_k exactly as v_{k+1} was. Some $\downarrow\downarrow^+$-witnesses for v_k could be lost however. Let B be a minimal (in the sense of $\subseteq$) set of elements providing the required $\downarrow\downarrow^+$-witnesses for v_k in $\mathfrak{T}$. Note that $|B| \leq m$. Let α be a 1-type realized in B. If all elements of 1-type α from B are in W_0 then there is nothing to do: they survive, and serve as proper $\downarrow\downarrow^+$-witnesses for v_k in $\mathfrak{T}_0$. Otherwise, there must be at least m realizations of α in W_0 (on P or outside P) whose projections onto P in $\mathfrak{T}$ are below v_{l+2}. We can modify the 2-types joining v_k with some of them securing the required $\downarrow\downarrow^+$-witnesses for v_k. This can be done without conflicts, since $v_k \notin W_0 \cup W_1$ and hence it is not required as a witness by any element of W_0.

The remaining cases can be treated similarly.

After the described adjustments all the elements of $\mathfrak{T}_0$ have appropriate witnesses. Since all the 2-types realized in $\mathfrak{T}_0$ are also realized in $\mathfrak{T}$ this ensures that the $\forall\forall$ conjunct of φ is not violated in $\mathfrak{T}_0$. Thus $\mathfrak{T}_0 \models \varphi$.

Note that the number of nodes of $\mathfrak{T}_0$ is strictly smaller than the number of nodes of $\mathfrak{T}$. We can repeat the same shrinking process starting from $\mathfrak{T}_0$, and continue it, obtaining eventually a model $\mathfrak{T}'$ whose paths are bounded as required. ◀

▶ **Lemma 8.** *Let φ be a normal form $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ formula and $\mathfrak{T} \models \varphi$. Then there exists a model $\mathfrak{T}' \models \varphi$, obtained by removing some subtrees from $\mathfrak{T}$ such that the degree of its every node is bounded by $\mathfrak{f}(\varphi)$.*

Proof. Assume that $\mathfrak{T}$ contains a node v having more than $\mathfrak{f}(\varphi)$ children. We show that then it is possible to remove some of these children together with the subtrees rooted at them and obtain a smaller model $\mathfrak{T}' \models \varphi$. The process is similar to the one described in the proof of Lemma 7. Let $P = (v_1, \ldots, v_k)$ be the $\rightarrow$-path in $\mathfrak{T}$ consisting of all the children of v. We first distinguish a set W of some relevant elements of $\mathfrak{T}$. It will consist of four disjoint sets W_0, W_1, W_2, W_3.

For each 1-type α we mark m greatest and m smallest realizations of α on P (or all realizations of α on P if there are less than m of them). Further we choose $m+1$ elements of P having a realization of α as a descendant (or all such elements if there are less than $m+1$ of them) and for each of them mark one descendant of 1-type α. Let W_0 be the set consisting of all the marked elements. Let W_1 be a minimal set of nodes such that all the elements from W_0 have all the required witnesses in $W_0 \cup W_1$. Similarly, let W_2 be a minimal set of nodes such that all the elements from W_1 have all the required witnesses in $W_0 \cup W_1 \cup W_2$. Finally, let W_3 be the set of those elements of P which are not in $W_0 \cup W_1 \cup W_2$ but have an element from $W_0 \cup W_1 \cup W_2$ in their subtree. Let $W := W_0 \cup W_1 \cup W_2 \cup W_3$. To estimate the size of W, observe that $|W_0| \leq (3m+1)|\boldsymbol{\alpha}_\varphi|$, $|W_1| \leq m|W_0|$, $|W_2| \leq m|W_1|$ $|W_3| \leq |W_0 \cup W_1 \cup W_2|$. Thus, after simple estimations, we have $|W| \leq 24m^3|\boldsymbol{\alpha}_\varphi|$.

An *interval* of P *of length* s is a sequence of nodes of the form $(v_i, v_{i+1}, \ldots, v_{i+s-1})$ for some i. Using arguments similar to those from the proof of Lemma 7 we can show that P contains an interval I with no elements in W, in which there are two disjoint pairs of nodes v_k, v_{k+1} and v_l, v_{l+1}, for some $k < l$ such that $\mathrm{tp}^{\mathfrak{T}}(v_{l+i}) = \mathrm{tp}^{\mathfrak{T}}(v_{k+i})$, for $i = 0, 1$. We build an auxiliary tree $\mathfrak{T}_0$ by removing the subtrees rooted at $v_{k+1}, \ldots, v_l$ and setting $\mathrm{tp}^{\mathfrak{T}_0}(v_k, v_{l+1}) := \mathrm{tp}^{\mathfrak{T}}(v_k, v_{k+1})$ (all the remaining 2-types are retained from $\mathfrak{T}$). Again the elements which lost their witnesses in our construction can regain them by changing their connections to elements from W_0. And again, as in the proof of Lemma 7, the process can be continued until a model with appropriately bounded degree of nodes is obtained. ◀

3.2 Global free witnesses

The small model property from the previous subsection is a crucial step towards an exponential space algorithm for satisfiability. However, it allows for models having doubly exponentially many nodes, which thus cannot be stored in memory. In the case of FO^2 without additional binary relations [2] the corresponding algorithm traversed $\downarrow$-paths guessing for each node v its *full type* storing the sets of 1-types of elements above, below, and in free position to v, similarly to the case of FO^2 with counting from Section 4. Then it took care of *realizing* such full types. This approach would not be sufficient for our current purposes, since the presence of additional binary relations requires us not only to ensure that appropriate 1-types of elements will appear above, below and in free position to a node but also that appropriate 2-types will be realized. This is especially awkward when dealing with free witnesses, since for a given node they are located on different paths. To overcome this difficulty we show that we always can assume that all elements have their free witnesses in a small, exponentially bounded fragment of some model.

▶ **Lemma 9.** *Let φ be a normal form $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ formula and $\mathfrak{T}$ its model. Let h be the length of the longest $\downarrow$-path in $\mathfrak{T}$ and d the maximal number of $\downarrow$-successors of a node.*

Then there exists a tree $\mathfrak{T}'$ and a set of nodes $F \subseteq T'$, called a global set of free witnesses *such that:*

- *the universes, the 1-types of all elements and the tree frames of $\mathfrak{T}$ and $\mathfrak{T}'$ are identical,*
- $\mathfrak{T}' \models \varphi$,
- *the size of F is bounded by $3(m+1)^3h^2d^2|\boldsymbol{\alpha}_\varphi|$,*
- *F is closed under $\uparrow$, $\leftarrow$ and $\rightarrow$,*
- *for each conjunct of φ of the form $\varphi_i = \forall x(\lambda_i(x) \to \exists y(x \not\sim y \wedge \chi(x,y)))$ and each node $v \in T'$, if $\mathfrak{T}' \models \lambda_i[v]$ then there is a witness for v and φ_i in F.*

Proof. We first describe a procedure which distinguishes in $\mathfrak{T}$ the desired set F. This will contain three disjoint subsets F_0, F_1, F_2. Start with $F_0 = F_1 = F_2 = \emptyset$. For each 1-type α choose $m+1$ maximal elements of type α in $\mathfrak{T}$ (or all of them if there are less than $m+1$ such elements) and make them members of F_0. Close F_0 under $\uparrow$, $\leftarrow$ and $\rightarrow$, i.e., for each member of F_0 add to F_0 also all its ancestors, siblings and all the siblings of its ancestors. This finishes the construction of F_0. Observe that $|F_0| \leq (m+1)hd|\boldsymbol{\alpha}_\varphi|$.

