One-Dimensional Fragment Over Words and Trees

Kieronski, Emanuel

2022-07-25

Kieronski , E & Kuusisto , A 2022 , ' One-Dimensional Fragment Over Words and Trees ' , Journal of Logic and Computation , vol. 32 , no. 5 , pp. 902-941 . https://doi.org/10.1093/logcom/exac002

http://hdl.handle.net/10138/347450
https://doi.org/10.1093/logcom/exac002

One-dimensional fragment over words and trees

EMANUEL KIEROŃSKI, *Faculty of Mathematics and Computer Science, University of Wrocław, ul. Joliot-Curie 15, 50-383 Wrocław, Poland.*
E-mail: emanuel.kieronski@cs.uni.wroc.pl

ANTTI KUUSISTO, *Faculty of Information Technology and Communication Sciences, Tampere University, Kalevantie 4, 33100 Tampere, Finland and Mathematics and Statistics, University of Helsinki, Pietari Kalmin katu 5, 00560 Helsinki, Finland.*
E-mail: antti.kuusisto@tuni.fi

Abstract

One-dimensional fragment of first-order logic is obtained by restricting quantification to blocks of existential (universal) quantifiers that leave at most one variable free. We investigate this fragment over words and trees, presenting a complete classification of the complexity of its satisfiability problem for various navigational signatures and comparing its expressive power with other important formalisms. These include the two-variable fragment with counting and the unary negation fragment.

1 Introduction

One-dimensional fragment of first-order logic (F_1) is obtained by restricting quantification to blocks of existential quantifiers that leave at most one variable free. As the logic is closed under negation, one may also use blocks of universal quantifiers. F_1 contains a few known decidable fragments of first-order logic: the prenex form class $\forall\exists^*$ with equality, the two-variable fragment FO^2 and (the so-called UN-normal form of formulas in) the unary negation fragment UNFO.

Unfortunately, over general relational structures, the satisfiability problem for F_1 is undecidable [13]. In such situation, one may attempt to regain the decidability in two principal ways: by imposing some additional restrictions on the syntax of the considered logic or by restricting attention to some specific classes of structures.

Regarding the first idea, a nice syntactic restriction of F_1, which turns out to be decidable over the class of all relational structures, is called the *uniform* one-dimensional fragment UF_1. It was introduced by Hella and Kuusisto in [13] as a generalization of the two-variable fragment of first-order logic to contexts with relations of all arities—in particular, relations with arities greater than two. Such contexts naturally include, e.g. databases. The readers interested in this variant are referred to [13], [17] and [18] and the survey [21], the latter also revealing some connections to description logics.

In this paper we will investigate F_1 over restricted classes of structures. There are two important options, well motivated in various areas of computer science, namely the class of words and the class of trees. Our aim is to investigate the complexity of the satisfiability problem of F_1 over words and trees and to compare the expressive power of F_1 over these classes of structures with a few

Vol. 32, No. 5, © The Author(s) 2022. Published by Oxford University Press.

Advance Access Publication on 19 February 2022 https://doi.org/10.1093/logcom/exac002

other formalisms considered in this context. To set up the scene, let us recall the main results on satisfiability and relative expressivity of fragments of first-order logic over words and trees.

Over words, it is known that the satisfiability problem for full first-order logic (FO) is decidable, but with non-elementary complexity. In fact, as shown by Stockmeyer [28], already the fragment with three variables (FO^3) is equally expressive to full FO and has non-elementary complexity. On the other hand, for linear temporal logic (LTL), which is in fact a fragment of FO^3, the satisfiability problem is only PSPACE-complete [25], even though its expressive power is still equal to that of FO [11, 14]. A reasonable complexity is also obtained when the number of variables is restricted to two. The satisfiability problem for FO^2 over words and ω-words was shown to be NEXPTIME-complete by Etessami et al. [10]. This time, however, the expressive power is weaker: In [10] it was observed that the expressive power of FO^2 over words is equal to the expressive power of *unary* temporal logic, UTL, i.e. LTL with the four navigational operators *next state*, *somewhere in the future*, *previous state*, *somewhere in the past*, but without the operators *since* and *until*. FO^2 turns out to be exponentially more succinct than UTL. The extension of FO^2 by counting quantifiers, C^2, was shown to be NEXPTIME-complete over words by Charatonik and Witkowski [8]. In fact, it is not difficult to observe that over words, C^2 has the same expressive power as plain FO^2. Another interesting extension of FO^2, this time significantly increasing its expressive power (but not to the power of full FO), is the extension by the *between* predicate recently studied by Krebs et al. [20]. Satisfiability for this logic is EXPSPACE-complete.

Turning then to the class of trees, both FO^2 and C^2 retain a reasonable complexity, namely their satisfiability problems over trees are EXPSPACE-complete. See Benaim et al. [2] for the analysis of FO^2 over trees and Bednarczyk et al. [3] for an extension covering C^2. Regarding the expressive power, the situation depends on the type of trees considered. In the case of *unordered* trees, FO^2 cannot count and is thus less expressive than C^2. Over *ordered* trees, both formalisms are equally expressive [3] and share the expressiveness with the navigational core of XPath, CoreXPath (cf. Marx and de Rijke [23]), which is a logic similar in spirit to UTL used to reason about XML trees.

Our results over words. We first analyse the expressive power and the complexity of the satisfiability problem of F_1 over words and ω-words. In our scenario we assume that at each position of a word (ω-word), multiple unary predicates may be true, and two navigational binary predicates are used to navigate structures: successor $\rightarrow$ and its transitive closure $\rightarrow^+$.

It is rather clear that we do not have the full power of FO. For example, in the standard translation of the LTL formula $P\ until Q$ to FO:

$$Q(x) \vee (P(x) \wedge \exists y(x \rightarrow^+ y \wedge Q(y) \wedge \forall z(x \rightarrow^+ z \wedge z \rightarrow^+ y \Rightarrow P(z)))),$$

the universal quantifier leaves two variables x and y free, and indeed there is no way to avoid this.

We show that the expressive power of F_1 over the considered classes of structures is the same as the expressive power of FO^2, and thus also of UTL and C^2.

The advantage of F_1 over these other formalisms is that it allows to specify many properties in a more natural and elegant way. If we want to say that a word contains some (especially not fully specified) pattern, consisting of more than two elements, we can just quantify the appropriate number of positions and say how they should be labelled and related to each other. Expressing the same in FO^2 will usually require some heavy recycling of the two available variables. Let us look at two simple examples. Consider a system whose behaviour we model as a word, or an ω-word, in which one or more of the atomic propositions $P_1, \ldots, P_n$ can hold in a given point of time. To say that there are m non-overlapping time intervals (sets of consecutive positions of the word) in each of which

each P_i holds at least once, we can use the following $F_1[\rightarrow^+]$ sentence:

$$\exists y_0 y_1 \dots y_m x_{11} \dots x_{1n} \dots x_{m1} \dots x_{mn} \Big(\bigwedge_{i=1}^{m} \bigwedge_{j=1}^{n} y_{i-1} \rightarrow^+ x_{ij} \wedge x_{ij} \rightarrow^+ y_i \wedge P_j x_{ij}\Big). \tag{1.1}$$

As another example[1] , not using the navigational predicates at all, consider the property saying that it is possible to choose m positions satisfying together all of the P_i:

$$\exists x_1 \dots x_m \Big(\bigwedge_{i=1}^{n} \bigvee_{j=1}^{m} P_i x_j\Big). \tag{1.2}$$

The reader can check that expressing the above properties in $FO^2[\rightarrow, \rightarrow^+]$ is indeed not straightforward and leads to complicated formulas.

In fact, our translation of $F_1[\rightarrow, \rightarrow^+]$ to $FO^2[\rightarrow, \rightarrow^+]$ has an exponential blow-up, which seems to be hard to avoid, and which thus suggests that $F_1[\rightarrow, \rightarrow^+]$ may be able to express some properties more succinctly than $FO^2[\rightarrow, \rightarrow^+]$, and possibly even more succinctly than $C^2[\rightarrow, \rightarrow^+]$.

Regarding the complexity, we show that satisfiability of F_1 over words and ω-words is NEXPTIME-complete, i.e. it is of the same complexity as satisfiability of FO^2 and C^2. While our proof has some similarities to the proof of Etessami et al. [10] for $FO^2[\rightarrow, \rightarrow^+]$, it is technically more difficult, due to the combinatorically more complicated nature of the objects involved. Not surprisingly, the basic idea in the proof is based on an appropriately tuned contraction procedure.

We also examine some possible extensions of $F_1[\rightarrow, \rightarrow^+]$. Perhaps the most significant of them is the extension of $F_1[\rightarrow, \rightarrow^+]$ by an equivalence relation, inspired by an analogous extension of $FO^2[\rightarrow, \rightarrow^+]$ (FO^2 over *data words*), studied by Bojańczyk et al. [4]. The satisfiability problem for FO^2 over data words, even though very hard, is decidable. We show that $F_1[\rightarrow, \rightarrow^+]$ over data words becomes undecidable.

Our results over trees. We consider finite unranked trees accessible by navigational signatures built out of (some of) the following relations: child $\downarrow$, descendant $\downarrow_+$, next sibling $\rightarrow$ and following sibling $\rightarrow^+$. Concerning the complexity of satisfiability, it turns out that it depends on whether $\downarrow$ is present or not. With $\downarrow$ the satisfiability problem is 2-EXPTIME-complete, and without $\downarrow$ it is EXPSPACE-complete. To show the upper complexity bound in the case of the full navigational signature, we will use the existing results for UNFO by Segoufin and ten Cate [27]. For the EXPSPACE bound we perform some surgery on models leading to small model properties, and then design an algorithm searching for such appropriate small models. Technically, we extend the approach from [6] used there in the context of FO^2. Roughly speaking, we appropriately abstract the information about a node by its *profile* (an analogous notion is called *a full type* in [6]) and then we contract trees, removing their fragments between nodes with the same profiles. We explain also how to use these techniques to directly reprove the upper bound for the full signature. The lower bounds are inherited from other formalisms.

It is worth mentioning that an orthogonal extension of the method from [6] is used in [3] in the context of C^2. In both cases the challenge is to carefully tune the notion of a profile (full type) in order to get the optimal complexity.

Regarding expressivity, we show that over ordered trees with all of the four navigational relations we consider, F_1 is expressively equivalent to each of CoreXPath, GF^2, FO^2, C^2, UNFO. We also show that over unordered trees equipped with both the descendant and the child relation, F_1 is still equivalent to C^2, but we establish that this time FO^2 is less expressive and that CoreXPath, GF^2 and

[1] Suggested to the authors by Jakub Michaliszyn.

UNFO are less expressive than FO^2 (and equiexpressive with each other). Most of these expressivity results are rather easy to obtain (though in some cases, slightly awkward to formally show). The exception is the equivalence of F_1 and C^2 in the absence of the sibling relations, which is less obvious and more difficult to prove. In our expressivity-related studies, we do not consider the cases of unordered trees accessible by only one of the descendant and the child relations.

Organization of the paper. The rest of the paper is organized as follows. In Section 2 we define the logics and structures we are interested in and introduce some basic notions and results that will then be used in the following sections. In Section 3 we compare the expressivity of F_1 with other formalisms over words and ω-words, and in Section 4 we analyse the complexity of F_1 over words and ω-words. Section 5 concerns the expressive power of F_1 over trees, and in Section 6 we analyse the complexity of F_1 over trees. Finally, in Section 7, we conclude the paper.

2 Preliminaries

2.1 Structures

We employ conventional terminology and notation from model theory throughout this article, assuming the reader is familiar with most of the standard concepts. We refer to structures using Gothic capital letters (e.g. $\mathfrak{M}$) and their domains using the corresponding Roman capitals (e.g. M).

We are interested in signatures of the form $\sigma = \sigma_0 \cup \sigma_{nav}$, where σ_0 consists of some number of unary relation symbols, and σ_{nav}, called the *navigational signature*, is a subset of $\{\rightarrow, \rightarrow^+, \downarrow, \downarrow_+\}$.

A *word* is a finite structure over a signature $\sigma_0 \cup \{\rightarrow, \rightarrow^+\}$ in which $\rightarrow^+$ is a (strict) linear order and $\rightarrow$ is its induced successor relation. An infinite structure over the same signature and containing a reduct isomorphic to $(\mathbb{N}, +1, <)$ is called an *ω-word*. Given a word $\mathfrak{M}$, its element a and a number $i \in \mathbb{N}$, we will sometimes refer by $a + i$ (respectively, $a - i$) to the element located i positions to the right (resp., left) from a. We will also use the notation $\mathfrak{M} = \mathfrak{M}_1 a$ to denote that the word $\mathfrak{M}$ is the concatenation of the word $\mathfrak{M}_1$ with the element a. In the similar vein we will write $\mathfrak{M} = \mathfrak{M}_1 a \mathfrak{M}_2$, etc.

Let $\mathbb{N}^*$ denote the set of finite sequences of natural numbers, containing in particular the empty sequence ϵ. For $\alpha, \beta \in \mathbb{N}^*$ and $i \in \mathbb{N}$, we denote by $\langle \alpha, i \rangle$ the sequence obtained as the result of appending i to α and by $\langle \alpha, \beta \rangle$ the result of concatenating α and β. A *tree* is a finite structure $\mathfrak{T}$ whose universe T is a subset of $\mathbb{N}^*$ such that if $\langle \alpha, i \rangle \in T$, then $\alpha \in T$, and in the case $i > 0$, also $\langle \alpha, i-1 \rangle \in T$. In a tree, at least one of $\downarrow$, $\downarrow_+$ and possibly one or both of $\rightarrow$, $\rightarrow^+$ are interpreted, each of them in the following fixed way. For $a, b \in T$, we have $\mathfrak{T} \models a \downarrow b$ iff $a = \alpha$ and $b = \langle \alpha, i \rangle$ for some $\alpha \in \mathbb{N}^*$ and $i \in \mathbb{N}$; $\mathfrak{T} \models a \downarrow_+ b$ iff $a = \alpha$ and $b = \langle \alpha, \beta \rangle$ for some $\alpha, \beta \in \mathbb{N}^*$, $\beta \neq \epsilon$; $\mathfrak{T} \models a \rightarrow b$ iff $a = \langle \alpha, i \rangle$ and $b = \langle \alpha, i+1 \rangle$ for some $\alpha \in \mathbb{N}^*$; and $i \in \mathbb{N}$; $\mathfrak{T} \models a \rightarrow^+ b$ iff $a = \langle \alpha, i \rangle$ and $b = \langle \alpha, j \rangle$ for some $\alpha \in \mathbb{N}^*$ and $i, j \in \mathbb{N}$, $i < j$.

When speaking about trees we use the natural terminology. The elements of T are sometimes called *nodes*. The element ϵ is called the *root* of $\mathfrak{T}$, nodes $\alpha \in T$ for which there is no $i \in \mathbb{N}$ such that $\langle \alpha, i \rangle \in T$, are called *leaves*. For a node α, the nodes $\langle \alpha, i \rangle$ are called its *children*, the node $\langle \alpha, 0 \rangle$ is its *leftmost child*, the node $\langle \alpha, i \rangle$ for which $\langle \alpha, i+1 \rangle \notin T$ is its *rightmost child*, the node β such that $\alpha = \langle \beta, i \rangle$ is its *parent*, the nodes $\langle \alpha, \beta \rangle$ where $\beta \neq \epsilon$ are its *descendants*, the nodes β such that β is a proper prefix of α are its *ancestors*, the node $\langle \alpha, i-1 \rangle$ (if $i > 0$) is its *previous sibling*, the node $\langle \alpha, i+1 \rangle$ (if it belongs to T) is its *next sibling*, the nodes $\langle \alpha, j \rangle$ for $j < i$ are its *preceding siblings* and the nodes $\langle \alpha, j \rangle$ for $j > i$ are its *following siblings*.

The relations $\downarrow$, $\downarrow_+$, $\rightarrow$, $\rightarrow^+$ are called, the *child*-, *descendant*-, *next sibling*- and *following sibling* relations, respectively. If a tree interprets at least one of $\rightarrow$, $\rightarrow^+$, then it is called an *ordered tree*; otherwise, it is an *unordered tree*. Trees interpreting all four navigational relations are called XML

trees. Trees in this paper are *unranked*, i.e., there is no *a priori* bound on the number of the children of a node.

We say that a chain of nodes $\epsilon, \langle i_1\rangle, \langle i_1, i_2\rangle, \ldots, \langle i_1, i_2, \ldots, i_l\rangle$, where the last element is a leaf, is a *vertical path*, and a chain of elements $\langle\alpha, 0\rangle, \langle\alpha, 1\rangle, \ldots, \langle\alpha, l\rangle$, where the last element is a rightmost child, is a *horizontal path*. We may speak about vertical (horizontal) paths even if the structure does not interpret $\downarrow$ ($\rightarrow$).

2.2 Logics

Over such structures we consider the *one-dimensional fragment*, F_1, and compare it with several other fragments of first-order logic. F_1 is the relational fragment in which quantification is restricted to blocks of existential quantifiers that leave at most one variable free. Formally, the set of formulas of F_1 over the relational signature σ and some countably infinite set of variables *Var* is the smallest set such that

- $R\bar{x} \in \mathrm{F}_1$ for all $R \in \sigma$ and all tuples $\bar{x}$ of variables from *Var* of the appropriate length,
- $x = y \in \mathrm{F}_1$ for all variables $x, y \in Var$,
- F_1 is closed under $\vee$ and $\neg$,
- if φ is an F_1 formula with the free variables $x_0, \ldots, x_k$, then the formulas $\exists x_0, \ldots, x_k\varphi$ and $\exists x_1, \ldots, x_k\varphi$ belong to F_1.

As usual, we can use standard abbreviations for other Boolean operations, like $\wedge, \Rightarrow, \top$, etc., as well as for universal quantification. The length of a formula φ is measured as the total number of symbols required to write down φ, and denoted $\|\varphi\|$. The *width* of a formula is the maximum of the numbers of free variables in its subformulas.

We will write $\mathrm{F}_1[\sigma_{nav}]$ to indicated that we are interested in F_1 formulas over the signature $\sigma_0 \cup \sigma_{nav}$ for some set σ_0 of unary relation symbols. We will use the same convention for other logics also.

Some results in this paper will refer to the *unary negation fragment*, UNFO [27]. The set of UNFO formulas is the smallest set of formulas such that

- $R\bar{x} \in$ UNFO for all $R \in \sigma$ and all tuples $\bar{x}$ of variables from *Var* of the appropriate length,
- $x = y \in$ UNFO for all variables $x, y \in Var$,
- UNFO is closed under $\vee$, $\wedge$ and existential quantification,
- if $\varphi(x)$ is an UNFO formula with no free variables besides (at most) x then $\neg\varphi(x)$ is also in UNFO.

We emphasize that UNFO is not closed under negation and does not allow for a direct universal quantification.

The following lemma, showing that UNFO may be seen as a fragment of F_1, is implicit in [27]:

LEMMA 1
There is a polynomial procedure that, given an UNFO formula φ, produces an equivalent formula φ' in UNFO $\cap$ F_1 over the same signature.

PROOF. In [27], it is shown that any UNFO formula can be converted into the so-called UN-normal form, which is one-dimensional by definition. □

We note that, generally, no translation from F_1 to UNFO exists. This non-existence is shown in [13] for the extension GNFO of UNFO. Actually, the satisfiability problem (over the class of all structures) for UNFO is decidable [27], and for F_1 it is undecidable [13].

Other relevant fragments of first-order logic, which will be mentioned in this paper, are the following: the *two-variable* fragment, FO^2; the *two-variable fragment with counting quantifiers*, C^2; the two-variable version of the *guarded fragment*, GF^2; the navigational core of XPath, CoreXPath; and the *unary temporal logic*, UTL.

The formulas of FO^2 are just those first-order relational formulas that use only the two variables x and y. GF^2 is the fragment of FO^2 in which every quantifier is appropriately relativized by an atomic formula (see, e.g., [12]). C^2 extends FO^2 by counting quantifiers, i.e. it adds to FO^2 constructs of the form $\exists^{\geq C} y\psi(x,y)$ and $\exists^{\leq C} y\psi(x,y)$, for $C \in \mathbb{N}$, with the natural semantics: for $a \in A$, we have that $\mathfrak{A} \models \exists^{\geq C} y\psi(a,y)$ if there are at least C elements $b \in A$ such that $\mathfrak{A} \models \psi(a,b)$. Analogously for $\exists^{\leq C}$.

UTL will be mentioned in the case of words. It is a temporal logic with four navigational operators: *next state*, *somewhere in the future*, *previous state* and *somewhere in the past*, but without binary operators *since* and *until* (see [10] for more details).

A corresponding formalism for trees is CoreXPath. We present it here as a modal logic with four pairs of modalities, each pair corresponding to one of the relations from the set $\{\downarrow, \downarrow_+, \rightarrow, \rightarrow^+\}$. Definitions in the literature slightly differ from ours, but the spirit is the same. Let Σ_0 be a set of propositional variables, and let us consider the following eight *modalities*: $\langle\downarrow\rangle$, $\langle\uparrow\rangle$, $\langle\downarrow_+\rangle$, $\langle\uparrow^+\rangle$, $\langle\rightarrow\rangle$, $\langle\leftarrow\rangle$, $\langle\rightarrow^+\rangle$ and $\langle\leftarrow^+\rangle$. The set of CoreXPath formulas over Σ_0 is the least set such that

- any P in Σ_0 is in CoreXPath,
- CoreXPath is closed under Boolean connectives
- if ψ is in CoreXPath then so is $\langle\cdot\rangle\psi$ for any modality $\langle\cdot\rangle$.

Identifying Σ_0 with σ_0 (i.e. treating propositional variables of Σ_0 as unary relation symbols in σ_0), we can interpret CoreXPath formulas over trees. Given a tree $\mathfrak{T}$ and its node a we inductively define what it means that a CoreXPath formula ψ holds at a, written $\mathfrak{T}, a \models \psi$. For $P \in \Sigma_0$ we have $\mathfrak{T}, a \models P$ iff $\mathfrak{T} \models P(a)$, $\mathfrak{T}, a \models \langle\downarrow\rangle\psi'$ if there is $b \in T$ such that $\mathfrak{T} \models a\downarrow b$ and $\mathfrak{T}, b \models \psi'$, and analogously for the other modalities, which require ψ' to be satisfied at, respectively, the parent, a descendant, an ancestor, the next sibling, the previous sibling, a following sibling and a preceding sibling.

Using the so-called *standard translation* we can translate CoreXPath formulas to equivalent first-order formulas with one free variable. By an appropriate reuse of variables this translation fits into $\mathrm{FO}^2[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$, and actually even in $\mathrm{GF}^2[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$ (cf. [23]). As an example, the formula $\langle\uparrow\rangle(P \wedge \langle\rightarrow^+\rangle(Q \vee \langle\downarrow_+\rangle R))$ can be translated to $\exists y(y\downarrow x \wedge P(y) \wedge \exists x(y \rightarrow^+ x \wedge (Q(x) \vee \exists y(x\downarrow_+ y \wedge R(y)))))$.

We remark that a similar translation exists for UTL [10].

2.3 Comparing expressive powers

In this paper we will compare the expressive powers of the logics mentioned in the previous paragraph over words and trees. We will concentrate on the case of formulas with one free variable. This is a natural choice when taking into account the character of the logics considered: e.g. (the standard translations of) formulas in CoreXPath and UTL always have exactly one free variable and quantified subformulas in F_1, GF^2, FO^2 and C^2 leave at most one variable free.

Let $\mathcal{C}$ be a class of structures. We say that a logic L_1 is *less or equally expressive* than a logic L_2 over $\mathcal{C}$, written $\mathsf{L}_1 \preceq \mathsf{L}_2$ ($\mathcal{C}$ will always be clear from the context) if for any formula with one free variable $\varphi_1(x)$ in L_1, there is a formula with one free variable $\varphi_2(x)$ in L_2 over the same alphabet such that for any structure $\mathfrak{A}$ and $a \in A$, we have $\mathfrak{A} \models \varphi_1(a)$ iff $\mathfrak{A} \models \varphi_2(a)$.

If $L_1 \preceq L_2$ and $L_2 \preceq L_1$, then we say that the logics are *equiexpressive* and write $L_1 \equiv L_2$. If $L_1 \preceq L_2$ but it is not the case that $L_2 \preceq L_1$, then we say that L_1 is *(strictly) less expressive* than L_2 and write $L_1 \prec L_2$.

2.4 Normal form for F_1

For the parts of this paper concerning satisfiability, we introduce a convenient normal form, inspired by the Scott normal form for FO^2 [26] (a similar normal form is used also in [17] for the uniform F_1 over arbitrary structures). We say that an $F_1[\sigma_{nav}]$ formula φ is in *normal form* if φ has the following shape:

$$\bigwedge_{1\leq i\leq m_\exists} \forall y_0 \exists y_1 \ldots y_{k_i} \varphi_i^\exists \wedge \bigwedge_{1\leq i\leq m_\forall} \forall x_1 \ldots x_{l_i} \varphi_i^\forall, \tag{2.1}$$

where $\varphi_i^\exists = \varphi_i^\exists(y_0, y_1, \ldots, y_{k_i})$ and $\varphi_i^\forall = \varphi_i^\forall(x_1, \ldots, x_{l_i})$ are quantifier-free. Note that the width of φ is the maximum of the set $\{k_i + 1\}_{1\leq i\leq m_\exists} \cup \{l_j\}_{1\leq j\leq m_\forall}$. The following fact can be proved in the standard fashion.

LEMMA 2
For every $F_1[\rightarrow, \rightarrow^+]$ formula φ, one can compute in polynomial time an $F_1[\rightarrow, \rightarrow^+]$ formula φ' in normal form (over the signature extended by some fresh unary symbols) such that (i) any model of φ can be expanded to a model of φ' by appropriately interpreting new unary symbols and (ii) any model of φ' restricted to the signature of φ is a model of φ.

PROOF. (Sketch) We successively replace innermost subformulas ψ of φ of the form

$$\exists y_1, \ldots, y_k \varphi(y_0, y_1, \ldots, y_k)$$

by atoms $P_\psi(y_0)$, where P_ψ is a fresh unary symbol, and axiomatize P_ψ using two normal form conjuncts: $\forall y_0 \exists y_1, \ldots, y_k (P_\psi(y_0) \Rightarrow \varphi(y_0, y_1, \ldots, y_k))$ and $\forall y_0, y_1, \ldots, y_k \ (\varphi(y_0, y_1, \ldots, y_k) \Rightarrow P_\psi(y_0))$. □

Lemma 2 allows us, when dealing with satisfiability or when analysing the size and shape of models, to restrict attention to normal form formulas.

2.5 Types

In this section we define the classical notion of (atomic or quantifier-free) type. For $k \in \mathbb{N} \setminus \{0\}$ a *k-type* π over a signature $\sigma = \sigma_0 \cup \sigma_{nav}$ is a maximal consistent set of σ-literals over variables $x_1, \ldots, x_k$ (often identified with the conjunction of its elements). This means that π is a k-type iff

- for each $P \in \sigma_0$ and $1 \leq i \leq k$ either Px_i or $\neg Px_i$ belongs to π;
- for each $\rightleftharpoons \in \sigma_{nav}$ and $1 \leq i, j, \leq k$, $i \neq j$, either $x_i \rightleftharpoons x_j$ or $\neg x_i \rightleftharpoons x_j$ belongs to π;
- for each $1 \leq i < j \leq k$, either $x_i = x_j$ or $x_i \neq x_j$ belongs to π;
- if $\sigma_{nav} = \{\rightarrow, \rightarrow^+\}$ (respectively, σ_{nav} contains at at least one of $\downarrow, \downarrow_+$), then π is satisfiable in a word (resp., tree), i.e. there exists a word (resp., tree) $\mathfrak{M}$ and its elements $a_1, \ldots, a_k$ such that $\mathfrak{M} \models \pi(a_1, \ldots, a_k)$.

The last condition can be replaced by a purely syntactic one, listing conditions ensuring consistency with a linear or, respectively, tree shape of structures. Listing such conditions would be routine but slightly awkward, so we omit them here.

A *type* is a k-type for some $k \geq 1$. Note that a 1-type is fully characterized by a subset of σ_0.

We say that a tuple of elements $a_1 \ldots, a_k$ of a structure (word or tree) $\mathfrak{A}$ *realizes* a k-type π if $\mathfrak{A} \models \pi(a_1, \ldots, a_k)$. In this case we write $type^{\mathfrak{A}}(a_1, \ldots, a_k) = \pi$. Note that every tuple of elements of a structure realizes precisely one type.

3 Expressivity of one-dimensional fragment over words

It is known that the two-variable fragment, FO^2, is expressively equivalent over words and ω-words to UTL [10]. It is also equivalent to C^2, [3]. Also, GF^2, as a fragment of FO^2 containing UTL, has the same expressive power. Here we show that F_1 and UNFO share this expressivity. To properly handle UTL in the following theorem we identify its formulas with their standard translations to FO^2, which is a formula with one free variable.

THEOREM 1
Over the class of words and ω-words we have $\mathrm{UTL} \equiv \mathrm{GF}^2 \equiv \mathrm{FO}^2 \equiv \mathrm{C}^2 \equiv \mathrm{UNFO} \equiv \mathrm{F}_1$.

Let us first make a simple observation about the equivalence of UNFO and F_1. By Lemma 1, UNFO is not more expressive than F_1. In the opposite direction, given any $F_1[\rightarrow, \rightarrow^+]$ formula we can, using basic logical lows, convert it into a form in which the only non-unary negated formulas are atomic, i.e. are of the form $\neg x \rightarrow y$ or $\neg x \rightarrow^+ y$. They can be quite easily translated into formulas not using negations at all. Indeed, the former can be expressed as $y \rightarrow^+ x \vee x = y \vee \exists z(x \rightarrow z \wedge z \rightarrow^+ y)$ and the latter as $y \rightarrow^+ x \vee x = y$. This gives a polynomial translation from $F_1[\rightarrow, \rightarrow^+]$ into $\mathrm{UNFO}[\rightarrow, \rightarrow^+]$.

To complete the proof of Thm. 1 we need to show the equivalence of FO^2 and F_1. Obviously, FO^2 is a fragment of F_1. It remains to show how to translate F_1 into FO^2. The crux is to show how to handle formulas starting with a block of quantifiers.