For each $v \in F_0$ and each conjunct of φ of the form $\varphi_i = \forall x(\lambda_i(x) \to \exists y(x \not\sim y \wedge \chi(x,y)))$ if $\mathfrak{T} \models \lambda_i[v]$ and there is no witness for v and φ_i in F_0 then find one in $\mathfrak{T}$ and add it to F_1. Similarly, For each $v \in F_1$ and each conjunct of φ of the form $\varphi_i = \forall x(\lambda_i(x) \to \exists y(x \not\sim y \wedge \chi(x,y)))$ if $\mathfrak{T} \models \lambda_i[v]$ and there is no witness for v and φ_i in $F_0 \cup F_1$ then find one in $\mathfrak{T}$ and add it to F_2.

Take as F the smallest set containing $F_0 \cup F_1 \cup F_2$ and closed under the relations $\uparrow$, $\leftarrow$ and $\rightarrow$. Note that $|F_1| \leq m|F_0| \leq m(m+1)hd|\boldsymbol{\alpha}_\varphi|$, and similarly $|F_2| \leq m|F_1| \leq m^2(m+1)hd|\boldsymbol{\alpha}_\varphi|$. This allows us to estimate the size of F as follows, $|F| \leq (m+1)hd|\boldsymbol{\alpha}_\varphi| + \big(m(m+1)hd|\boldsymbol{\alpha}_\varphi| + m^2(m+1)hd|\boldsymbol{\alpha}_\varphi|\big)hd \leq 3(m+1)^3h^2d^2|\boldsymbol{\alpha}_\varphi|$, as required.

To obtain $\mathfrak{T}'$ we modify some 2-types joining pairs of elements in free position, one of which is in $T \setminus (F_0 \cup F_1)$ and the other in F_0. Consider any element $v \in T \setminus (F_0 \cup F_1)$ and let B be a minimal (with respect to $\subseteq$) set of elements providing the required free witnesses for v in $\mathfrak{T}$. Note that $|B| \leq m$. Let α be a 1-type realized in B. If all elements of 1-type α from B are in F_0 then there is nothing to do: we just retain the connections of v with the elements of type α in F_0. Otherwise there are $m+1$ maximal realizations of α in F_0, and at least m of them is in free position to v. Indeed, $v \notin F_0$ and thus it cannot be an ancestor or a sibling of any of those $m+1$ maximal realizations of α (since F_0 is closed under $\uparrow$, $\leftarrow$ and $\rightarrow$), so if it is not in free position to all then it is a descendant of one of them. But in this case it is in free position to all the other (since maximal realizations of α are in free position to each other). Thus, in this case, for any $w \in B$ of type α we can choose a fresh w' of type α in F_0 in free position to v and set $\mathrm{tp}^{\mathfrak{T}'}(v,w') := \mathrm{tp}^{\mathfrak{T}}(v,w)$. We repeat this step for all 1-types of elements of B, thus ensuring that v has all the required free witnesses in F_0. We repeat this process for all elements of $T \setminus (F_0 \cup F_1)$.

This finishes our construction of $\mathfrak{T}'$. Note that our surgery does not affect the 2-types inside $\mathfrak{T}\restriction(F_0 \cup F_1)$ and the 2-types joining the elements of F_1 with the elements of $T \setminus (F_0 \cup F_1)$. Thus in $\mathfrak{T}'$ all elements of $F_0 \cup F_1$ retain their free witnesses in F and all the remaining elements have appropriate free witnesses in F_0 due to our construction. As we do not change the 2-types joining the elements which are not in free position thus all the upper, lower and sibling witnesses are retained in $\mathfrak{T}'$. Since $\mathfrak{T}'$ realizes only 2-types realized in $\mathfrak{T}$ the universal conjunct of $\forall xy\chi(x,y)$ of φ is satisfied in $\mathfrak{T}'$. Hence, $\mathfrak{T}' \models \varphi$. ◀

3.3 The algorithm

We are now ready to present an alternating algorithm for the finite satisfiability problem for $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$, working in exponential time. Since AExpTime=ExpSpace this justifies Thm. 5. Due to Lemma 2 we can assume that the input formula is given in normal form.

We first sketch our approach. For a given normal form φ the algorithm attempts to build a model $\mathfrak{T} \models \varphi$. It first guesses its fragment $\mathfrak{F}$, of size exponentially bounded in $|\varphi|$, intended to provide free witnesses for all elements of $\mathfrak{T}$, and then expands it down. Namely, it universally chooses one of the leaves v of $\mathfrak{F}$, guesses all its children $w_1, \ldots, w_k$ (at most exponentially many), and guesses 2-types joining w_i-s with all their ancestors, with all elements of $\mathfrak{F}$, and among each other. The algorithm verifies that the guessed elements are consistent with the partial model constructed so far, and if so it universally chooses one of w_i and proceeds with w_i analogously like with v. This process is continued until the algorithm decides that a leaf of $\mathfrak{T}$ is reached.

We must ensure that the structure $\mathfrak{T}$ which is constructed by our algorithm is indeed a model of φ, i.e., all elements of $\mathfrak{T}$ have appropriate witnesses for $\forall\exists$ conjuncts, and that no pair of elements of $\mathfrak{T}$ violates the $\forall\forall$ conjunct. Note that when the algorithm inspects a node v all its siblings and ancestors are present in the memory. This allows to verify that v has the required upper and sibling witnesses. Checking the existence of free witnesses is not problematic too, because, owing to Lemma 9 we assume that they are provided by $\mathfrak{F}$, which is never removed from the memory. Verifying $\downarrow$-witnesses is also straightforward, since we guess all the children $w_1, \ldots, w_k$ of v at once. To deal with $\downarrow^+$-witnesses the algorithm stores some additional data. Namely, together with each w_i it guesses the list of all 2-types (called *promised* 2-*types*) which will be assigned to the pairs consisting of v or its ancestor and a descendant of w_i. This is obviously sufficient to see if v will have the required $\downarrow^+$-witnesses. The algorithm will take care of the consistency of the information about promised types stored in various nodes, and then ensure that all the promised 2-types will indeed be realized.

Turning to the problem of verifying that the universal conjunct of φ is not violated by any pair of elements of $\mathfrak{T}$ note that it is easy for pairs of elements which are not in free position, since at some point during the execution of the algorithm they are both present in the memory and their 2-type is then available. For a pair of elements u_1, u_2 in free position there is an element v such that u_1, u_2 are descendants of two different children of v from the list $w_1, \ldots, w_k$. From information about the promised 2-types guessed together with w_i-s, we can extract the list of 1-types that will appear below each of w_i. Reading this information we see that the 1-types of u_1 and u_2 will appear in free position, and we just need to verify that there is a 2-type consistent with the $\forall\forall$-conjunct which can join them.

A more detailed description of the algorithm together with arguments for its correctness is given in Appendix A.

4 $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ on trees

In this section we prove that the finite satisfiability problem for $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ over trees is ExpSpace-complete. Intuitively, the upper bound proof is a combination of the two proofs from [5] and [7] that solve the problem for FO^2 on trees and for C^2 on linear orders respectively (note that a linear order is just a tree whose each node has at most one child). However, the method in [5] heavily depends on the normal form from Definition 1 where each conjunct corresponds to at most one relative position $\theta \in \Theta$. Although it is possible to bring a $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ formula into an analogous normal form, it seems to require a doubly

exponential blowup (recall that we assume binary coding of the numbers C_i and observe that the number of possible divisions of a set of C_i witnesses into 10 subsets corresponding to 10 order formulas is exponential in C_i). Therefore, to keep the complexity under control, we stay with the usual, less refined normal form from Definition 3, but to compensate it we introduce a new idea combining type information with witness counting.