LEMMA 3
For any $F_1[\rightarrow, \rightarrow^+]$ formula $\psi = \exists y_1 \ldots, y_k \psi_0(y_0, y_1, \ldots, y_k)$ with the free variable y_0 there exists an $FO^2[\rightarrow, \rightarrow^+]$ formula ψ' with one free variable such that for every word or ω-word $\mathfrak{M}$ and every $a \in M$, we have $\mathfrak{M} \models \psi[a]$ iff $\mathfrak{M} \models \psi'[a]$.

PROOF. We prove this lemma by induction over the quantifier depth of ψ, measured as the maximal nesting depth of maximal blocks of quantifiers rather than of individual quantifiers. W.l.o.g. we assume that every subformula of ψ starting with such a block indeed has a free variable (if it would not be the case we could always add a dummy variable). Let us take any

$$\psi = \exists y_1 \ldots, y_k \psi_0(y_0, y_1, \ldots, y_k), \tag{3.1}$$

and assume that its every subformula starting with a maximal block of quantifiers has an equivalent FO^2-formula. Convert ψ_0 into disjunctive form (treating subformulas starting with a quantifier as atoms) and distribute existential quantifiers over disjunctions, obtaining

$$\psi \equiv \bigvee_{i=1}^{s} \exists y_1 \ldots, y_k \psi_i(y_0, y_1, \ldots, y_k), \tag{3.2}$$

for some $s \in \mathbb{N}$, where each ψ_i is a conjunction of literals, subformulas with one free variable of the form $\exists z_1, \ldots, z_l \chi(y_j, z_1, \ldots, z_l)$ and negations of such subformulas.

Recall that the possible atoms are $P(y_i)$ for a unary symbol P, $y_i \rightarrow y_j$, $y_i \rightarrow^+ y_j$ and $y_i = y_j$, for some i, j.

An *ordering scheme* over variables $y_0, \ldots, y_k$ is a formula of the form $\eta_0(y_{i_0}, y_{i_1}) \wedge \eta_1(y_{i_1}, y_{i_2}) \wedge \ldots \wedge \eta_{k-1}(y_{i_{k-1}}, y_{i_k})$, where $\eta_i(v, w)$ is one of the following formulas: $v = w$, $v \rightarrow w$ or $v \rightarrow^+ w \wedge \neg v \rightarrow w$ and $i_0, i_1, i_2, \ldots, i_k$ is a permutation of $0, 1, \ldots, k$.

Consider now a single-disjunct $\exists y_1 \ldots, y_k \psi_i(y_0, y_1, \ldots, y_k)$ of (3.2) and replace it by the following disjunction over all possible ordering schemes δ over $y_0, \ldots, y_k$:

$$\bigvee_{\delta} \exists y_1 \ldots, y_k (\delta(y_0, \ldots, y_k) \wedge \psi_i^{\delta}(y_0, y_1, \ldots, y_k)), \tag{3.3}$$

where ψ_i^{δ} is obtained from ψ_i by replacing all atoms $y_i \rightarrow y_j$, $y_i \rightarrow^+ y_j$ and $y_i = y_j$, which are not bounded by the quantifiers from ψ_i by $\top$ or $\bot$, in accordance with the information recorded in δ. Let us write ψ_i^{δ} as $\bigwedge_{j=0}^{k} \psi_{i,j}^{\delta}(y_j)$, where $\psi_{i,j}^{\delta}(y_j)$ consists of the conjuncts with the free variable y_j. We now explain how to translate a single-disjunct

$$\exists y_1 \ldots, y_k (\delta(y_0, \ldots, y_k) \wedge \bigwedge_{j=0}^{k} \psi_{i,j}^{\delta}(y_j)) \tag{3.4}$$

of (3.3). Let $i_0, i_1, \ldots, i_k$ be the permutation used to generate δ, and let t be the index such that $i_t = 0$. By the inductive assumption we can replace in each $\psi_{i,j}^{\delta}(y_j)$ any conjunct of the form $\exists z_1, \ldots, z_l \chi(y_j, z_1, \ldots, z_l)$ by an equivalent two-variable conjunct with one free variable y_0. Thus, in turn, $\psi_{i,j}^{\delta}(y_j)$ can be replaced by an equivalent FO^2 formula $\psi_{i,j}'^{\delta}$ with one free variable.

We finally replace (3.4) by the conjunction of

$$\psi_{i,i_t}'^{\delta}(y_0), \tag{3.5}$$

$$\exists y(\eta_{t-1}(y, y_0) \wedge \psi_{i,i_{t-1}}'^{\delta}(y) \wedge \exists y_0(\eta_{t-2}(y_0, y) \wedge \psi_{i,i_{t-2}}'^{\delta}(y_0) \wedge \ldots)), \tag{3.6}$$

$$\exists y(\eta_t(y_0, y) \wedge \psi_{i,i_t}'^{\delta}(y) \wedge \exists y_0(\eta_{t+1}(y, y_0) \wedge \psi_{i,i_{t+1}}'^{\delta}(y_0) \wedge \ldots)), \tag{3.7}$$

in which (3.5) takes care of subformulas with the free variable y_0, (3.6) takes care of witnesses smaller than (or equal) to y_0, passing the word from y_0 to the left, and (3.7) takes care of witnesses greater than (or equal to) y_0, passing the word from y_0 to the right. Of course, in all the above formulas we appropriately rename the variables if necessary, so that only y_0 and y are used. □

Having translated formulas starting with blocks of quantifiers, we can easily translate other formulas with one free variable, since they are just Boolean combinations of the former and unary literals, all of them with the same free variable. This gives a translation from $\mathrm{F}_1[\rightarrow, \rightarrow^+]$ to $\mathrm{FO}^2[\rightarrow, \rightarrow^+]$.

Observe that starting from an $\mathrm{F}_1[\rightarrow, \rightarrow^+]$ formula this translation may produce a formula in $\mathrm{FO}^2[\rightarrow, \rightarrow^+]$, which is exponentially longer. Essentially, there are two sources of this exponential blow-up. The first is the transformation to disjunctive form, and the second is considering all possible permutations of variables quantified in a single block of quantifiers. The question whether this blow-up is necessary is left open.

4 Satisfiability of one-dimensional fragment over words

We next turn our attention to satisfiability of F_1 over words. Some upper bounds for the problem can be obtained using the translation to FO^2 given in the previous section. As this translation involves an exponential blow-up and the satisfiability problem for FO^2 over words is NExpTime-complete, this

gives a 2-NEXPTIME-upper bound. This could be improved by translating F_1 directly to UTL, which can be done without problems using the same method. As satisfiability of UTL is PSPACE-complete, we would get an EXPSPACE-upper bound this way.

However, we can do even better. We prove that the satisfiability problem for $F_1[\rightarrow, \rightarrow^+]$ both over words and ω-words is NEXPTIME-complete.

To this end we develop a contraction method involving a careful analysis of certain similarities between elements in a model and explain how to use it in order to obtain small model properties for $F_1[\rightarrow, \rightarrow^+]$ both over words and ω-words. The complexity result will then easily follow.

4.1 Profiles

Now we define *profiles*. Profiles are intended to abstract the information about relations of a given element to the other elements of a word. Namely, they say what the types of tuples are (of some bounded size) containing the given element. For convenience we will additionally distinguish types of tuples built of the elements located to the *left* and to the *right* from the given element.

We say that an element a of a word $\mathfrak{M}$ *realizes* (or *has*) a *k-profile* $prof_k^{\mathfrak{M}}(a) = (\mathcal{F}, \mathcal{L}, \mathcal{R})$ if $\mathcal{F}$ is the set of all s-types, $1 \leq s \leq k$, realized by tuples $a_1, a_2, \ldots, a_s$ such that $a = a_1$, $\mathcal{L}$ is the set of all s-types, $1 \leq s \leq k$, realized by tuples $a_1, a_2, \ldots, a_s$ such that $a = a_1$ and for all $2 \leq i \leq s$ we have $\mathfrak{M} \models a_i \rightarrow^+ a$, and, analogously, $\mathcal{R}$ is the set of all s-types, $1 \leq s \leq k$, realized by tuples $a_1, a_2, \ldots, a_s$ such that $a = a_1$ and for all $2 \leq i \leq s$ we have $\mathfrak{M} \models a \rightarrow^+ a_i$. Given a profile θ we will sometimes refer to its components with $\theta.\mathcal{F}$, $\theta.\mathcal{L}$ and $\theta.\mathcal{R}$. Note that $\theta.\mathcal{L} \cup \theta.\mathcal{R} \subseteq \theta.\mathcal{F}$. Note also that $\theta.\mathcal{F}$ is determined by $\theta.\mathcal{L}$ and $\theta.\mathcal{R}$, and vice versa.

LEMMA 4
Let $\mathfrak{M}$ be a word or an ω-word over $\sigma = \sigma_0 \cup \{\rightarrow, \rightarrow^+\}$ and $k > 0$ a natural number. Then the number of k-profiles realized in $\mathfrak{M}$ is bounded by a fixed function $\mathfrak{h}$ exponential in $|\sigma_0|$ and k.

PROOF. We introduce a binary relation $\sim_k$ on M as follows. For $a, b \in M$ we set $a \sim_k b$ iff the one-type of $a + i$ is equal to the 1-type of $b + i$ (or both $a + i$ and $b + i$ do not exist) for all $-k < i < k$. Clearly, $\sim_k$ is an equivalence relation and the number of its equivalence classes is bounded by $(2^{|\sigma_0|})^{2k-1} + 2k - 2 = 2^{|\sigma_0| \cdot (2k-1)} + 2k - 2$ (the number of combinations of 1-types of elements $a - k + 1, \ldots, a + k - 1$ plus the classes of the first $k - 1$ and, in the case of a finite word, the last $k - 1$ elements).

We show that if $a \sim_k b$ and $\mathfrak{M} \models a \rightarrow^+ b$ then for every type π if $\pi \in prof_k^{\mathfrak{M}}(b).\mathcal{R}$ then $\pi \in prof_k^{\mathfrak{M}}(a).\mathcal{R}$. Take any $\pi \in prof_k^{\mathfrak{M}}(b).\mathcal{R}$ and let $b_1, b_2, \ldots, b_k$, with $b_1 = b$ be its realization. Let $u_1, \ldots, u_k$ be a permutation of $\{1, \ldots, k\}$ such that $u_1 = 1$ and $\mathfrak{M} \models b_{u_i} \rightarrow^+ b_{u_{i+1}} \vee b_{u_i} = b_{u_{i+1}}$ for $1 \leq i < k$, that is a permutation 'sorting' the elements of the given tuple. Let l be the maximal index such that $\mathfrak{M} \models b_{u_i} \rightarrow b_{u_{i+1}} \vee b_{u_i} = b_{u_{i+1}}$ for all $1 \leq i \leq l$. Consider now the tuple $a_{u_1}, \ldots, a_{u_k}$, such that $a_{u_1} = a$, $a_{u_i} = a + (i-1)$ for $1 < i \leq l$, and $a_{u_i} = b_{u_i}$ for $l < i \leq k$. Note that $type^{\mathfrak{M}}(a_{u_1}, \ldots, a_{u_k}) = type^{\mathfrak{M}}(b_{u_1}, \ldots, b_{u_k})$ and thus also $type^{\mathfrak{M}}(a_1, \ldots, a_k) = type^{\mathfrak{M}}(b_1, \ldots, b_k) = \pi$. Since $a = a_1$ this means $\pi \in prof_k^{\mathfrak{M}}(a).\mathcal{R}$.

Strictly analogously we can show that if $\pi \in prof_k^{\mathfrak{M}}(a).\mathcal{L}$ then $\pi \in prof_k^{\mathfrak{M}}(b).\mathcal{L}$.

Thus, when moving along the elements of a single equivalence class of $\sim_k$ in $\mathfrak{M}$ from left to right, the $\mathcal{R}$-components of the profiles of elements either stay unchanged or diminish, and the $\mathcal{L}$-components either stay unchanged or grow. As the set of types contained in each component is determined by the set of k-types in this component, and as the number of k-types in a component can be roughly estimated by $(2^{|\sigma_0|})^k \cdot 5k(k-1)$ (the number of possible assignments of one-types to the elements of a tuple of k elements, times the number of possible binary connections: equal, y

a successor of x, x a successor of y, y to the left from x but not the successor, y to the right from x but not the predecessor, for every pair of elements) it follows that the $\sim_k$-equivalent elements may have at most $2 \cdot (2^{|\sigma_0|})^k \cdot 5k(k-1)+1$ different k-profiles (recall that the $\mathcal{F}$-components are determined by $\mathcal{L}$- and $\mathcal{R}$-components). Finally, the total number of k-profiles is bounded by $(2^{|\sigma_0|\cdot(2k-1)} + 2k - 2) \cdot (2 \cdot 2^{|\sigma_0|\cdot k} \cdot 5k(k-1)+1)$, which is indeed exponential in both k and $|\sigma_0$□

The notion of profiles can be easily connected to satisfaction of normal form formulas. Given a normal form formula φ of width k we say that a k-profile θ is *compatible* with φ if

- for every conjunct $\forall x_1 \dots x_{l_i} \varphi_i^{\forall}(x_1 \dots x_{l_i})$ of φ and every l_i-type $\pi \in \mathcal{F}$, we have $\pi \models \varphi_i^{\forall}$.
- for every conjunct $\forall y_0 \exists y_1 \dots y_{k_i} \varphi_i^{\exists}(y_0, y_1 \dots y_{k_i})$ of φ there is a (k_i+1)-type $\pi \in \theta.\mathcal{F}$ such that $\pi \models \varphi_i^{\exists}(x_1, \dots, x_{k_i+1})$.

It is straightforward to see:

LEMMA 5
A normal form formula φ of width k is satisfied in a word (ω-word) $\mathfrak{M}$ iff every k-profile realized in $\mathfrak{M}$ is compatible with φ.

4.2 Contraction

We are ready to prove the contraction lemma. Namely, we observe that removing a fragment of a word between two realizations of the same profile (including one of them and excluding the other) does not change the profiles of the surviving elements.

LEMMA 6
Let $\mathfrak{M} = \mathfrak{M}_1 c \mathfrak{M}_2 d \mathfrak{M}_3$ be a word or ω-word, and $k > 0$ a natural number. Assume that $prof_k^{\mathfrak{M}}(c) = prof_k^{\mathfrak{M}}(d)$ and $\mathfrak{M}' = \mathfrak{M}_1 c \mathfrak{M}_3$. Then, for every $b \in M'$, we have $prof_k^{\mathfrak{M}'}(b) = prof_k^{\mathfrak{M}}(b)$.

PROOF. Consider the case where $b \in M_1 \cup \{c\}$. Note that the prefix of $\mathfrak{M}$ ending in b is then equal to the prefix of $\mathfrak{M}'$ ending in b. It follows that $prof_k^{\mathfrak{M}'}(b).\mathcal{L} = prof_k^{\mathfrak{M}}(b).\mathcal{L}$. It remains to show that $prof_k^{\mathfrak{M}'}(b).\mathcal{R} = prof_k^{\mathfrak{M}}(b).\mathcal{R}$.

To show that $prof_k^{\mathfrak{M}'}(b).\mathcal{R} \subseteq prof_k^{\mathfrak{M}}(b).\mathcal{R}$, take any s-type π, $1 \leq s \leq k$, belonging to $prof_k^{\mathfrak{M}'}(b).\mathcal{R}$ and let $b_1, \dots, b_s$ be a realization of π in $\mathfrak{M}'$, with $b_1 = b$. Let $u_1, \dots, u_s$ be a permutation of $\{1, \dots, s\}$ such that $u_1 = 1$ and $\mathfrak{M}' \models b_{u_i} \rightarrow^+ b_{u_{i+1}} \vee b_{u_i} = b_{u_{i+1}}$ for $1 \leq i < s$. Let l be the maximal index such that $b_{u_l} \in M_1 \cup \{c\}$. Since $b_{u_1} = b \in M_1 \cup \{c\}$, l is well defined. Let $\pi' = type^{\mathfrak{M}}(d, b_{u_{l+1}), \dots, b_{u_s}}$ and observe that $\pi' \in prof_k^{\mathfrak{M}}(d).\mathcal{R}$. By assumption $\pi' \in prof_k^{\mathfrak{M}}(c).\mathcal{R}$, and thus there is a realization $c, a_{u_{l+1}}, \dots, a_{u_s}$ of π' in $\mathfrak{M}$. Set $a_{u_i} := b_{u_i}$ for $1 \leq i \leq l$. It is now not difficult to see that $type^{\mathfrak{M}}(a_{u_1}, \dots, a_{u_s}) = type^{\mathfrak{M}'}(b_{u_1}, \dots b_{u_s})$ and thus also $type^{\mathfrak{M}}(a_1, \dots, a_s) = type^{\mathfrak{M}'}(b_1, \dots b_s) = \pi$. Since $a_1 = b_1 = b$ it follows that $\pi \in prof_k^{\mathfrak{M}}(b)$.

To show that $prof_k^{\mathfrak{M}}(b).R_i \subseteq prof_k^{\mathfrak{M}'}(b).R_i$ we take any s-type π, $1 \leq s \leq k$ belonging to $prof_k^{\mathfrak{M}}(b).R_i$ and let $b_1, \dots, b_s$ be a realization of π in $\mathfrak{M}$, with $b_1 = b$. Let $u_1, \dots, u_s$ be a permutation of $\{1, \dots, s\}$ such that $u_1 = 1$ and $\mathfrak{M} \models b_{u_i} \rightarrow^+ b_{u_{i+1}} \vee b_{u_i} = b_{u_{i+1}}$ for $1 \leq i < s$. Let l be the maximal index such that $b_{u_l} \in M_1 \cup \{c\}$. Again note that l is well defined. Let $\pi' = type^{\mathfrak{M}}(c, b_{u_{l+1}), \dots, b_{u_s}}$ and observe that $\pi' \in prof_k^{\mathfrak{M}}(c).\mathcal{R}$. By assumption $\pi' \in prof_k^{\mathfrak{M}}(d).\mathcal{R}$, and thus there is a realization $d, a_{u_{l+1}}, \dots, a_{u_s}$ of π' in $\mathfrak{M}$ with the a_j from M_3. Let $a_{u_i} := b_{u_i}$ for $1 \leq i \leq l$. It is not difficult to see that $type^{\mathfrak{M}'}(a_{u_1}, \dots, a_{u_s}) = type^{\mathfrak{M}}(b_{u_1}, \dots, b_{u_s})$ and thus also $type^{\mathfrak{M}'}(a_1, \dots, a_s) = type^{\mathfrak{M}}(b_1, \dots, b_s) = \pi$ and since $a_1 = b$ it follows that $\pi \in prof_k^{\mathfrak{M}'}(b).\mathcal{R}$.

The case when $a' \in M_3$ can be treated symmetrically: this time we get the equality of the $\mathcal{R}$-components of the profiles for free and to show the equality the $\mathcal{L}$-components we use the equality of the $\mathcal{L}$-components of the profiles of c and d. □

4.3 Surgery on ω-words

In this section we work over ω-words. Namely, we show how to transform a given ω-word into a periodic one without introducing any new profiles.

LEMMA 7
Let $\mathfrak{M}$ be an ω-word and $k > 0$ a natural number. Let $\mathfrak{M}_0$ be the shortest prefix of $\mathfrak{M}$ such that it contains all the elements having the k-profiles, which are realized finitely many times in $\mathfrak{M}$. Note that $\mathfrak{M}_0$ has length at least $k-1$. Let a_* be the first element not belonging to M_0, and θ_* its k-profile. Let $\mathfrak{M}_1$ be the shortest fragment of $\mathfrak{M}$ such that

- it starts at a_*,
- contains a realization of every k-profile, which is realized in $\mathfrak{M}$ infinitely many times,
- ends at an element whose successor b_* has k-profile θ_*.

Consider the ω-word $\mathfrak{M}' = \mathfrak{M}_0\mathfrak{M}_1^\omega$, that is the word obtained by concatenating $\mathfrak{M}$ and infinitely many copies of $\mathfrak{M}_1$. We will call its initial fragment $\mathfrak{M}_0$ and the subsequent copies of $\mathfrak{M}_1$ *blocks*. Let $f : M' \to M$ be the function returning for every $a' \in M'$ the element from $\mathfrak{M}$, which a' is a copy of. Then, for every $a' \in M'$, $prof_k^{\mathfrak{M}'}(a') = prof_k^{\mathfrak{M}} f(a)$.

PROOF. Let us start with a simple observation.

CLAIM. For every $-k < i < k$ either both $a'+i$ and $f(a')+i$ do not exist or their 1-types are identical.
□

PROOF. The claim is obvious if a' and $a'+i$ belong to the same block, and easily follows from the requirement that a_* and b_* have the same k-profiles in the other case (for this observe also that $\mathfrak{M}_0$ contains at least k elements, which follows from the fact that the profiles of the first k elements of a word are unique). □

Let $g : M \to M'$ be the partial function defined on $M_0 \cup M_1$ such that $g(a) = a$ if $a \in M_0$ and $g(a)$ is the counterpart of a in the first copy of M_1.

Take any $a' \in \mathfrak{M}'$. First, let us consider the $\mathcal{L}$-components of the profiles. Take any $\pi \in prof_k^{\mathfrak{M}} f(a').\mathcal{L}$ and let a tuple $\bar{a}_\pi$ be its realization. Let us write the elements of $\bar{a}_\pi$, in the increasing order, removing duplicates, as $\bar{a}_\pi^{sort} = a_1^0, \ldots, a_{s_0}^0, a_1^1, \ldots, a_{s_1}^1, \ldots, a_1^l, \ldots, a_{s_l}^l = f(a')$, where for each i, $a_1^i, \ldots, a_{s_i}^i$ is a maximal sequence of consecutive elements of $\mathfrak{M}$. Observe now, using Claim 6, that the structure on the sequence $g(a_1^0), \ldots, g(a_{s_0}^0), \ldots, g(a_1^{l-1}), \ldots, g(a_{s_{l-1}}^{l-1}), a' - (s_l - 1), \ldots, a'-1, a'$ is isomorphic to the structure on $\bar{a}_\pi^{sort}$. It follows that $\pi \in prof_k^{\mathfrak{M}'}(a').\mathcal{L}$.

Take now $\pi \in prof_k^{\mathfrak{M}'}(a').\mathcal{L}$. Let $\bar{a}_\pi$ be its realization, and let us write the elements of $\bar{a}_\pi$, in the increasing order, removing duplicates, as $\bar{a}_\pi^{sort} = a_1^0, \ldots, a_{s_0}^0, a_1^1, \ldots, a_{s_1}^1, \ldots, a_1^l, \ldots, a_{s_l}^l = a'$. Take the maximal u such that $a_{s_u}^u \in M_0$. For all $i \leq u$ and all j let $b_j^i := g(a_j^i)$. Now, for $i = u+1, \ldots, k$ repeat the following. Consider the sequence $f(a_{s_i}^i) - s_i + 1, \ldots, f(a_{s_i}^i) - 1, f(a_{s_i}^i)$. By Claim 6 the structure on this sequence is isomorphic to the structure on $a_1^i, \ldots, a_{s_i}^i$. Let $b_{s_i}^i$ be an element of $\mathfrak{M}$

whose profile is identical to the profile of $f(a^i_{s_i})$ and is located at least $k+1$ positions to the right from $b^{i-1}_{s_{i-1}}$. Such an element exists since the profile of $a^i_{s_i}$ is realized in $\mathfrak{M}$ infinitely many times. For $j = 1, \ldots, s_i - 1$ take $b^i_j := b^i_{s_i} - s_i + j$. Note that the structure on the sequence $b^0_1, \ldots, b^0_{s_0}, a^1_1, \ldots, b^1_{s_1}, \ldots, b^l_1, \ldots, b^l_{s_l}$ is isomorphic to the structure on the sequence $\bar{a}^{sort}_\pi$. Thus, $\pi \in prof^{\mathfrak{M}}_k(b^l_{s_l}).\mathcal{L}$. But $prof^{\mathfrak{M}}_k(b^l_{s_l}) = prof^{\mathfrak{M}}_k f(a^l_{s_l}) = prof^{\mathfrak{M}}_k f(a')$. So $\pi \in prof^{\mathfrak{M}}_k f(a').\mathcal{L}$.

The reasoning for the equality of the $\mathcal{R}$-components is similar but simpler and we omit it here.

4.4 Complexity

Using the tools prepared in the previous section, we can now show the following small model properties.

LEMMA 8
Every $F_1[\rightarrow, \rightarrow^+]$ formula φ satisfiable over a finite word has a model of size bounded exponentially in $\|\varphi\|$.

PROOF. Due to Lemma 2, we can assume that φ is in normal form. Let k be its width. We take any finite model $\mathfrak{M} \models \varphi$ and perform on it the contraction procedure from Lemma 6, as many times as possible, i.e. if it still contains a pair of distinct elements with the same k-profile. By Lemma 4 the number of elements in the resulting model $\mathfrak{M}'$ is bounded exponentially in $\|\varphi\|$. By Lemma 6, the profiles of the elements in $\mathfrak{M}'$ are retained from $\mathfrak{M}$. As $\mathfrak{M} \models \varphi$, these profiles are compatible with φ. By Lemma 5, we get that $\mathfrak{M}'$ indeed satisfies φ. □

LEMMA 9
Every $F_1[\rightarrow, \rightarrow^+]$ formula φ satisfiable over an ω-word has a model $\mathfrak{N}_0\mathfrak{N}_1^\omega$ where both $|N_0|$ and $|N_1|$ are bounded exponentially in $\|\varphi\|$.

PROOF. Due to Lemma 2 we can assume that φ is in normal form. Let k be its width. We take an arbitrary ω-model $\mathfrak{M} \models \varphi$. Let $\mathfrak{M} = \mathfrak{M}_0\mathfrak{M}_1\mathfrak{M}_2$ where $\mathfrak{M}_0$ and $\mathfrak{M}_1$ are as in Lemma 7. Using Lemma 6 for $\mathfrak{M}$, contract its fragments $\mathfrak{M}_0$ and $\mathfrak{M}_1$ to, resp., $\mathfrak{N}_0$ and $\mathfrak{N}_1$ so that every k-profile from $\mathfrak{M}$ is realized at most once in $\mathfrak{N}_0$ and at most once in $\mathfrak{N}_1$. By Lemma 4 the number of elements in both $\mathfrak{N}_0$ and $\mathfrak{N}_1$ are bounded exponentially in $\|\varphi\|$. Note that $\mathfrak{N}_0\mathfrak{N}_1\mathfrak{M}_2 \models \varphi$. By Lemma 7 the k-profiles of elements in $\mathfrak{N}_0\mathfrak{N}_1^\omega$ are retained from $\mathfrak{N}_0\mathfrak{N}_1\mathfrak{M}_2$ and the latter are realized in $\mathfrak{M}$. By Lemma 5 we get that $\mathfrak{N}_0\mathfrak{N}_1^\omega$ is indeed a model of φ. □

Finally, we can state the main complexity result of this section.

THEOREM 2
The satisfiability problems for $F_1[\rightarrow, \rightarrow^+]$ over words (ω-words) is NEXPTIME-complete.

PROOF. For a given $F_1[\rightarrow, \rightarrow^+]$ formula φ, convert it into its normal form φ'. Then guess a finite model of φ' of size bounded exponentially as guaranteed by Lemma 8 (exponentially bounded initial and periodic parts of a regular ω-model as guaranteed by Lemma 9) and check that all the profiles realized in this model (in the model generated by the guessed parts) indeed are compatible with φ'. In the case of finite words the profiles are computed in an exhaustive way: for every element a of the guessed model $\mathfrak{M}$ we consider all possible tuples $a_2, \ldots, a_s$ of at most $k-1$ elements and add $type^{\mathfrak{M}}(a, a_2, \ldots, a_s)$ to the profile.

In the case of ω-words, note that all the k-profiles realized in the periodic model are realized in the finite model in which the periodic part is taken $2k$ times (k times assuming that the length of the periodic part is bigger than 1). Thus, it suffices to compute the profiles in such finite model. □

We also get the following corollary concerning UNFO.

COROLLARY 1
The satisfiability problems for UNFO over words (ω-words) is NEXPTIME-complete.

PROOF. The upper bound follows from Lemma 1 and Thm. 2. The lower bound follows from NEXPTIME-hardness of FO^2 with only unary relations (without any structure). □

4.5 Undecidable extensions

The two variable fragment over words, $FO^2[\rightarrow, \rightarrow^+]$ remains decidable when extended in various orthogonal directions. Here we show that three such important analogous extensions are undecidable in the case of F_1.

4.5.1 Data words A *data word* (*ω-data word*) is a word (ω-word) with an additional binary relation $\sim$ that is required to be interpreted as an equivalence relation and that is intended to model the equality of data from a potentially infinite alphabet. Data words are motivated by their connections to XML. FO^2 over data words becomes at least as hard as reachability in Petri nets [4]. Nevertheless, the satisfiability problem remains decidable. We show that $F_1[\rightarrow, \rightarrow^+]$ over data words is undecidable, even in the absence of $\rightarrow^+$.

THEOREM 3
The satisfiability problem for $F_1[\rightarrow]$ over finite data words and over ω-data-words is undecidable.

PROOF. We employ the standard apparatus of tiling systems. A *tiling system* is a quintuple $\mathcal{T} = \langle C, c_0, c_1, Hor, Ver\rangle$, where C is a non-empty, finite set of *colours*, c_0, c_1 are elements of C, and *Hor*, *Ver* are binary relations on C called the *horizontal* and *vertical* constraints, respectively. We say that $\mathcal{T}$ *tiles* the $m \times n$ grid if there is a function function $f : \{0, 1, \ldots, m-1\} \times \{0, 1, \ldots, n-1\} \rightarrow C$ such that $f(0,0) = c_0$, $f(m-1, n-1) = c_1$, for all $0 \leq i < m-1$, $0 \leq j \leq n-1$ we have $\langle f(i,j), f(i+1,j)\rangle$ is in *Hor*, and for all $0 \leq i < m$, $0 \leq j < n-1$ we have $\langle f(i,j), f(i,j+1)\rangle$ is in *Ver*. It is well know that the problem of checking if for a given tiling system $\mathcal{T}$ there are m, n such that $\mathcal{T}$ tiles the $m \times n$ grid is undecidable. The problem remains undecidable if we require m to be even and n odd.