4.1 Multisets

Any element of a model of a normal form conjunct $\forall x \exists^{\bowtie C_i} y\ \chi$ may require up to C_i witnesses, so we are interested in *multisets* counting these witnesses. To simulate counting up to the value k, we use the function $cut_k : \mathbb{N} \to \{0, 1, 2, \ldots, k, \infty\}$, where $cut_k(i) = i$ for $i \leq k$ and $cut_k(i) = \infty$ otherwise.

Formally, for a given $k \in \mathbb{N}$, a *k-multiset* M of elements from a set S is a function $M : S \to \{0, 1, 2, \ldots, k, \infty\}$. For every element $e \in S$ we interpret $M(e)$, called the *multiplicity* of e in M, as the number of occurrences of e in the multiset M, counted up to k. We employ standard set-theoretic operations, i.e., union $\cup$ and intersection $\cap$ with their natural semantics defined as follows: for given multisets A and B and an arbitrary element e from their domains, we define $(A \cup B)(e) = cut_k\,(A(e) + B(e))$ and $(A \cap B)(e) = \min(A(e), B(e))$. Additionally, we define the empty multiset $\emptyset$ as the function that for any argument returns 0 and the singleton $\{e\}$ of e as the function such that $\{e\}(e) = 1$ and $\{e\}(e') = 0$ for all $e' \neq e$.

4.2 Full types, witness counting and reduced types

We abstract information about nodes in a tree using the following notion.

▶ **Definition 10** (Full type)**.** A *k-full type* $\overline{\alpha}$ (over a signature $\tau = \tau_0 \cup \tau_{nav}$) is a function of type $\overline{\alpha} : \Theta \to \{0, 1, 2, \ldots, k, \infty\}^{2^{\tau_0}}$ (a function which takes a position from Θ and returns a k-multiset of 1-types over τ), that satisfies the following conditions:

- $\overline{\alpha}(\theta_\uparrow)$, $\overline{\alpha}(\theta_\rightarrow)$, $\overline{\alpha}(\theta_\leftarrow)$ is either empty or a singleton,
- $\overline{\alpha}(\theta_=)$ is a singleton, and
- if $\overline{\alpha}(\theta_\uparrow)$ (respectively, $\overline{\alpha}(\theta_\downarrow)$, $\overline{\alpha}(\theta_\rightarrow)$, $\overline{\alpha}(\theta_\leftarrow)$) is empty, then also the multiset $\overline{\alpha}(\theta_{\uparrow\uparrow^+})$ (respectively, $\overline{\alpha}(\theta_{\downarrow\downarrow^+}), \overline{\alpha}(\theta_{\rightrightarrows^+}), \overline{\alpha}(\theta_{\leftleftarrows^+})$) is empty.

Let C be the function that for a given normal form φ (cf. Def. 3) returns $C(\varphi) = \max\{C_i\}_{1 \leq i \leq m}$. We work with k-full types usually in contexts in which a normal form φ is fixed, and we are then particularly interested in $C(\varphi)$-full types. The purpose of a k-full type is to say for a given node v, for each $\theta \in \Theta$ and each 1-type α', how many vertices (counting up to k) of 1-type α' are in position θ to v. Formally:

▶ **Definition 11.** For a given tree $\mathfrak{T}$ and $v \in T$ we denote by $\mathrm{ftp}_k^{\mathfrak{T}}(v)$ the unique k-full type *realized* by v, i.e., the k-full type $\overline{\alpha}$ such that $\overline{\alpha}(\theta_=)$ contains the 1-type of v and for all positions $\theta \in \Theta$ and for all atomic 1-types α' we have that

$$\overline{\alpha}(\theta)(\alpha') = cut_k\left(\#\{w \in T : \mathfrak{T} \models \theta[v, w] \wedge \mathrm{tp}^{\mathfrak{T}}(w) = \alpha'\}\right)$$

where $\#S$ denotes the cardinality of the set S.

We next define functions which for a normal form φ and a $C(\varphi)$-full type $\overline{\alpha}$ say how many witnesses a realization of $\overline{\alpha}$ has for each of the $\forall\exists$ conjuncts of φ (recall that m is the number of such conjuncts) in all possible positions θ.

▶ **Definition 12** (Witness counting functions). Let φ be a normal form formula, and let $\overline{\alpha}$ be a $C(\varphi)$-full type. Assume that $\overline{\alpha}(\theta_{=}) = \{\alpha\}$. We associate with φ and $\overline{\alpha}$ a function $W^{\varphi}_{\overline{\alpha}} : \{1,\ldots,m\} \times \Theta \to \{0,1,\ldots,C(\varphi),\infty\}$, whose values are defined in the following way:

- for $\theta \in \{\theta_{=}, \theta_{\rightarrow}, \theta_{\leftarrow}, \theta_{\downarrow}, \theta_{\uparrow}\}$ and any i:

$$W^{\varphi}_{\overline{\alpha}}(i,\theta) = \begin{cases} 1 & \text{if } \overline{\alpha}(\theta)=\{\alpha'\} \text{ and } \alpha(x)\wedge\alpha'(y)\wedge\theta(x,y)\models\chi_i(x,y) \\ 0 & \text{otherwise,} \end{cases}$$

- for $\theta \in \{\theta_{\leftleftarrows^+}, \theta_{\rightrightarrows^+}, \theta_{\downarrow\downarrow^+}, \theta_{\uparrow\uparrow^+}, \theta_{\nearrow}\}$ and any i:

$$W^{\varphi}_{\overline{\alpha}}(i,\theta) = cut_{C(\varphi)}\left(\sum_{\alpha'\in A_{\alpha,\theta,i}} (\overline{\alpha}(\theta))(\alpha')\right),$$

where $A_{\alpha,\theta,i} = \{\alpha' :\ \alpha(x)\wedge\alpha'(y)\wedge\theta(x,y) \models \chi_i(x,y)\}$.

This way $W^{\varphi}_{\overline{\alpha}}(i,\theta)$ is the number of witnesses (counted up to $C(\varphi)$), in relative position θ, for a node of full type $\overline{\alpha}$ and the formula χ_i from φ.

Now we relate the notion of full types with the satisfaction of normal form formulas.

▶ **Definition 13** (φ-consistency). Let φ be a $\mathrm{C}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+]$ formula in normal form. Let $\overline{\alpha}$ be a $C(\varphi)$-full type. Assume that $\overline{\alpha}(\theta_{=})$ consists of a 1-type α. We say that $\overline{\alpha}$ is *φ-consistent* if it satisfies the following conditions.

- $\alpha(x) \models \chi(x,x)$,
- $\alpha(x) \wedge \alpha'(y) \wedge \theta(x,y) \models \chi(x,y)$ for all $\theta \in \Theta$ and all $\alpha' \in \overline{\alpha}(\theta)$, and
- for all $1 \leq i \leq m$ the inequality $\sum_{\theta\in\Theta} W^{\varphi}_{\overline{\alpha}}(i,\theta) \bowtie_i C_i$ holds.

Proving the following lemma is routine.