To show undecidability of the satisfiability problem for $F_1[\rightarrow, \sim]$ over finite words we construct a formula $\Phi_{\mathcal{T}}$, which is satisfied in a finite word iff $\mathcal{T}$ tiles the $m \times n$ grid for some even m and odd n. We begin the construction of $\Phi_{\mathcal{T}}$ with enforcing that its model is a finite grid-like structure, in which the relation $\rightarrow$ forms a snake-like path from its lower-left corner to the upper-right corner and the equivalence relation connects some elements from neighbouring columns. See Figure 1. As mentioned, we assume that the number of columns is odd and the number of rows is even. We employ the following unary predicates: B, T, E_c, E_r, whose intended purpose is to mark elements in the bottom row, top row, even columns and even rows, respectively.

The first two formulas say that the lower left and upper right corners of the grid exist:

$$\exists x(Bx \wedge \neg Tx \wedge E_c x \wedge E_r x \wedge \neg\exists y(y \rightarrow x)) \tag{4.1}$$

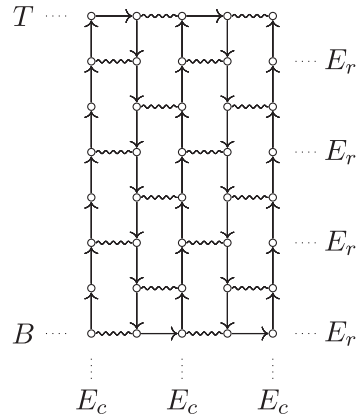

Figure 1 The grid-like structure used to show undecidability of $\mathrm{F}_1[\rightarrow, \sim]$. Solid arrows represent $\rightarrow$; wavy lines represent $\sim$.

$$\exists x(Tx \wedge \neg Bx \wedge E_c x \wedge \neg E_r x \wedge \neg \exists y(x \rightarrow y)) \tag{4.2}$$

Next we take care of the $\rightarrow$ relation, ensuring that it respects the intended meaning of the unary predicates:

$$\begin{aligned} \forall xy \quad & (x \rightarrow y \Rightarrow \\ & (E_c x \wedge E_c y \Rightarrow (\neg By \wedge \neg Tx \wedge (E_r x \leftrightarrow \neg E_r y)) \wedge \\ & (E_c x \wedge \neg E_c y \Rightarrow (Tx \wedge Ty \wedge \neg Bx \wedge \neg By \wedge \neg E_r x \wedge \neg E_r y)) \wedge \\ & (\neg E_c x \wedge E_c y \Rightarrow (Bx \wedge By \wedge \neg Tx \wedge \neg Ty \wedge E_r x \wedge E_r y)) \wedge \\ & (\neg E_c x \wedge \neg E_c y \Rightarrow (\neg Bx \wedge \neg Ty \wedge (E_r \leftrightarrow \neg E_r y)))) \end{aligned} \tag{4.3}$$

Further, we enforce the appropriate $\sim$-connections. (We abbreviate a formula guaranteeing that $x_1, \ldots, x_k$ agree on the E_c-predicate by $\mathit{SameColumn}(x_1, \ldots, x_k)$.)

$$\forall xyzt(x \rightarrow y \wedge y \rightarrow z \wedge z \rightarrow t \wedge Ty \wedge Tz \Rightarrow x \sim t) \tag{4.4}$$

$$\forall xyzt(x \rightarrow y \wedge y \rightarrow z \wedge z \rightarrow t \wedge By \wedge Bz \Rightarrow x \sim t) \tag{4.5}$$

$$\begin{aligned} & \forall xyztuw(\mathit{SameColumn}(x, y, z) \wedge \mathit{SameColumn}(t, u, w) \wedge \\ & \quad x \rightarrow y \wedge y \rightarrow z \wedge z \sim t \wedge t \rightarrow u \wedge u \rightarrow w \Rightarrow x \sim w \end{aligned} \tag{4.6}$$

And finally, we say that T and B are appropriately propagated:

$$\forall xy(x \sim y \Rightarrow (Tx \leftrightarrow Ty) \wedge (Bx \leftrightarrow By)) \tag{4.7}$$

$$\begin{aligned} & \forall xyzt(\mathit{SameColumn}(x, y) \wedge \mathit{SameColumn}(z, t) \wedge \\ & \quad x \rightarrow y \wedge y \sim z \wedge z \rightarrow t \Rightarrow (Tx \leftrightarrow Tt) \wedge (Bx \leftrightarrow Bt)) \end{aligned} \tag{4.8}$$

Formulas (4.1)–(4.8) ensure that all the vertical segments of the snake-like path are of the same length and thus that any model indeed looks like in Figure 1. It remains to encode the tiling problem.

We use a unary predicate P_c for each $c \in C$. We say that each node of the grid is coloured by precisely one colour from C, that $(0,0)$ is coloured by c_0 and that $(m-1, n-1)$ is coloured with c_1:

$$\forall x(\bigvee_{c\in C} P_c(x) \wedge \bigwedge_{c\neq d} \neg(P_c(x) \wedge P_d(x))), \tag{4.9}$$

$$\forall x((\neg\exists y y{\rightarrow}x) \Rightarrow P_{c_0}(x)), \tag{4.10}$$

$$\forall x((\neg\exists y x{\rightarrow}y) \Rightarrow P_{c_1}(x)). \tag{4.11}$$

Let us abbreviate by $\Theta_H(x,y)$ the formula $\bigwedge_{\langle c,d\rangle\notin Hor}(\neg P_c(x) \wedge \neg P_d(y))$ stating that x, y respect the horizontal constraints of $\mathcal{T}$ and by $\Theta_V(x,y)$ the analogous formula for vertical constraints. We take care of vertical adjacencies:

$$\forall xy(E_c(x) \wedge E_c(y) \wedge x{\rightarrow}y \vee \neg E_c(x) \wedge \neg E_c(y) \wedge y{\rightarrow}z \Rightarrow \Theta_V(x,y)), \tag{4.12}$$

and of horizontal adjacencies:

$$\forall xyzt(x{\rightarrow}y \wedge y{\rightarrow}z \wedge z{\rightarrow}t \wedge Ty \wedge Tz \Rightarrow \Theta_H(y,z)), \tag{4.13}$$

$$\forall xyzt(x{\rightarrow}y \wedge y{\rightarrow}z \wedge z{\rightarrow}t \wedge By \wedge Bz \Rightarrow \Theta_H(y,z))), \tag{4.14}$$

$$\begin{aligned}\forall xyztuw(&SameColumn(x,y,z) \wedge SameColumn(t,u,w) \wedge z \sim t \wedge\\ &x{\rightarrow}y \wedge y{\rightarrow}z \wedge t{\rightarrow}u \wedge u{\rightarrow}w \Rightarrow \Theta_H(x,w){\wedge}\Theta_H(y,u){\wedge}\Theta_H(z,t)).\end{aligned} \tag{4.15}$$

Let $\Phi_{\mathcal{T}}$ be the conjunction of (4.1)–(4.15). From any model of $\Phi_{\mathcal{T}}$, we can read off a tiling of an $m \times n$ grid by inspecting the colours assigned to the elements of the model. On the other hand, given any tiling for $\mathcal{T}$, we can construct a finite model of $\Phi_{\mathcal{T}}$ in the obvious way. We leave the detailed arguments to the reader.

The case of ω-words can be treated essentially in the same way. We just mark one element in a model, corresponding to the upper-right corner of the grid, with a special unary symbol, and relativize all our formulas to positions smaller than this element (marked with another fresh unary symbol). In effect, it is irrelevant what happens in the infinite fragment of a model starting from this marked element.

What is probably worth commenting is that in our undecidability proof we use the equivalence relation $\sim$ in a very limited way, actually not benefiting from its transitivity or symmetry. In fact, the transitivity of $\sim$ does not help, being rather an obstacle in our construction. □

4.5.2 Uninterpreted binary relation Both $FO^2[\rightarrow]$ and $FO^2[\rightarrow^+]$ remain decidable when, besides $\rightarrow$ or $\rightarrow^+$, the signature may contain other binary symbols, whose interpretation is not fixed ([24], [9]). We can easily see that this is not the case for F_1.

THEOREM 4
The satisfiability problem for $F_1[\rightarrow]$ and $F_1[\rightarrow^+]$ is undecidable when an additional uninterpreted binary relation is available.

PROOF. We can use the proof of Thm 3 without assuming that $\sim$ is an equivalence relation. □

Actually, undecidability holds even without using the linear order: we can simply axiomatize grid-like structures using a single binary predicate and some unary coordinate predicates. This can

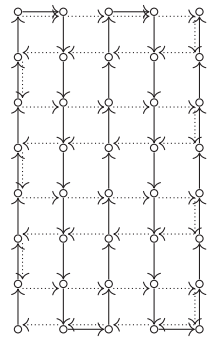

FIGURE 2 The grid-like structure used to show undecidability of $F_1[\rightarrow_1, \rightarrow_2]$. Solid arrows represent $\rightarrow_1$; dotted arrows represent $\rightarrow_2$.

be done by a simple modification of the undecidability proof for F_1 over the class of all structure [13], which uses two binary symbols.

4.5.3 Two linear orders Let us now consider a variation in which we have two linear orders rather than just one. The second linear order may be interpreted, e.g. as a comparison relation on data values. $FO^2[\rightarrow_1, \rightarrow_2]$, the two-variable fragment accessing the linear orders through their successor relations only, is decidable in NEXPTIME [9]. Showing that a corresponding variant of F_1 is undecidable is again easy. We can define a grid-like structure using the first linear order to form a snake-like path as in the proof of Thm. 3 and the second to form another snake-like path, starting in the upper-left corner, ending in the lower-left corner and going horizontally through our grid, with steps down only on the borders. See Figure 2. The required structure can be defined with help of some additional unary predicates. Since the details of the construction do not differ significantly from the details of the proof of Thm. 3 we omit them here.

THEOREM 5
The satisfiability problem for $F_1[\rightarrow_1, \rightarrow_2]$ is undecidable.

5 Expressivity of one-dimensional fragment over trees

In this section we compare the expressive power of F_1 with related logics in the case of two important tree signatures: $\{\downarrow, \downarrow_+, \rightarrow, \rightarrow^+\}$ and $\{\downarrow, \downarrow_+\}$, or, in other words, over the class of XML trees and unordered trees.

As in the case of UTL over words we will identify CoreXPath formulas with their standard translations into GF^2, which are formulas with one free variable.

It turns out that over ordered trees all logics we are interested in are equiexpressive, as it was in the case over words.

THEOREM 6
For $\sigma = \{\downarrow, \downarrow_+, \rightarrow, \rightarrow^+\}$ we have $\text{CoreXPath} \equiv GF^2[\sigma] \equiv FO^2[\sigma] \equiv C^2[\sigma] \equiv \text{UNFO}[\sigma] \equiv F_1[\sigma]$.

PROOF. Let us first observe that F_1 is equivalent to UNFO. The argument is similar to the one in the case of words. Due to Lemma 1, UNFO is not more expressive than F_1. In the opposite direction, given any $F_1[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$ formula we can, using basic logical laws, convert it into a form in which

the only non-unary negated formulas are atomic, i.e. they are of the form $\neg x{\downarrow}y$, $\neg x{\downarrow_+}y$, $\neg x{\rightarrow}y$ or $\neg x{\rightarrow^+}y$. Taking into consideration the shape of trees, we translate them into formulas not using negations at all as follows:

$$\begin{aligned}
trans(\neg x{\downarrow_+}y) &:= \exists z((z = x \vee z{\downarrow_+}x) \wedge (y = z \vee \exists t(z{\rightarrow^+}t \vee t{\rightarrow^+}z) \wedge (y = t \vee t{\downarrow_+}y)))\\
trans(\neg x{\downarrow}y) &:= trans(\neg x{\downarrow_+}y) \vee \exists z(x{\downarrow}z \wedge z{\downarrow_+}y)\\
trans(\neg x{\rightarrow^+}y) &:= x{\downarrow_+}y \vee \exists z(z{\rightarrow^+}x \wedge (y = z \vee z{\downarrow_+}y)) \vee \exists z(x{\rightarrow^+}z \wedge z{\downarrow_+}y) \vee\\
&\qquad \exists z(z{\downarrow_+}x \wedge (y = z \vee \exists t(z{\rightarrow^+}t \vee t{\rightarrow^+}z) \wedge (y = t \vee t{\downarrow_+}y)))\\
trans(\neg x{\rightarrow}y) &:= trans(\neg x{\rightarrow^+}y) \vee \exists z(x{\rightarrow}z \wedge z{\rightarrow^+}y)
\end{aligned} \tag{5.1}$$

This gives a polynomial translation from $F_1[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$ into $UNFO[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$ and establishes their equivalence.

Further, a translation from $UNFO[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$ to CoreXPath is given in [27]. CoreXPath is a fragment of GF^2, GF^2 is a fragment of FO^2 and FO^2 is a fragment of C^2. Finally, C^2 can be easily translated to F_1 (over any class of structures): consider, e.g. a subformula of the form $\exists^{\geq k} y\psi(x,y)$ and note that it can be written as $\exists y_1, \ldots, y_k(\bigwedge_{i\neq j} y_i \neq y_j \wedge \bigwedge_i \psi(x,y_i))$. This completes the proof in the case of XML trees. □

Over unordered trees the situation if more interesting and the considered languages turn out to vary in their expressive power, in particular F_1 is more expressive than UNFO and FO^2. Interestingly, it is, however, equivalent to C^2. The full picture is as follows.

THEOREM 7
For $\sigma = \{\downarrow, \downarrow_+\}$ we have $CoreXPath[\sigma] \equiv GF^2[\sigma] \equiv UNFO[\sigma] \prec FO^2[\sigma] \prec C^2[\sigma] \equiv F_1[\sigma]$.

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. The C^2 formula $\exists^{\geq 3} y\, x{\downarrow_+}y$ distinguishes $\mathfrak{T}_3$ and $\mathfrak{T}_2$ (it is true at the root of the former and false in the latter), while the FO^2 formula $\exists y(\neg x{\downarrow_+}y \wedge \neg y{\downarrow_+}x \wedge x \neq y)$ distinguishes $\mathfrak{T}_2$ and $\mathfrak{T}_1$. It is not difficult to see that FO^2 cannot distinguish between $\mathfrak{T}_3$ and $\mathfrak{T}_2$ (a simple 2-pebble game argument, cf. [3]) and that GF^2 cannot distinguish between $\mathfrak{T}_2$ and $\mathfrak{T}_1$ (use guarded bisimulations, cf. [1]). These observations justify the relations $GF^2 \prec FO^2$ and $FO^2 \prec C^2$.

C^2 can be translated to F_1 as in the previous proof. Translation in the opposite direction is a harder task and we devote for it a separate section.

It remains to show the equivalence of CoreXPath, GF^2 and UNFO. To this end we provide a translation from GF^2 to UNFO and from UNFO to CoreXPath. The cycle is then closed by recalling that CoreXPath is a fragment of GF^2.

From GF^2 to UNFO. Take any $GF^2[\downarrow, \downarrow_+]$ formula and write it without using the universal quantifiers. Then push down all the negations with the exception of those standing just before the existential quantifiers (they are allowed in UNFO since a GF^2 formula starting with an existential quantifier has at most one free variable). Let φ be the resulting formula. We need to eliminate from φ all occurrences of negated binary literals. We will do this in a bottom-up manner.