▶ **Lemma 14.** *Assume that a formula $\varphi \in \mathrm{C}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+]$ is in normal form. Then $\mathfrak{T} \models \varphi$ iff every $C(\varphi)$-full type realized in $\mathfrak{T}$ is φ-consistent.*

The next notion will be used to describe information from full types in a (lossy) compressed form. We need this form to obtain tight complexity bounds.

▶ **Definition 15** (φ-reduced type). Let φ be a normal form $\mathrm{C}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+]$ formula. For a given $C(\varphi)$-full type $\overline{\alpha}$, its φ-reduced form, $\mathrm{rftp}_{\varphi}(\overline{\alpha})$, is the tuple $\left(\alpha, W^{\varphi}_{\overline{\alpha}}, A, B, F\right)$, where $A = \overline{\alpha}(\theta_{\uparrow}) \cup \overline{\alpha}(\theta_{\uparrow\uparrow^+})$, $B = \overline{\alpha}(\theta_{\downarrow}) \cup \overline{\alpha}(\theta_{\downarrow\downarrow^+})$, $F = \overline{\alpha}(\theta_{\rightarrow}) \cup \overline{\alpha}(\theta_{\leftarrow}) \cup \overline{\alpha}(\theta_{\rightrightarrows^+}) \cup \overline{\alpha}(\theta_{\leftleftarrows^+}) \cup \overline{\alpha}(\theta_{\nearrow})$ and $\overline{\alpha}(\theta_{=})$ is the singleton of the 1-type α. If the $C(\varphi)$-full type $\overline{\alpha}$ is realized by a vertex v in $\mathfrak{T}$ then we say that $\mathrm{rftp}_{\varphi}(\overline{\alpha})$ is the φ-reduced type of v. This reduced full type will be denoted also as $\mathrm{rftp}^{\mathfrak{T}}_{\varphi}(v)$.

Intuitively, if a k-full type $\overline{\alpha}$ is realized by a vertex v in a structure $\mathfrak{T}$ then the multisets A, B, F in $\mathrm{rftp}_{\varphi}(\overline{\alpha})$ are respectively the k-multisets of 1-types realized in $\mathfrak{T}$ above, below and in a "non-vertical" position to v.

Let $\overline{\alpha}, \overline{\beta}$ be k-full types. A *combined k-full type* is a k-full type $\overline{\gamma}$, such that $\overline{\gamma}(\theta) = \overline{\alpha}(\theta)$ or $\overline{\gamma}(\theta) = \overline{\beta}(\theta)$ for all positions $\theta \in \Theta$.

▶ **Lemma 16.** *Let $\overline{\alpha}, \overline{\beta}$ be φ-consistent $C(\varphi)$-full types such that their φ-reduced forms are equal. Then the combined $C(\varphi)$-full type $\overline{\gamma}$ of the form $\overline{\gamma}(\theta) = \overline{\alpha}(\theta)$ for $\theta \in \{\theta_{\uparrow}, \theta_{\uparrow\uparrow^+}, \theta_{\rightarrow}, \theta_{\rightrightarrows^+}, \theta_{\nearrow}, \theta_{\leftarrow}, \theta_{\leftleftarrows^+}\}$ and $\overline{\gamma}(\theta) = \overline{\beta}(\theta)$ for $\theta \in \{\theta_{=}, \theta_{\downarrow}, \theta_{\downarrow\downarrow^+}\}$ is also φ-consistent.*

Proof. Obviously $\overline{\gamma}$ satisfies the first two conditions from Definition 13 because $\overline{\alpha}$ and $\overline{\beta}$ do. The third condition is guaranteed by the equality of the witness counting components. ◀

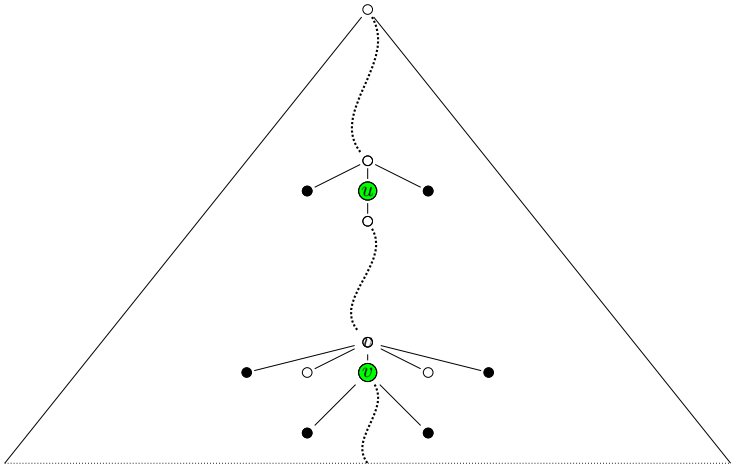

Figure 1 Naive combination of full types.

▶ **Example 17.** Let us observe that in the above lemma the assumption about equality of φ-reduced full types, and in particular their witness counting components, is essential. In [5, Proposition 2] the authors prove that in the setting without counting quantifiers a combined type remains φ-consistent without the assumption about equality of the reduced forms of the original types. The following example shows that in our scenario it is no longer true.

Let φ be a formula saying that every green vertex has at most three direct black neighbors below, on the left or on the right; formally φ is defined as

$$\forall x \exists^{\leq 3} y \ (\mathit{green}(x) \Rightarrow (\mathit{black}(y) \wedge (x \downarrow y \vee x \rightarrow y \vee y \rightarrow x))).$$

Let $\mathfrak{T}$ be a tree model from Fig. 1. Denote $\overline{\alpha} = \mathrm{ftp}^{\mathfrak{T}}_{C(\varphi)}(u)$ and $\overline{\beta} = \mathrm{ftp}^{\mathfrak{T}}_{C(\varphi)}(v)$. Because $\mathfrak{T} \models \varphi$, the $C(\varphi)$-full types $\overline{\alpha}$ and $\overline{\beta}$ are φ-consistent. However the combined $C(\varphi)$-full type $\overline{\gamma}$, in form described in Lemma 16, is not φ-consistent (the black nodes appear in $\overline{\gamma}$ at positions $\theta_{\downarrow}$, $\theta_{\leftarrow}$, $\theta_{\rightarrow}$ four times in total).

4.3 Small model theorem

The general scheme of the decidability proof of finite satisfiability of $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ is similar to the one from Section 3. Namely, we demonstrate the small-model property of the logic, showing that every satisfiable formula φ has a tree model of depth and degree bounded exponentially in $|\varphi|$. It is also obtained in a similar way, by first shortening $\downarrow$-paths and then shortening the $\rightarrow$-paths. The technical details differ however.

Recall that given a normal form φ we denote by m the number of its $\forall\exists$ conjuncts, and by $\boldsymbol{\alpha}_\varphi$ the set of 1-types over the signature consisting of the symbols appearing in φ.

▶ **Theorem 18** (Small model theorem). *Let φ be a formula of $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ in normal form. If φ is satisfiable then it has a a tree model in which every path has length bounded by $3 \cdot (C(\varphi)+2)^{10m+1} \cdot |\boldsymbol{\alpha}_\varphi|^2$ and every vertex has degree bounded by $(4C(\varphi)^2 + 8C(\varphi)) \cdot |\boldsymbol{\alpha}_\varphi|^5$.*

We split the proof of this theorem into two parts. First, in Lemmas 19 and 20, we show how to reduce the length of paths in a tree and then, in Lemma 21, we show how to reduce the degree of every vertex. We skip most of the details of the proofs due to the space limit.