Take an innermost subformula ψ of φ starting with a maximal block of quantifiers. If $\psi = \exists x\psi_0(x)$ or $\psi = \exists y\psi_0(y)$ then there is nothing to do, as there are no negated binary literals in ψ. Otherwise, ψ has one of the three forms: $\exists y(\alpha(x,y) \wedge \psi_0(x,y))$ or $\exists x(\alpha(x,y) \wedge \psi_0(x,y))$ or $\exists x,y(\alpha(x,y) \wedge \psi_0(x,y))$, where α is one of the four possible guards $x{\downarrow}y, y{\downarrow}x, x{\downarrow_+}y, y{\downarrow_+}x$.

Consider the first form (the other one can be treated similarly). If $\alpha = x{\downarrow}y$ then replace any literal $\neg x{\downarrow}y$ by $\bot$, $\neg y{\downarrow}x$ by $\top$, $\neg x{\downarrow_+}y$ by $\bot$ and $\neg y{\downarrow_+}x$ by $\top$. If $\alpha = y{\downarrow}x$ then proceed analogously. If $\alpha = x{\downarrow_+}y$ then first convert ψ into the equivalent formula $(\exists y(x{\downarrow}y \wedge \psi_0(x,y))) \vee (\exists y(x{\downarrow_+}y \wedge \neg x{\downarrow}y \wedge \psi_0(x,y)))$. With the first disjunct we proceed as described above. The second one is replaced by $\exists z, y(x{\downarrow}z \wedge z{\downarrow_+}y \wedge \psi_0(x,y)$ and then in $\psi_0(x,y)$ we replace $\neg x{\downarrow}y$ by $\top$, $\neg y{\downarrow}x$ by $\top$, $\neg x{\downarrow_+}y$ by $\bot$ and $\neg y{\downarrow_+}x$ by $\top$. If $\alpha = y{\downarrow_+}x$ then we proceed analogously. In all cases we obtain an UNFO replacement of ψ. We then proceed up in the syntax tree of the input formula and finally end up with a UNFO formula equivalent to φ.

From UNFO to CoreXPath. Let φ be a formula in UNFO$[{\downarrow}, {\downarrow_+}]$. Recall that by Lemma 1 we may assume that $\varphi \in \mathrm{U}\mathit{NFO} \cap \mathrm{F}_1$. Here we proceed similarly as in the translation from $\mathrm{F}_1[{\rightarrow}, {\rightarrow^+}]$ to $\mathrm{F}O^2[{\rightarrow}, {\rightarrow^+}]$ in the proof of Thm. 1. Again, the crux is to show how to translate the subformulas of φ starting with a block of quantifiers. Assume that

$$\psi = \exists y_1, \ldots, y_k \psi_0(y_0, y_1, \ldots, y_k) \tag{5.2}$$

is such a subformula. W.l.o.g. we may additionally assume that every subformula of ψ starting with a maximal block of existential quantifiers has a free variable (if it was not the case, we could add a dummy free variable).

Convert ψ_0 into disjunctive form (treating its subformulas starting with a quantifier as atoms) and distribute existential quantifiers over disjunctions, obtaining

$$\psi \equiv \bigvee_{i=1}^{s} \exists y_1 \ldots, y_k \psi_i(y_0, y_1, \ldots, y_k), \tag{5.3}$$

for some $s \in \mathbb{N}$, where each ψ_i is a conjunction of unary literals, binary atoms of the form $y_i{\downarrow}y_j$, $y_i{\downarrow_+}y_j$ or $y_i = y_j$ and subformulas with one free variable of the form $\exists z_1, \ldots, z_l \chi(y_j, z_1, \ldots, z_l)$ or their negations. Note that we do not have negated binary literals.

A *tree-ordering scheme* over variables $y_0, \ldots, y_k$ is a conjunction δ of atoms of the form $y_i{\downarrow}y_j$, $y_i{\downarrow_+}y_j$ or $y_i = y_j$ that can be satisfied in a tree in such a way that this tree satisfies no binary atoms over $y_0 \ldots, y_k$ except those from δ. For example $y_0{\downarrow}y_1 \wedge y_0{\downarrow}y_2 \wedge y_3{\downarrow_+}y_4$ is a tree-ordering scheme, but $y_0{\downarrow_+}y_1 \wedge y_0{\downarrow_+}y_2 \wedge y_1{\downarrow_+}y_3 \wedge y_2{\downarrow_+}y_3 \wedge y_0{\downarrow_+}y_3$ is not since to satisfy it, one needs to add (at least) either $y_2{\downarrow_+}y_1$ or $y_1{\downarrow_+}y_2$.

Consider now a single-disjunct $\exists y_1 \ldots, y_k \psi_i(y_0, y_1, \ldots, y_k)$ of (5.3), and replace it by the following disjunction over all possible tree-ordering schemes δ over $y_0, \ldots, y_k$ containing all the binary atoms of ψ_i, which are not bounded by the quantifiers of the subformulas of ψ_i:

$$\bigvee_{\delta} \exists y_1 \ldots, y_k (\delta(y_0, \ldots, y_k) \wedge \psi_i^{\delta}(y_0, y_1, \ldots, y_k)), \tag{5.4}$$

where ψ_i^{δ} is obtained from ψ_i by removing all the binary atoms (except those bounded by the quantifiers from ψ_i), as they are now present in δ.

Let us now write the conjunction ψ_i^{δ} as $\bigwedge_{j=0}^{k}(\mu_{i,j}^{\delta}(y_j) \wedge \nu_{i,j}^{\delta}(y_j))$, where $\mu_{i,j}^{\delta}(y_j)$ consists of the literals with free variable y_j and $\nu_{i,j}^{\delta}(y_j)$ consists of the subformulas starting with a maximal block of quantifiers with free variable y_j. We now explain how to translate a single-disjunct

$$\exists y_1 \ldots, y_k (\delta(y_0, \ldots, y_k) \wedge \bigwedge_{j=0}^{k} (\mu_{i,j}^{\delta}(y_j) \wedge \nu_{i,j}^{\delta}(y_j))) \tag{5.5}$$

of (5.4).

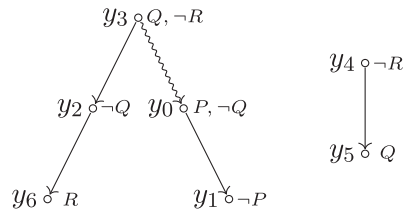

Figure 3 A visualization of a tree-ordering scheme. Straight arrows represent $\downarrow$; wavy arrows represent $\downarrow_+$. Connections implied by transitivity are omitted for clarity.

The idea is to start from y_0 and 'visit the other variables' in the order specified by δ; first go down from y_0 then up, and finally jump to the nodes whose connection to y_0 is not required by δ, at each visited node making all the necessary forks, including those required by the $\nu_{i,j}$. Since making the above intuition formal is cumbersome let us just illustrate it by a representative example.

Let $\delta(y_0, \ldots, y_6) = y_3 \downarrow y_2 \wedge y_3 \downarrow_+ y_0 \wedge y_3 \downarrow_+ y_6 \wedge y_3 \downarrow_+ y_1 \wedge y_0 \downarrow y_1 \wedge y_2 \downarrow y_6 \wedge y_4 \downarrow y_5$. Let $\mu_{i,0} = P(y_0) \wedge \neg Q(y_0)$, $\mu_{i,1} = \neg P(y_1)$, $\mu_{i,2} = \neg Q(y_2)$, $\mu_{i,3} = Q(y_3) \wedge \neg R(y_3)$, $\mu_{i,4} = \neg R(y_4)$, $\mu_{i,5} = Q(y_5)$, $\mu_{i,6} = R(y_6)$, and let $\nu_{i,0} = \exists z_1, z_2 \gamma_0(y_0, z_1, z_2)$, $\nu_{i,2} = \exists z_1 \gamma_2(y_2, z_1)$; the other $\nu_{i,j}$ are empty. See Figure 3. By the inductive assumption we have CoreXPath formulas ν'_0 and ν'_2 equivalent to $\nu_{i,0}$ and, respectively, $\nu_{i,2}$. The translation looks then as follows:

$$
\begin{aligned}
& P \wedge \neg Q \wedge \nu'_0 \\
& \quad \wedge <\downarrow> \neg P \\
& \quad \wedge <\uparrow^+> (Q \wedge \neg R \\
& \qquad \wedge <\downarrow> (\neg Q \wedge \nu_2 \\
& \qquad \wedge <\downarrow> R)) \\
& \quad \wedge <\uparrow^+><\downarrow_+> (\neg R \\
& \qquad \wedge <\downarrow> Q).
\end{aligned}
\tag{5.6}
$$

The first line describes what happens at y_0, the second corresponds to a step down to y_1 and lines 3–5 correspond to a step up to y_3 (from which we go down to y_2 and then once more down to y_6). In lines 6 and 7 we jump to y_4 and then step down to y_5. Note that y_4 and y_5 do not need to be related to the other variables, which is captured by $<\uparrow^+><\downarrow_+>$ that works as the universal modality and allows us to move to any part of the tree.

The correctness of the translation relies on the fact that our formulas do not have literals with negated binary atoms, and thus we do not need to worry about them. Indeed, in our example above one can construct models in which, say, $y_5 \downarrow y_3$, or $y_2 = y_0$ holds.

This gives a procedure translating formulas from UNFO $\cap$ F_1 starting with a block of existential quantifiers into CoreXPath. An arbitrary UNFO $\cap$ F_1 formula with one free variable is a Boolean combination of such formulas (with the same free variable), and its translation is just the Boolean combination of the translations of its constituents. □

5.1 Translation from $F_1[\downarrow, \downarrow_+]$ into $C^2[\downarrow, \downarrow_+]$

In this section we provide the missing translation from the proof of Thm. 7.

Before presenting the details, we comment informally on the intuition behind the proofs. The main issue is that F_1 formulas can be presented in a normal form where every subformula $\exists\bar{x}\psi$ essentially describes a substructure with $|\bar{x}|$ (or $|\bar{x}|+1$) elements. In this normal form, when considering trees only, subformulae $\exists\bar{x}\psi$ describe substructures of trees. Such substructures consist essentially of disjoint subtrees (called components). Such components can be described in C^2 by working inductively from the leaves of the subtrees towards the root. However, with the help of the relation $\downarrow_+$, we can write our F_1 formula in such a form that formulae $\exists\bar{x}\psi$ contain also the unique root of the full tree, and thus the disjoint components are connected (using $\downarrow_+$). Thus, we can write a formula that describes the components and also puts them together in a suitable way.

We then turn to the formal argument. Let σ be an arbitrary finite relational vocabulary. (Note indeed that σ is not necessarily a vocabulary for trees.) We allow σ to contain *nullary* predicates, i.e. Boolean variables; a nullary predicate $Q \in \sigma$ is an atomic formula such that any σ-model $\mathfrak{M}$ interprets Q either such that $\mathfrak{M} \models Q$ or such that $\mathfrak{M} \not\models Q$. A *$\sigma$-diagram* of *width* $k \in \mathbb{Z}_+$ is a quantifier-free conjunction consisting of the following:[2]

1. A conjunction expressing that the variables $x_1, ..., x_k$ are mutually pairwise distinct.
2. A conjunction containing *exactly one* of the literals $R\bar{x}, \neg R\bar{x}$ for each $R \in \sigma$ and each $\bar{x} \in \{x_1, \dots, x_k\}^{ar(R)}$, where $ar(R)$ is the arity of R.

A quantifier-free formula is a diagram if it is a σ-diagram of width k for some σ and some k.

Let $\exists\bar{x}\varphi$ be a formula of F_1. The formula φ is a Boolean combination of atoms and existential formulae $\exists\bar{y}\psi$. We call such existential formulae $\exists\bar{y}\psi$ *relative atoms* of φ. The other atoms are called *free atoms* of φ. For example, the formula $R(x,y) \wedge \exists x(S(y,x) \wedge \psi'(y))$ contains a binary free atom $R(x,y)$ and a unary relative atom $\exists x(S(y,x) \wedge \psi'(y))$.

A formula χ of F_1 is said to be in *diagram normal form* if every subformula $\exists\bar{x}\varphi$ of χ has the property that φ is a diagram with respect to the vocabulary σ' defined as follows:

1. σ' contains the predicate symbols of the free atoms of φ.
2. If $\psi(x)$ is a unary relative atom of φ (i.e. a relative atom with one free variables, x), then this relative atom is considered a unary predicate in σ'.
3. Similarly, if ψ' is a nullary relative atom of φ, then this relative atom is considered a nullary predicate in σ'.

LEMMA 10
Formulae of F_1 have equivalent representations in diagram normal form.

PROOF. Consider an F_1-formula $\exists\bar{x}\varphi$. Now, the formula φ can be modified into a disjunctive normal form formula φ_{DNF} (without modifying its atoms or relative atoms). Then the quantifier prefix $\exists\bar{x}$ can be distributed over the disjunctions of φ_{DNF}. The resulting formula is of the form $(\exists\bar{x}\varphi_1)\vee...\vee(\exists\bar{x}\varphi_k)$, where the formulae φ_i are conjunctions of (possibly negated) free atoms and (possibly negated) relative atoms of φ.

Consider one such disjunct $\exists\bar{x}\varphi_i$. We first apply the induction hypothesis to the free atoms and relative atoms of φ_i and put them in diagram normal form. Let the obtained formula be denoted by φ_i'. We next describe how the formula $\exists\bar{x}\varphi_i'$ can be put to diagram normal form. The intuitive idea is roughly to (1) consider all distributions of equalities and inequalities for the free variables in φ_i'; (2)

[2] We note that diagrams of width k are similar to k-types.

for each such distribution, define all diagrams consistent with φ_i; and (3) take the disjunction over all the diagrams and distribute the existential quantifiers of $\exists\bar{x}$ over the disjunction.

The general formal construction is straightforward but cumbersome, so we begin by sketching some examples. First consider the case where φ_i' is $R(x,y)$.

- The formula $R(x,y)$ gives rise to, e.g. the diagram $x \neq y \wedge R(x,y) \wedge \neg R(y,x) \wedge \neg R(x,x) \wedge R(y,y)$.
- Furthermore, the same formula $R(x,y)$ gives rise to the diagram $R(x,x)$ obtained under the distribution of equalities, which requires that $x = y$. We note that $R(x,x)$ loses the free variable y, which is an undesirable effect. Therefore, we shall use the formula $x = y \wedge R(x,x)$ instead. This in not strictly speaking a diagram, but we shall discuss below how to deal with this problem.

To obtain the diagram normal form formula equivalent to $\exists\bar{x}\varphi_i'$ in this particular case where φ_i' is $R(x,y)$, we take a disjunction of all the diagrams consistent with $R(x,y)$ and also the formulas that are not strictly speaking diagrams, such as $x = y \wedge R(x,x)$, and distribute the existential quantifiers $\exists\bar{x}$ inwards over the disjunctions. To put the non-diagram conjuncts such as $\chi := \exists\bar{x}(x = y \wedge R(x,x))$ into diagram normal form, we consider three cases. In each case we replace χ by an equivalent formula in diagram normal form. Firstly, if $\exists\bar{x} = \exists x$, then we replace χ by $R(y,y)$. Secondly, if $\exists\bar{x} = \exists y$, we replace χ by $R(x,x)$. Finally, if $\exists\bar{x} = \exists x\exists y$, then we replace χ with $\exists x R(x,x)$. These new formulae are in diagram normal form. All remaining cases are similar.