▶ **Lemma 19** (Cutting lemma). *Let $\varphi \in \mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ be a formula in normal form and $\mathfrak{T}$ be its model. If there are two vertices $u, v \in T$, such that v is below u and $\mathrm{rftp}^{\mathfrak{T}}_{\varphi}(u) = \mathrm{rftp}^{\mathfrak{T}}_{\varphi}(v)$,*

then the tree $\mathfrak{T}'$, obtained by replacing the subtree rooted at u by the subtree rooted at v, is also a model of φ.

To show this we observe that the $C(\varphi)$-full type of u in tree $\mathfrak{T}'$ is a combination of the $C(\varphi)$-full types of u and v in $\mathfrak{T}$ and thus, by Lemma 16, it is φ-consistent. Then we show that for every other vertex w in $\mathfrak{T}'$ we have $\mathrm{ftp}^{\mathfrak{T}}_{C(\varphi)}(w) = \mathrm{ftp}^{\mathfrak{T}'}_{C(\varphi)}(w)$. Then Lemma 14 guarantees that the obtained tree $\mathfrak{T}'$ is indeed a model of φ.

▶ **Lemma 20.** *Let φ be a satisfiable formula of $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ in normal form. Then there exists a tree model of φ whose every $\downarrow$-path has length bounded by $3 \cdot (C(\varphi)+2)^{10m+1} \cdot |\boldsymbol{\alpha}_\varphi|^2$.*

Proof. According to Lemma 19 we can restrict attention to models with the property that every φ-reduced full type appears only once on every $\downarrow$-path. Let $\mathfrak{T} \models \varphi$ be a tree model with this property. Let $v_1, v_2, \ldots, v_n$ be a $\downarrow$-path in $\mathfrak{T}$. Observe that the φ-reduced full types on this path behave in a monotonic way in the sense that for every i and the φ-reduced full types of the $i, (i{+}1)$-th vertices $R_i = (\alpha_i, W_i, A_i, B_i, F_i)$ and $R_{i+1} = (\alpha_{i+1}, W_{i+1}, A_{i+1}, B_{i+1}, F_{i+1})$, we have $A_i \subseteq A_{i+1}, B_{i+1} \subseteq B_i$ and $F_i \subseteq F_{i+1}$. A 1-type α can occur in a multiset from 0 to $C(\varphi)$ times. If α appears more than $C(\varphi)$ times, its multiplicity is ∞. Hence the number of modifications of each multiset from A, B, F is bounded by $(C(\varphi)+2) \cdot |\boldsymbol{\alpha}_\varphi|$. There are up to $|\boldsymbol{\alpha}_\varphi| \cdot (C(\varphi)+2)^{10m}$ φ-reduced full types with fixed multisets A, B, F (because it is the number of all possible 1-types multiplied by the number of all possible witness-counting functions). Combination of these two observations gives us the desired estimation $(C(\varphi)+2)^{10m+1} \cdot |\boldsymbol{\alpha}_\varphi|^2 \cdot 3$. ◀

▶ **Lemma 21.** *Let φ be a formula in normal form of $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ satisfied in a finite tree $\mathfrak{T}$. Then there exists a tree model of φ, obtained by removing some subtrees from $\mathfrak{T}$, such that the degree of every vertex is bounded by $(4C(\varphi)^2 + 8C(\varphi)) \cdot |\boldsymbol{\alpha}_\varphi|^5$.*

To prove this lemma, we first limit the degree of a single vertex. Given a vertex v from T, we mark a small number of children of v as important vertices. Marked vertices are then used as required witnesses for v. Then the reasoning is similar to that of Lemma 19. We introduce an appropriate notion of type and remove all nodes on the horizontal path between two children of the same type, provided that the path does not contain any marked vertex. By repeating this procedure as long as there are vertices of high degree we obtain a desired model of φ.

4.4 Algorithm

In this section we design an algorithm checking if a given formula $\varphi \in \mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ has a finite tree model. First, by Lemma 4, we can assume that φ is in normal form. Second, by Theorem 18, we can restrict attention to models with exponentially bounded vertex degree and $\downarrow$-path length. The algorithm works in alternating exponential time. The idea of the algorithm is quite simple (see Procedure 4.2 below). For each vertex v we will guess its $C(\varphi)$-full type and check if it is φ-consistent. If it is, we guess the v's children and their full types. After that, we check if their $C(\varphi)$-full types are locally consistent, i.e., if the guessed types coincide with the types realized in the constructed model (see Procedure 4.1). The algorithm starts with $v =$ root and works recursively with its children. The procedure is an adaptation of the one from [5] used in the context of FO^2 without counting quantifiers.

Let us now sketch the arguments for the correctness of Procedure 4.2.

▶ **Lemma 22.** *Procedure 4.2 accepts its input φ iff φ is satisfiable.*

Procedure 4.1 Checking if given $C(\varphi)$-full types are locally-consistent

Input: $C(\varphi)$-full types $\overline{\alpha}$, $\overline{\alpha_1}$, …, $\overline{\alpha_k}$

1: **Return True** if all of the statements below are true. **Return False** otherwise.
2: $\overline{\alpha_i}(\theta_{\leftarrow}) = \overline{\alpha_{i-1}}(\theta_{=})$ for $i > 1$ and $\overline{\alpha_1}(\theta_{\leftarrow}) = \emptyset$
3: $\overline{\alpha_i}(\theta_{\Leftarrow^+}) = \overline{\alpha_{i-1}}(\theta_{\leftarrow}) \cup \overline{\alpha_{i-1}}(\theta_{\Leftarrow^+})$ for $i > 1$ and $\overline{\alpha_1}(\theta_{\Leftarrow^+}) = \emptyset$
4: $\overline{\alpha_i}(\theta_{\rightarrow}) = \overline{\alpha_{i+1}}(\theta_{=})$ for $i < k$ and $\overline{\alpha_k}(\theta_{\rightarrow}) = \emptyset$
5: $\overline{\alpha_i}(\theta_{\Rightarrow^+}) = \overline{\alpha_{i+1}}(\theta_{\rightarrow}) \cup \overline{\alpha_{i+1}}(\theta_{\Rightarrow^+})$ for $i < k$ and $\overline{\alpha_k}(\theta_{\Rightarrow^+}) = \emptyset$
6: $\overline{\alpha}(\theta_{\downarrow}) = \bigcup_{j=1}^{k} \overline{\alpha_j}(\theta_{=})$
7: $\overline{\alpha}(\theta_{\downarrow\downarrow^+}) = \bigcup_{i=1}^{k} \left(\overline{\alpha_i}(\theta_{\downarrow}) \cup \overline{\alpha_i}(\theta_{\downarrow\downarrow^+})\right)$
8: for $1 \leq i \leq k$: $\overline{\alpha_i}(\theta_{\uparrow}) = \overline{\alpha}(\theta_{=})$
9: for $1 \leq i \leq k$:
$\overline{\alpha_i}(\theta_{\not=}) = \overline{\alpha}(\theta_{\not=}) \cup \overline{\alpha}(\theta_{\leftarrow}) \cup \overline{\alpha}(\theta_{\rightarrow}) \cup \overline{\alpha}(\theta_{\Leftarrow^+}) \cup \overline{\alpha}(\theta_{\Rightarrow^+}) \cup \bigcup_{j \neq i} \left(\overline{\alpha_j}(\theta_{\downarrow}) \cup \overline{\alpha_j}(\theta_{\downarrow\downarrow^+})\right)$

Procedure 4.2 Satisfiability test for $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$

Input: Formula $\varphi \in \mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ in normal form.

1: Let MaxDepth $:= 3 \cdot (C(\varphi) + 2)^{10m+1} \cdot |\boldsymbol{\alpha}_\varphi|^2$
2: Let MaxDeg $:= (4C(\varphi)^2 + 8C(\varphi)) \cdot |\boldsymbol{\alpha}_\varphi|^5$
3: Lvl $:= 0$.
4: **guess** a $C(\varphi)$-full type $\overline{\alpha}$ s.t. $\overline{\alpha}(\theta) = \emptyset$ for all $\theta \in \{\theta_{\uparrow}, \theta_{\uparrow\uparrow^+}, \theta_{\rightarrow}, \theta_{\leftarrow}, \theta_{\Rightarrow^+}, \theta_{\Leftarrow^+}, \theta_{\not=}\}$.
5: **while** Lvl $<$ MaxDepth **do**
6: **if** $\overline{\alpha}$ is not φ-consistent **then reject**
7: **if** $\overline{\alpha}(\theta_{\downarrow}) = \overline{\alpha}(\theta_{\downarrow\downarrow^+}) = \emptyset$ **then accept**
8: **guess** an integer $1 \leq k \leq$ MaxDeg
9: **guess** $C(\varphi)$-full types $\overline{\alpha_1}$, $\overline{\alpha_2}$, …, $\overline{\alpha_k}$
10: **if not** locally-consistent$(\overline{\alpha}, \overline{\alpha_1}, \overline{\alpha_2}, \ldots, \overline{\alpha_k})$ **then reject**
11: Lvl := Lvl + 1
12: **universally choose** $1 \leq i \leq k$; let $\overline{\alpha} = \overline{\alpha_i}$
13: **reject**

Proof. Assume φ is satisfiable. Then there exists a small tree model $\mathfrak{T}$ as guaranteed by Theorem 18. We can run the algorithm and guess exactly the same $C(\varphi)$-full types as in $\mathfrak{T}$. The guessed $C(\varphi)$-full types are locally-consistent and φ-consistent, so Procedure 4.2 accepts.

Assume that Procedure 4.2 accepts its input φ. Then we can reconstruct the tree $\mathfrak{T}$ from the received $C(\varphi)$-full types. The guessed $C(\varphi)$-full types are φ-consistent, which guarantees that we have the right number of witnesses to satisfy the formula. Moreover, the function locally-consistent ensures that the $C(\varphi)$-full types realized in $\mathfrak{T}$ are indeed as we guessed. By Lemma 14, $\mathfrak{T}$ is a tree model for φ and thus φ is satisfiable. ◀

As AExpTime=ExpSpace, and the corresponding lower bound follows from [2] we can conclude this section with the following result.

▶ **Theorem 23.** *The satisfiability problem for* $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ *over finite trees is* ExpSpace*-complete.*

5 Expressive power

A natural question is whether adding counting quantifiers increases the expressive power of two-variable logic over trees. We answer this question concentrating on the classical scenario assuming that signatures contain no common binary symbols. Under this scenario $\mathrm{FO}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+]$ is known to be expressively equivalent to the navigational core of XPath [20]. Here we show that $\mathrm{C}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+]$ shares the same expressivity. However, it is the presence of the sibling relations which makes FO^2 and C^2 equivalent. Indeed, over unordered trees FO^2 cannot count:

▶ **Theorem 24.** $\mathrm{FO}^2[\downarrow,\downarrow^+]$ *is less expressive than* $\mathrm{C}^2[\downarrow,\downarrow^+]$.

Proof. Let us assume that the signature contains no unary predicates and for $i \in \mathbb{N}$ let $\mathfrak{T}_i$ denote the tree consisting just of a root and its i children. Obviously $\mathfrak{T}_3 \models \exists x \exists^{\geq 3} y\ x{\downarrow^+}y$ while $\mathfrak{T}_2 \not\models \exists x \exists^{\geq 3} y\ x{\downarrow^+}y$. On the other hand, $\mathfrak{T}_2$ and $\mathfrak{T}_3$ are indistinguishable in $\mathrm{FO}^2[\downarrow,\downarrow^+]$. This can be seen by observing that Duplicator has a simple winning strategy in the standard two-pebble game of any length played on $\mathfrak{T}_2$ and $\mathfrak{T}_3$. ◀

Now we turn to the case of full navigational signature.

▶ **Theorem 25.** $\mathrm{FO}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+]$ *and* $\mathrm{C}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+]$ *are expressively equivalent.*

The core of the proof is the following technical result, allowing us to translate formulas of a simple shape.

▶ **Lemma 26.** *Let φ be a formula of shape $\exists^{\geq k} y(\theta(x,y) \wedge \psi(y))$, where θ is an order formula, ψ is an $\mathrm{FO}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+,\tau_{com}]$ formula with at most one free variable y. Then there exists an $\mathrm{FO}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+]$ formula $trans(\varphi)$ such that for any tree $\mathfrak{T}$, and any $v \in T$ we have $\mathfrak{T} \models \varphi[v]$ iff $\mathfrak{T} \models trans(\varphi)[v]$.*

This lemma is proved by induction on k. E.g., $trans(\exists^{\geq k} y((y{\downarrow^+}x \wedge \neg y{\downarrow}x) \wedge \psi(y)))$ can be simply defined as $\exists y((y{\downarrow^+}x \wedge \neg y{\downarrow}x) \wedge \psi(y) \wedge trans(\exists^{\geq k-1} x(x{\downarrow^+}y \wedge \psi(x))))$. Note that $x{\downarrow^+}y$ is not an order formula and indeed, to make the induction work we need to formulate the thesis for a wider class of possible specifications of the related position of x and y, including not only the order formulas, but also some simple navigational atoms and, more importantly, even some much more complicated descriptions of the relation between x and y. Intuitively, this allows us always to say where the "first" witness is and how the remaining $k-1$ witnesses are related to it.

Lemma 26 can be then generalized to arbitrary formulas with one free variable. This gives Theorem 25. We skip the details due to the space limit.

6 Combining the two extensions

We have proved that two extensions of two-variable logic on trees: the extension with counting quantifiers, $\mathrm{C}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+]$, and the extension with additional uninterpreted binary relations, $\mathrm{FO}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+,\tau_{com}]$, remain decidable and retain ExpSpace-complexity of $\mathrm{FO}^2[\rightarrow,\rightarrow^+,\downarrow,\downarrow^+]$. It is tempting to combine both variants into a single logic, i.e., to consider $\mathrm{C}^2[\downarrow,\downarrow^+,\rightarrow,\rightarrow^+,\tau_{com}]$, the two-variable logic with counting quantifiers and additional binary relation over trees. However, this turns out to lead to a very difficult formalism. Namely, we can reduce to it the long standing open problem of checking non-emptiness of vector addition tree automata.

▶ **Theorem 27.** *The satisfiability problem for* $\mathrm{C}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ *is at least as hard as checking non-emptiness of vector addition tree automata.*

Proof. To prove the theorem we can mimic the reduction of vector addition tree automata to two-variable logic on *data trees* given in Thm. 4.1 in [4]. Data trees are just trees with an additional, uninterpreted equivalence relation on nodes. In the reduction there the intended equivalence classes are of size at most two. We can easily simulate this by using a common binary symbol $E \in \tau_{com}$, constraining it to be reflexive and symmetric (which is naturally expressible in FO^2), and using counting quantifiers to force each element to be connected by E to at most one other element. The remaining details of the proof remain unchanged. In the proof we do not need to use $\rightarrow$ nor $\rightarrow^+$. ◀

References

1 Bartosz Bednarczyk, Witold Charatonik, and Emanuel Kieronski. Extending two-variable logic on trees. *CoRR*, abs/1611.02112, 2016. URL: `http://arxiv.org/abs/1611.02112`.