The general way to deal with any formula $\exists\bar{x}\varphi_i'$ is as follows. Recall that φ_i' is a conjunction of formulas in diagram normal form. We may assume, w.l.o.g., that φ_i' is quantifier-free (because the relative atoms will be treated as if they were atoms). If the quantifier-free formula φ_i' is inconsistent, then we simply take an arbitrary formula ψ in diagram normal form and replace $\exists\bar{x}\varphi_i'$ with $\psi \wedge \neg\psi$. Assuming φ_i' is consistent, we perform the following steps:

1. Let X denote the set of (free) variables in the conjunction φ_i', and let τ be the vocabulary of φ_i'. By a *proto diagram over* φ_i' we mean a conjunction $\delta \wedge \varphi_i''$ such that the following conditions hold:
   - δ is a conjunction of equality atoms and negated equality atoms in the variables X that contains, for any two distinct variables x and y, either the conjunct $x = y$ or $x \neq y$. Furthermore, δ is consistent with the conjunction φ_i' and therefore contains all equalities and inequalities that already occur in φ_i'.
   - By a *relational τ-literal over X* we mean a atom or negated atom (of vocabulary τ and with its variables from X) that is not an equality or negated equality atom. Now, φ_i'' is a maximal set of relational τ-literals over X that is consistent with φ_i'.

   We define χ' to be the disjunction of all proto diagrams $\delta \wedge \varphi_i''$ over φ_i'.
2. We distribute the existential quantifiers $\exists\bar{x}$ over the disjuncts of χ'. Consider one such disjunct $\exists\bar{x}(\delta \wedge \varphi_i'')$. Eliminate positive equalities from $\delta \wedge \varphi_i''$ one-by-one by modifying $\delta \wedge \varphi_i''$ such that for each positive equality (e.g. $x = y$), choose one of the variables (e.g. x) and replace each instance of the chosen variable in φ_i'' by the other variable (e.g. replace instances of x by y). In the process, do not eliminate the possible free variable of $\exists\bar{x}(\delta \wedge \varphi_i'')$ (note that there is at most one such possible free variable since we are dealing with F_1). This way we get rid of positive equalities, and the obtained formula is in diagram normal form.

This process converts the original formula φ into diagram normal form. □

Above we considered general F_1, but now we return to considering $\mathrm{F}_1[\downarrow, \downarrow_+]$ over trees (labelled with extra unary predicates).

A formula of $\mathrm{F}_1[\downarrow, \downarrow_+]$ is in *rooted diagram form* if every diagram contains a point z and an extra conjunct $\gamma(z)$ stating that z is the root of the tree; thus, $\gamma(z)$ states that z is an element that does not have any parent, $\gamma(z) := \neg\exists y(y\downarrow z)$. It is easy to modify diagram normal form formulae into rooted diagram form by taking big disjunctions of all possibilities for the location of z in a diagram: potentially any variable of the original diagram can be the root, and additionally, it may be that none of the variables is the root and thus a root must be added. We take a disjunction of all the possible configurations that arise.

Note that in rooted diagram form, every diagram is connected, meaning that for all distinct variables x, y, there is an undirected path using the relations $\downarrow$, $\downarrow_+$ that begins from x and ends with y. This connectedness property will help us with the inductive argument in the proof of the following lemma.

LEMMA 11
Let $\varphi(x) = \exists\bar{y}\psi$ be a formula of $\mathrm{F}_1[\downarrow, \downarrow_+]$ where x a free variable. Then there exists a formula $\varphi^*(x)$ of $\mathrm{C}^2[\downarrow, \downarrow_+]$ that is equivalent to $\varphi(x)$

PROOF. We first note that if $\varphi(x)$ is not satisfiable over trees, the desired C^2 formula is $\bot$. Thus, we assume that $\varphi(x)$ is satisfiable over trees. We let σ denote the vocabulary of $\varphi(x)$ and σ_0 the proposition symbols that occur in φ, i.e. the unary relation symbols in φ.

As argued above, we may assume that $\varphi(x)$ is in rooted diagram normal form. Therefore, ψ is a diagram that describes a substructure M_ψ of a tree; M_ψ contains a root node and is connected. The nodes of M_ψ are the variables of ψ.

To express $\varphi(x) = \exists\bar{x}\psi$ in C^2, we first define, for each leaf u of M_ψ, the unique σ_0-diagram of width 1 that is satisfied by u and denote this formula by $\delta_u(y)$. The idea is then to proceed inductively upwards from the leaves and define, for each point v in the diagram M_ψ, a formula that characterizes—in a way specified later on—the substructure of M_ψ below v in the tree. However, since the free variable x of $\varphi(x)$ is not necessarily the root of M_ψ, and since we want the induction to end at x, we will in fact proceed such that from the perspective of the *undirected* tree induced by M_ψ, we work from the leaves towards x. Since x is a legitimate root for the undirected tree, the induction ends at x.

We illustrate the flow of the induction by an example first. Assume the diagram M_ψ consists of the seven points r, s, s', w, x, x', y such that M_ψ satisfies the following conditions (see Figure 4 for an illustration):

1. The the child relation $\downarrow$ in the diagram M_ψ is $\{(w,x), (w,x'), (x,y), (s,s')\}$.
2. r connects via $\downarrow_+$ to s and w (and thus to x, x', y, s' as well).
3. If $u \in \{s, s'\}$ and $t \in \{w, x, x', y\}$, then $u \not\downarrow_+ t$ and $t \not\downarrow_+ u$.

In this case we first deal with the leaves s', x' and y and define suitable C^2 formulae for them. The exact properties these formulae are to satisfy, will be specified below. Next the induction takes care of s (the parent of s') using the formula already specified for s'. Thus, we obtain a C^2 formula for s. Then, using the formula for s, we define a C^2 formula for r. After this, we define a C^2 formula for the node w using the formulae defined for r and x'; notice that this time (when going from r to w) the relation $\downarrow_+$ is scanned in a different direction as in the previous step when going from s to r. Thus, we have now defined the formulae for each of the neighbours of x in the diagram M_ψ, *i.e.*, the formulae for w and y. Therefore, we can finally define the formula for x (scanning the relation $\downarrow$ both ways, upwards from y and downwards from w). Notice that since $\varphi(x)$ is in rooted diagram normal form, M_ψ is connected, which in the this example is due to the node r. Thus, the induction does not have to deal with any disconnected components.

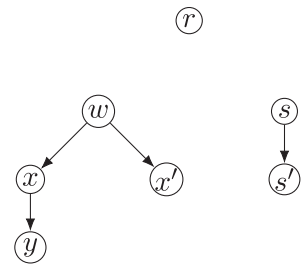

Figure 4 The seven-point diagram with the root node r. The relation $\downarrow_+$ has not been drawn for the sake of clarity.

If u is a point in the diagram M_ψ, then we let $M_\psi(u)$ denote the substructure of M_ψ induced by the set that contains u and all the nodes occurring before u in the induction from the leaves towards x. Formally, the domain of $M_\psi(u)$ consists of the points $u' \in M_\psi$ such that the undirected path in M_ψ from x to u' contains u. We next show how to inductively define, for each node $v \in M_\psi$, a formula $\chi_v(y)$ of C^2 (with the sole free variable y) such that the following condition holds.
Let N be a σ-tree and v' a point in N. Then $N \models \chi_v(v')$ iff N contains an induced substructure isomorphic to $M_\psi(v)$ via an isomorphism that sends v' to v.

Let $v \neq x$ be a leaf of M_ψ. Then we let $\chi_v(y)$ be the formula $\delta_v(y)$ (defined above). Now assume v is not a leaf of M_ψ. In order to show how to define $\chi_v(y)$, we first give some auxiliary definitions.

1. Let $c_1, \ldots, c_{n_+}$ denote the children of v in $M_\psi(v)$, i.e. the elements c_i in $M_\psi(v)$ such that $(v, c_i) \in \downarrow$. (We note that possibly $n_+ = 0$.)
2. Let c' denote a possible parent of v in $M_\psi(v)$, i.e. a point c' in $M_\psi(v)$ such that $(c', v) \in \downarrow$. (We note that possibly there exists no such point c'.)
3. Let $d_1, \ldots, d_{m_+}$ denote the descendants d_i of v in $M_\psi(v)$ such that
   a. $(v, d_i) \in \downarrow_+ \setminus \downarrow$, *i.e.*, d_i is a descendant of v but not a child of v,
   b. there exists no point d in the diagram M_ψ such that $v \downarrow_+ d \downarrow_+ d_i$.
   We note that possibly $m_+ = 0$.
4. Let d' denote the (possibly non-existing) ancestor of v in $M_\psi(v)$ such that d' is *not* a parent of v and there does not exist a point d in the diagram M_ψ such that $d' \downarrow_+ d \downarrow_+ v$. Note that if we have the point c' (specified in bullet 2 above) in the diagram, then there is no d' in the diagram. Vice versa, if there is a d' in the diagram, there is no c'. Also, it is possible that neither c' nor d' exists in the diagram.

By the induction hypothesis, we have defined all the required formulae denoted by

$$\chi_{c_1}, \ldots, \chi_{c_{n_+}}, \chi_{c'}, \chi_{d_1}, \ldots, \chi_{d_{m_+}}, \chi_{d'}$$

each with the free variable y. (We note that $\chi_{c'}$ and $\chi_{d'}$ cannot not *both* be in the list of required formulae.) Now, we shall next consider *collections* of these formulae defined as follows. A collection (in the variable y) is here defined to be a conjunction that contains, for each of the formulae $\chi(y)$ defined in the previous stage of the induction (i.e. some subset of the above listed formulae $\chi_{c_1}, \ldots, \chi_{c_{n_+}}, \chi_{c'}, \chi_{d_1}, \ldots, \chi_{d_{m_+}}, \chi_{d'}$), exactly one of the formulae $\chi(y), \neg\chi(y)$ as a conjunct. (We note that a collection may be unsatisfiable if for example χ_{c_1} and χ_{c_2} are equivalent and we include the negation of exactly one of them in the collection. Note also that a node satisfying some formula

$\chi_{v_1}(y)$ can also satisfy a non-equivalent formula $\chi_{v_2}(y)$ as the formulae are only supposed to assert that a certain substructure exists 'below' y in the undirected tree with root x.)

Our next step is to show how $\chi_v(y)$ can be defined based on the corresponding formulae $\chi_{c_1}, ..., \chi_{c_{n_+}}, \chi_{c'}, \chi_{d_1}, ..., \chi_{d_{m_+}}, \chi_{d'}$ for the nodes that occur immediately before v in the induction. We illustrate how this is done via an elucidating example. Consider a situation where $n_+ = 2$, $m_+ = 1$ and neither c' nor d' exists.[3] The formula $\chi_v(y)$ will be a big disjunction over all possible suitable scenarios that the nodes in the previous stage could realize in some (any) tree with the desired substructure. One possible scenario in some structure is the one where one child of the current node v satisfies $\chi_{c_1}(y) \wedge \chi_{c_2}(y)$, another child satisfies $\neg\chi_{c_1}(y) \wedge \chi_{c_2}(y)$, and a third child satisfies $\chi_{c_1}(y) \wedge \neg\chi_{c_2}(y) \wedge \exists x(\downarrow_+(y,x) \wedge \chi_{d_1}(x))$. In this scenario there must exist two distinct children such that one satisfies χ_{c_1} and the other one χ_{c_2}, and furthermore, there must be a third child that connects via $\downarrow_+$ to a node satisfying χ_{d_1}. This kind of a condition is easily expressible in C^2. Another suitable scenario is that three or more children all satisfy the formula

$$\alpha(y) := \chi_{c_1}(y) \wedge \chi_{c_2}(y) \wedge \exists x(\downarrow_+(y,x) \wedge \chi_{d_1}(x)),$$

as this ensures that there is a child u_1 that satisfies $\chi_{c_1}(y)$ and another child u_2 that satisfies $\chi_{c_2}(y)$, and furthermore, a descendant (which is neither a child nor reachable via u_1 or u_2) that satisfies χ_{d_1}. Here all the three children satisfy the same collection $\chi_{c_1}(y) \wedge \chi_{c_2}(y)$. Again it is easy to describe this scenario in C^2 by stating the existence of at least three children satisfying $\alpha(y)$. It is not difficult to see how to write the full disjunction $\chi_v(y)$ that covers all the possible suitable scenarios and thereby enumerates the ways to connect v to the nodes in the previous stage.

We consider one more example. This time we assume that $m_+ = n_+ = 0$ and that c' exists. Furthermore, we assume that c' has two children in the diagram, v and u, and the induction proceeds from u via c' to v. Now note that we cannot simply define $\chi_v(y)$ to be the formula $\exists x(x{\downarrow}y \wedge \chi_{c'}(x))$, but instead, we must define $\chi_v(y)$ to be the formula

$$(\neg\chi_u(y) \wedge \exists x(x{\downarrow}y \wedge \chi_{c'}(x))) \vee (\chi_u(y) \wedge \exists x(x{\downarrow}y \wedge \chi_{c'}(x) \wedge \exists^{\geq 2} y(x{\downarrow}y \wedge \chi_u(y)))).$$

We omit further details since it is now easy to see in general how to write the formulae $\chi_v(y)$ in C^2 using collections and *counting*; one simply enumerates all possible situations with sufficient numbers of correctly oriented neighbours satisfying the formulae $\chi_{c_1}, ..., \chi_{c_{m_+}}, \chi_{c'}, \chi_{d_1}, ..., \chi_{d_{n_+}}, \chi_{d'}$. Then the big disjunction over all the resulting possible ways to connect v to the nodes in the previous stage is the desired formula. □

COROLLARY 2

Let $\varphi = \exists\bar{y}\psi$ be a formula of $F_1[\downarrow, \downarrow_+]$. The formula φ may be a sentence or contain a single free variable. Then there exists a formula φ^* of $C^2[\downarrow, \downarrow_+]$ that is equivalent to φ.

PROOF. The case where φ has a single free variable follows from the previous lemma. The case where φ is a sentence is covered as follows.

Assume $\varphi = \exists\bar{x}\psi$. Remove a single variable from $\bar{x}$ and apply the argument for the case with a free variable. Then reintroduce a quantifier that quantifies the remaining free variable away. □

[3] For simplicity, we assume here and in all subsequent examples that the set of unary predicates considered is empty.

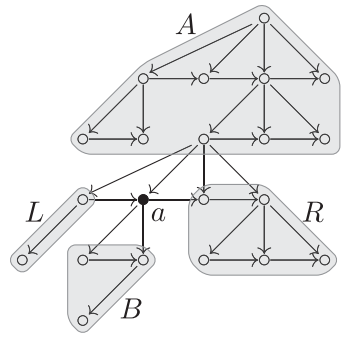

FIGURE 5 Positions in a tree with respect to a node a.

6 Satisfiability of one-dimensional fragment over trees

The aim of this section is to establish the complexity of satisfiability of F_1 over trees for all navigational signatures contained in $\{\downarrow, \downarrow_+, \rightarrow, \rightarrow^+\}$. En route we show some small model properties, allowing us, when designing algorithms deciding satisfiability, to restrict attention to models with appropriately bounded vertical and horizontal paths.

Actually, we can show that when the child relation is present in the signature then the satisfiability problem is 2-EXPTIME-complete using some known results on UNFO.