2 Saguy Benaim, Michael Benedikt, Witold Charatonik, Emanuel Kieroński, Rastislav Lenhardt, Filip Mazowiecki, and James Worrell. Complexity of two-variable logic on finite trees. *ACM Transactions on Computational Logic*, 17(4):32:1–32:38, 2017. Extended abstract in ICALP 2013.

3 M. Bojanczyk, C. David, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data words. *ACM Transactions on Computational Logic*, 12(4):27, 2011.

4 M. Bojanczyk, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data trees and XML reasoning. *Journal of the ACM*, 56(3), 2009.

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

6 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. `doi:10.1145/2603088.2603134`.

7 Witold Charatonik and Piotr Witkowski. Two-variable logic with counting and a linear order. *Logical Methods in Computer Science*, 12(2), 2016. `doi:10.2168/LMCS-12(2:8)2016`.

8 Witold Charatonik and Piotr Witkowski. Two-variable logic with counting and trees. *ACM Transactions on Computational Logic*, 17(4):31:1–31:27, 2017. Extended abstract in LICS 2013.

9 Philippe de Groote, Bruno Guillaume, and Sylvain Salvati. Vector addition tree automata. In *19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings*, pages 64–73. IEEE Computer Society, 2004. `doi:10.1109/LICS.2004.1319601`.

10 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. `doi:10.1006/inco.2001.2953`.

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

12 E. Grädel, P. Kolaitis, and M. Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3(1):53–69, 1997.

13 E. Grädel, M. Otto, and E. Rosen. Two-variable logic with counting is decidable. In *LICS 1997, Proceedings*, pages 306–317, 1997.

14 Emanuel Kieroński. Results on the guarded fragment with equivalence or transitive relations. In C.-H. Luke Ong, editor, *CSL*, volume 3634 of *Lecture Notes in Computer Science*, pages 309–324. Springer, 2005. doi:10.1007/11538363_22.

15 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. doi:10.4230/LIPIcs.CSL.2011.337.

16 Emanuel Kieroński, Jakub Michaliszyn, Ian Pratt-Hartmann, and Lidia Tendera. Two-variable first-order logic with equivalence closure. In *LICS*, pages 431–440. IEEE Computer Society, 2012. doi:10.1109/LICS.2012.53.

17 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. doi:10.1109/LICS.2005.49.

18 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. doi:10.1109/LICS.2009.39.

19 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. doi:10.1007/978-3-642-15155-2_45.

20 Maarten Marx and Maarten de Rijke. Semantic characterization of navigational XPath. In *First Twente Data Management Workshop (TDM 2004) on XML Databases and Information Retrieval*, pages 73–79, 2004.

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

22 Leszek Pacholski, Wieslaw Szwast, and Lidia Tendera. Complexity of two-variable logic with counting. In *LICS*, pages 318–327, 1997. doi:10.1109/LICS.1997.614958.

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

24 I. Pratt-Hartmann. Logics with counting and equivalence. In *CSL-LICS 2014, Proceedings*, page 76, 2014.

25 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. doi:10.1007/978-3-642-15205-4_38.

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

27 Thomas Zeume and Frederik Harwath. Order-invariance of two-variable logic is decidable. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS'16, pages 807–816, New York, NY, USA, 2016. ACM. doi:10.1145/2933575.2933594.

A Algorithm for two-variable logic over trees

In this section we give a more detailed description of the algorithm solving the satisfiability problem for $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$. Then we explain that it has the desired complexity and sketch the argument for its correctness.

The algorithm employs a data structure, storing for each node v the following three components:

- v.`1tp` – the 1-type of v,
- v.`2tp()` – the function which for each w being a sibling of v, an ancestor of v or a member of F, returns the 2-type of (v, w),

Procedure A.1 $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$-sat-test

Input: a formula φ in $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$ normal form

guess a tree $\mathfrak{F}$ of depth and degree of nodes bounded by $\mathfrak{f}(\varphi)$ and the number of nodes bounded by $3(m{+}1)^3(\mathfrak{f}(\varphi))^4|\boldsymbol{\alpha}_\varphi|$
for each $v \in F$ **do**
  if v is not a leaf in $\mathfrak{F}$ **then**
    if not *consistent-with-ancestors-siblings-F*(v) **then reject**
    if not *has-upper-sibling-free-witnesses*(v) **then reject**
    Let $w_1, \ldots, w_k$ be the list of the children of v
    if not *ensure-lower-witnesses*$(v, w_1, \ldots, w_k)$ **then reject**
    if not *propagates-promised-2-types*$(v, w_1, \ldots, w_k)$ **then reject**
    if not *respects-universal-conjunct*$(v, w_1, \ldots, w_k)$ **then reject**
universally choose a leaf v of $\mathfrak{F}$; let l be the depth of v in $\mathfrak{F}$
while $l \leq \mathfrak{f}(\varphi)$ **do**
  if not *consistent-with-ancestors-siblings-F*(v) **then reject**
  if not *has-upper-sibling-free-witnesses*(v) **then reject**
  guess a list $w_1, \ldots, w_k$ of children of v; **if** $k > \mathfrak{f}(\varphi)$ **then reject**
  if not *ensure-lower-witnesses*$(v, w_1, \ldots, w_k)$ **then reject**
  if not *propagates-promised-2-types*$(v, w_1, \ldots, w_k)$ **then reject**
  if not *respects-universal-conjunct*$(v, w_1, \ldots, w_k)$ **then reject**
  if $k = 0$ **then accept** *% v is a leaf*
  universally choose $1 \leq j \leq k$ and set $v := w_j$
reject

- v.`p2tp()` – a function which for each ancestor w of v returns a list of promised 2-types, intended to contain all the 2-types which will be realized by w with descendants of v.

We assume that if a node v is guessed during the execution of our procedure then all the above components are constructed.

To avoid presentational clutter in the description of our algorithm we omit some natural conditions on 2-types guessed during its execution, always assuming that they contain the intended navigational atoms, i.e., the 2-type joining an element with its child contains $x{\downarrow}y$, with its right sibling $x{\rightarrow}y$, and so on.

The algorithm is presented as a main Procedure A.1 employing five auxiliary Functions (A.2, A.3, A.4, A.5, A.6).

Function A.2 checks if all guessed components of v are consistent with the information about v's siblings, its ancestors and all the elements of the substructure $\mathfrak{F}$ of global free witnesses.

The next function A.3 checks if v has the required upper, sibling and free witnesses.

Function A.4 checks if the guess of the v's children $w_1, \ldots, w_k$ guarantees lower witnesses for v.

Function A.5 checks if the guess of v.`p2tp()` is propagated to the children of v and consistent with w_i.`p2tp()`.

The last function A.6 checks if the 2-types formed by v with all elements of the constructed model (existing or promised) respect the $\forall\forall$ conjunct.

Using the introduced functions we build our main procedure $\mathrm{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$-`sat-test` (Procedure A.1). We now sketch the arguments showing that it has the advertised complexity and returns correct results.

Function A.2 *consistent-with-ancestors-siblings-F(v)*

for each w being a sibling of v **do**
  let $\beta = v.\mathtt{2tp}(w)$
  if $w.\mathtt{2tp}(v) \neq \beta^{-1}$ **then return false**
if $v \in F$ **then**
  for each $w \in F$ **do**
    let $\beta = v.\mathtt{2tp}(w)$
    if $w.\mathtt{2tp}(v) \neq \beta^{-1}$ **then return false**
if v is the root **then return true**
let u be the father of v
for each w being an ancestor of u **do**
  if $w.\mathtt{2tp}(v) \notin u.\mathtt{p2tp}(w)$ **then return false**
return true

Function A.3 *has-upper-sibling-free-witnesses(v)*

for each conjunct $\forall x(\lambda_i(x) \to \exists y(\theta_i(x,y) \wedge \chi_i(x,y)))$ of φ
with $\theta_i \in \{\theta_\downarrow, \theta_{\downarrow\downarrow^+}, \theta_\rightarrow, \theta_\leftarrow, \theta_{\rightarrow^+}, \theta_{\leftarrow^+}, \theta_{\not\sim}\}$ **do**
  if $v.\mathtt{1tp} \models \lambda_i(x)$ and there is no element w being an ancestor or a sibling of v or a member of F such that $v.\mathrm{2tp}(w) \models \theta_i(x,y) \wedge \chi_i(x,y)$ **then return** false
return true

Function A.4 *ensure-lower-witnesses*$(v, w_1, \ldots, w_k)$

for each conjunct $\forall x(\lambda_i(x) \to \exists y\, \theta_\downarrow(x,y) \wedge \chi_i(x,y))$ of φ **do**
  if $v.\mathtt{1tp} \models \lambda_i(x)$ and there is no w_i such that $v.\mathtt{2tp}(w_i) \models \chi_i(x,y)$ **then**
    return false
for each conjunct $\forall x(\lambda_i(x) \to \exists y\, \theta_{\downarrow\downarrow^+}(x,y) \wedge \chi_i(x,y))$ of φ **do**
  if $v.\mathtt{1tp} \models \lambda_i(x)$ and there is no w_i such that for some $\beta \in w_i.\mathtt{p2tp}(v) : \beta \models \chi_i(x,y)$ **then return false**
return true

Function A.5 *propagates-promised-2-types*$(v, w_1, \ldots, w_k)$

for each u being an ancestor of v **do**
  if $v.\mathtt{p2tp}(u) \neq \bigcup_{i=1}^{k} \left(\{w_i.\mathtt{2tp}(u)^{-1}\} \cup w_i.\mathtt{p2tp}(u)\right)$ **then return false**
return true

▶ **Lemma 28.** *The procedure* FO$^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+, \tau_{com}]$-`sat-test` *works in alternating exponential time.*

Proof. During its execution the algorithm guesses $\mathfrak{F}$, and builds a single path P in $\mathfrak{T}$ together with the siblings of the elements from P. The size of $\mathfrak{F}$ is bounded by $3(m+1)^3(\mathfrak{f}(\varphi))^4|\boldsymbol{\alpha}_\varphi|$, the length of P and the degree of nodes are bounded by $\mathfrak{f}(\varphi)$, where m is linear in $|\varphi|$ and $\mathfrak{f}(\varphi)$ and $|\boldsymbol{\alpha}_\varphi|$ are exponential in $|\varphi|$. Thus the algorithm constructs exponentially many nodes. For each node it guesses its 1-type, 2-types joining it with its siblings, ancestors and the elements of $\mathfrak{F}$ (exponentially many in total) and promised 2-types for each of its ancestors (again, information about the 2-types for a single ancestor is bounded exponentially, since the total number of possible 2-types is so bounded). The algorithm makes some consistency

Function A.6 *respects-universal-conjunct*$(v, w_1, \ldots, w_k)$

1: **for each** u being an ancestor of v, a sibling of v, a member of F **do**
2: **if** $v.\mathtt{2tp}(u) \not\models \chi(x,y)$ **then return false**
3: **if** $(v.\mathtt{2tp}(u))^{-1} \not\models \chi(x,y)$ **then return false**
4: **if** v is the root **then return true else** let u be the father of v
5: **for each** w_i **do**
6: let $desc_{w_i} := \{\alpha : \exists \beta \in w_i.\mathtt{p2tp}(u) \wedge \alpha = \beta{\restriction}y\}$.
% $desc_{w_i}$ *is the list of promised* 1*-types of descendants of* w_i
7: **for each** $i \neq j$ **do**
8: **for each** 1-type α from $desc_{w_i}$ **do**
9: **for each** 1-type $\alpha' \in desc_{w_j} \cup \{w_j.\mathtt{1tp}\}$ **do**
10: **if** there is no 2-type β such that $\beta{\restriction}x = \alpha'$ and $\beta{\restriction}y = \alpha$ and $\beta(x,y) \models \theta_{\mathcal{A}}(x,y) \wedge \chi(x,y)$ **then return false**
11: **return** true

and correctness checking, which can be easily done in time polynomial in the size of the guesses. Hence the lemma follows. ◀

▶ **Lemma 29.** *The procedure* $\mathrm{FO}^2[{\downarrow}, {\downarrow^+}, {\rightarrow}, {\rightarrow^+}, \tau_{com}]$-`sat-test` *accepts its input* φ *iff* φ *is satisfiable.*

Proof. (Sketch.) Assume φ has a model. By Theorem 6 it has a model $\mathfrak{T}$ whose depth and degree of nodes are bounded by $\mathfrak{f}(\varphi)$. By Lemma 9 there is a model $\mathfrak{T}'$ based on the same frame as $\mathfrak{T}$, in which one can distinguish a set F, of size at most $3(m+1)^3(\mathfrak{f}(\varphi))^4|\boldsymbol{\alpha}_\varphi|$, providing free witnesses for all elements of $\mathfrak{T}'$. Our algorithm can just take $\mathfrak{F} := \mathfrak{T}'{\restriction}F$ and make all its guesses in accordance with $\mathfrak{T}$.

For the opposite direction assume that our algorithm has an accepting run. From this run we can naturally extract a partially defined tree structure $\mathfrak{T}$ and its substructure $\mathfrak{F}$. $\mathfrak{T}$ has defined its tree frame, 1-types of all nodes ($v.\mathtt{1tp}$ components), 2-types of nodes not in free position and 2-types of nodes in free position at least one of which is in F: the 2-type joining v and w is stored in $v.\mathtt{2tp}(w)$ if v is a descendant of w, or if $w \in F$ and $v \notin F$, and in both $v.\mathtt{2tp}(w)$ and $w.\mathtt{2tp}(v)$ if v and w are siblings or $v, w \in F$. Note that the function *consistent-with-ancestors-siblings-F* ensures that the 2-types can be assigned without conflicts. This function, together with function *propagates-promised-2-types* ensures also the consistency of the information about promised 2-types.

What is missing is 2-types of pairs of elements u_1, u_2 in free position none of which is in F. In this case there is an element v such that u_1, u_2 are descendants of two different children of v from the list $w_1, \ldots, w_k$. Then, due to lines 7-10 of the function *respects-universal-conjunct*, there exists a 2-type consistent with the $\forall\forall$-conjunct which can join them.

The constructed tree $\mathfrak{T}$ is indeed a model of φ: *respects-universal-conjunct* takes care of $\forall\forall$ constraint of φ, the sibling, upper and free witnesses are ensured due to function *has-upper-sibling-free-witnesses* and lower witnesses are guaranteed by function *ensure-lower-witnesses* which uses the information about promised-2-types. ◀