THEOREM 8
Let $\{\downarrow\} \subseteq \sigma_{nav} \subseteq \{\downarrow, \downarrow_+, \rightarrow, \rightarrow^+\}$. Then the satisfiability problem for $\mathrm{F}_1[\sigma_{nav}]$ is 2-EXPTIME-complete.

PROOF. In [27] a 2-EXPTIME-upper bound is given for UNFO$[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$, and the corresponding lower bound—for UNFO$[\downarrow]$. Using the polynomial translation from $\mathrm{F}_1[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$ to UNFO$[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$ in the proof of Thm. 6 and, respectively, Lemma 1 we can transfer these bounds to F_1. □

We will soon see that for the navigational signatures not containing $\downarrow$ but containing $\downarrow_+$ the complexity drops down to EXPSPACE. The machinery we will develop will also allow us to give an alternative, direct proof of the upper bound in Thm. 8

6.1 Profiles for trees

We begin with an adaptation of the notion of profiles for the case of trees. As in the case of words, the *profile* of a node says what the types of all tuples (of some bounded size) containing this node are.

Given a tree $\mathfrak{T}$ and its nodes a, b, $b \neq a$, we say that b is *in position B* to a (or *below a*) if it belongs to the subtree of a, *in position L* to a (or *left* to a) if it belongs to the subtree of some left sibling of a, *in position R* to a (or *right* to a) if it belongs to the subtree of some right sibling of a, and *in position A* to a (or *above a*) if it is not in any of the previous positions to a. See Figure 5.

Let $\sigma = \sigma_0 \cup \sigma_{nav}$ be a signature such that $\rightarrow^+ \in \sigma_{nav}$[4] and let $\mathfrak{T}$ be a tree over this signature. We say that an element $a \in T$ *realizes* (or *has*) a *k-σ-profile* σ-$prof^{\mathfrak{T}}_k(a) = (\mathcal{F}, \mathcal{A}, \mathcal{B}, \mathcal{L}, \mathcal{R})$ if $\mathcal{F}$ is

[4]This assumption is of technical character, and our approach could be also developed for signatures not containing $\rightarrow^+$.

the set of all s-types, $1 \leq s \leq k$, realized by tuples $a_1, \ldots, a_s$ such that $a_1 = a$, and for any position $P \in \{A, B, L, R\}$, the component $\mathcal{P}$ is the subset of $\mathcal{F}$ consisting of the types realized by those tuples for which for all $2 \leq i \leq k$ the element a_i is in position P to a. When σ is clear from context we will just speak about k-profiles and write $prof_k^{\mathfrak{T}}(a)$ instead of k-σ-profiles and σ-$prof_k^{\mathfrak{T}}(a)$. Given a σ-k-profile θ we will sometimes refer to its components with $\theta.\mathcal{F}$, $\theta.\mathcal{A}$, $\theta.\mathcal{B}$, $\theta.\mathcal{L}$ and $\theta.\mathcal{R}$.

LEMMA 12
Let θ be a profile of a node in a tree over a signature containing $\rightarrow^+$. Then $\theta.\mathcal{F}$ is unequivocally determined by $\theta.\mathcal{A}$, $\theta.\mathcal{B}$, $\theta.\mathcal{L}$ and $\theta.\mathcal{R}$ Moreover, there is a procedure $\mathtt{f}ulltype(\mathcal{A}, \mathcal{B}, \mathcal{L}, \mathcal{R})$ which given $\theta.\mathcal{A}$, $\theta.\mathcal{L}$, $\theta.\mathcal{B}$ and $\theta.\mathcal{R}$ computes $\theta.\mathcal{F}$ in time polynomial in $|\theta|$ and exponential in k.

PROOF. The unique determination of $\mathcal{F}$ for an element a follows from the fact that for any elements b_1, b_2, if we know their positions to a and the truth values of the atoms $a \rightleftharpoons b_1$, $b_1 \rightleftharpoons a$, $a \rightleftharpoons b_2$, $b_2 \rightleftharpoons a$ for all $\rightleftharpoons \in \sigma_{nav}$ then the truth values of $b_1 \rightleftharpoons b_2$ and $b_2 \rightleftharpoons b_1$ are determined for all $\rightleftharpoons$.[5]

More specifically, to construct all types in $\mathcal{F}$ we proceed as follows. Construct all possible tuples consisting of at most one type from each of the components $\theta.\mathcal{A}$, $\theta.\mathcal{B}$, $\theta.\mathcal{L}$ and $\theta.\mathcal{R}$ and for each such tuple combine its types together into a single type in a natural way, that is identify their x_1 variables, appropriately renumber the other variables, and appropriately set the navigational relations. The latter is done using the observation that the only navigational connections between elements in different positions to an element a are as follows:

- if b is in position B (to a) then $c{\downarrow_+}b$ holds for those c in position A for which $c{\downarrow_+}a$ holds;
- if b is in position L then $c{\downarrow_+}b$ holds for those c in position A for which $c{\downarrow_+}a$ holds, $c{\downarrow}b$ holds if $c{\downarrow}a$ holds, and $b{\rightarrow^+}d$ holds for those d in position R for which $a{\rightarrow^+}d$ holds;
- symmetrically for b in position R.

If the number l of variables in the so obtained type π is not greater than k then the procedure adds to $\mathcal{F}$ the type π, together with all the types obtained from π by permuting its variables $x_2, \ldots, x_l$. □

For example let us consider the signature σ with $\sigma_0 = \{P\}$ and $\sigma_{nav} = \{\downarrow_+, \rightarrow^+\}$ and assume that the 2-type $\pi_1 = \{Px_1, Px_2, x_2{\downarrow_+}x_1\} \in \mathcal{A}$, and the 3-type $\pi_2 = \{Px_1, Px_3, x_3{\downarrow_+}x_2\} \in \mathcal{R}$ (we list only non-negated literals in the types). The combination of π_1 and π_2 is the 4-type $\pi_3 = \{Px_1, Px_2, Px_4, x_2{\downarrow_+}x_1, x_2{\downarrow_+}x_3, x_2{\downarrow_+}x_4, x_4{\downarrow_+}x_3\}$. See Fig. 6. The type π_3 together with the types obtained from π_3 by permuting the variables x_2, x_3, x_4 in all possible ways are added to $\mathcal{F}$ if $k \geq 4$.

LEMMA 13
The number of k-σ-profiles is bounded from above by $\mathfrak{g}^\star(|\sigma_0|, k)$ where $\mathfrak{g}^\star : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ is a fixed function, doubly exponential in its both arguments.

PROOF. Each component of a profile is determined by the set of the k-types it contains. The number of k-types in a component can be roughly estimated by $(2^{|\sigma_0|})^k \cdot 9k(k-1)$ (the number of possible assignments of 1-types to the elements of a tuple of k elements, times the number of possible connections by relations from σ_{nav} for a pair of elements a, b: a is equal to b, b is the next sibling of a, b is a following sibling of a but not the next one, b is a child of a, b is a descendant of a but not

[5] This could be not true if $\rightarrow^+ \notin \sigma_{nav}$: assuming that b_1 is the parent of a and b_2 is in position R to a but not joined to it by any relation then we do not know if b_1 is the parent of b_2 or just an ancestor.

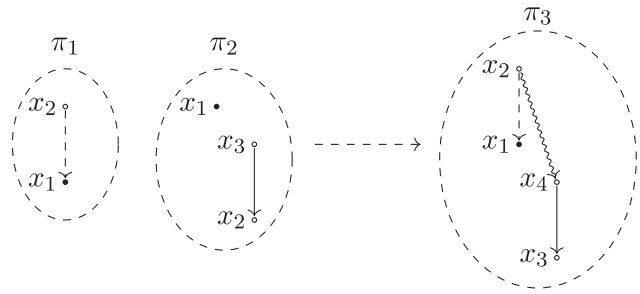

FIGURE 6 Combining type π_1 and π_2 into a single type π_3. The dashed and solid connection in π_3 are present in π_1 and π_2, respectively; the wavy connection follows from the definition of profiles.

its child; and vice versa). So, each component has at most $2^{(2^{|\sigma_0|})^k \cdot 9k(k-1)}$ possible values. From this the existence of the desired $\mathfrak{g}^\star$ follows. □

In contrast to the case of words, where in a single model at most exponentially many profiles are realized (Lemma 4), in the case of trees there are F_1 formulas whose models must realize doubly exponentially many profiles. In the next section we will however see, that for some signatures we can at least bound exponentially the number of profiles on all the vertical or horizontal paths in 'minimal' models of any formula, which will allow us to prove our small model properties.

Again, we connect the notion of profiles with satisfaction of normal form formulas. Given a normal form formula φ of width k over a signature σ we say that a k-σ-profile θ is *compatible* with φ if

- for every conjunct $\forall x_1 \dots x_{l_i} \varphi_i^\forall(x_1 \dots x_{l_i})$ of φ and every l_i-type $\pi \in \theta.\mathcal{F}$, we have $\pi \models \varphi_i^\forall$;
- for every conjunct $\forall y_0 \exists y_1 \dots y_{k_i} \varphi_i^\exists(y_0, y_1 \dots y_{k_i})$ of φ there is a (k_i+1)-type $\pi \in \theta.\mathcal{F}$ such that $\pi \models \varphi_i^\exists(x_1, \dots, x_{k_i+1})$.

It is straightforward to see the following:

LEMMA 14
A normal form formula φ of width k over a signature σ is satisfied in a tree $\mathfrak{T}$ iff every k-σ-profile realized in $\mathfrak{T}$ is compatible with φ.

For further use we make the following observation:

LEMMA 15
Let $\mathfrak{T}$ be a σ-tree, a, a' its nodes and θ, θ' their respective k-σ-profiles.

(i) If a' is a child of a, then
  a. $\theta'.\mathcal{A}$ is uniquely determined by the 1-type of a and $\theta.\mathcal{L}$, $\theta.\mathcal{R}$ and $\theta.\mathcal{A}$
  b. $\theta.\mathcal{B}$ is uniquely determined by the 1-type of a' and $\theta'.\mathcal{L}$, $\theta'.\mathcal{R}$ and $\theta.\mathcal{B}$

(ii) If a' is the next sibling of a, then
  a. $\theta'.\mathcal{L}$ is uniquely determined by the 1-type of a and $\theta.\mathcal{L}$ and $\theta.\mathcal{B}$.
  b. $\theta.\mathcal{R}$ is uniquely determined by the 1-type of a' and $\theta'.\mathcal{R}$, $\theta'.\mathcal{B}$.

Moreover, there is a procedure $\texttt{computeA}(\mu, \mathcal{L}, \mathcal{A}, \mathcal{R})$ that computes the component $\theta'.\mathcal{A}$ when given the 1-type of a and $\theta.\mathcal{L}$, $\theta.\mathcal{R}$ and $\theta.\mathcal{A}$, in time polynomial in $|\theta|$ and exponential in k. Analogously there are procedures $\texttt{computeB}(\mu, \mathcal{L}, \mathcal{R}, \mathcal{B})$, $\texttt{computeL}(\mu, \mathcal{L}, \mathcal{B})$ and

$\texttt{computeR}(\mu, \mathcal{R}, \mathcal{B})$ computing $\theta.\mathcal{B}$, $\theta'.\mathcal{L}$, $\theta.\mathcal{R}$, respectively, when fed with the appropriate parameters (as in points (i)(b), (ii)(a), (ii)(b)).

PROOF. Consider the statement (i)(a). Note that the set of nodes in position A to a' consists precisely of a and the elements in positions L, R and A to a. Thus, the procedure $\texttt{computeA}$ can work as follows.

Construct all possible tuples consisting of at most one type from each of the components $\theta.\mathcal{L}, \theta.\mathcal{R}$ and $\theta.\mathcal{A}$ and for each such tuple combine its types together into a single type π in a natural way, that is identify their x_1 variables, appropriately renumber the other variables, and appropriately set the navigational relations (similarly as it was done in the proof of Lemma 12). Construct π' by increasing the number of every variable in π by 1 and adding x_1 as 'a child of 1-type μ of (the current) x_2' (that is by setting the truth of σ_0-atoms containing x_1 in accordance with μ and appropriately setting the truth of σ_{nav}-atoms containing x_1 and the other x_i). Then construct π'' from π' by removing all the literals that contain x_2 and then decreasing the number of each variable x_i, $i \geq 2$ by 1. If the number l of variables in the so obtained type π' (π'') is not greater than k then add to $\mathcal{F}$ the type π' (π''), together with all the types obtained from π' (π'') by permuting its variables $x_2, \ldots, x_l$.

The other statements can be justified analogously. □

We say that an s-type is *trivial* if for every $1 \leq i,j \leq s$ it contains $x_i = x_j$, that is it is realized only by singletons. We say that a trivial s-type is *based on* 1-*type* μ if its restriction to x_1 is equal to μ. We now define the following notion of local consistency.

DEFINITION 1
Let $\mathfrak{T}$ be a tree, k a natural number, Ω a function assigning to each $a \in T$ a 1-type and Ξ a function assigning to each $a \in T$ a tuple $(\mathcal{F}, \mathcal{A}, \mathcal{B}, \mathcal{L}, \mathcal{R})$ of collections of s-types such that $\mathcal{F} = \texttt{fulltype}(\mathcal{A}, \mathcal{B}, \mathcal{L}, \mathcal{R})$, $s \leq k$. We say that the pair (Ω, Ξ) is *locally consistent* on $\mathfrak{T}$ if the following conditions hold:

(a) if $a \in T$ is the root then $\Xi(a).\mathcal{A}$ is trivial and based on $\Omega(a)$;
(b) if $a \in T$ is a leaf then $\Xi(a).\mathcal{B}$ is trivial and based on $\Omega(a)$;
(c) if $a \in T$ has no preceding sibling then $\Xi(a).\mathcal{L}$ is trivial and based on $\Omega(a)$;
(d) if $a \in T$ has no following sibling then $\Xi(a).\mathcal{R}$ is trivial and based on $\Omega(a)$;
(e) for any $a, a' \in T$ such that a' is a child of a we have
  (i) $\Xi(a').\mathcal{A} = \texttt{computeA}(\Omega(a'), \Xi(a).\mathcal{L}, \Xi(a).\mathcal{R}, \Xi(a).\mathcal{A})$
  (ii) $\Xi(a).\mathcal{B} = \texttt{computeB}(\Omega(a), \Xi(a').\mathcal{L}, \Xi(a').\mathcal{R}, \Xi(a').\mathcal{B})$
(f) for any $a, a' \in T$ such that a' is the next sibling of a we have
  (i) $\Xi(a').\mathcal{L} = \texttt{computeL}(\Omega(a'), \Xi(a).\mathcal{L}, \Xi(a).\mathcal{B})$
  (ii) $\Xi(a).\mathcal{R} = \texttt{computeR}(\Omega(a), \Xi(a).\mathcal{R}, \Xi(a).\mathcal{B})$

Obviously, if Ω returns the 1-types of elements of $\mathfrak{T}$ and Ξ returns their k-profiles then the pair (Ω, Ξ) is locally consistent. In the following lemma we show that the opposite is also true.

LEMMA 16
Let $\mathfrak{T}$ be a tree, k a natural number, Ω the function assigning to each $a \in T$ the 1-type of a in $\mathfrak{T}$ and Ξ a function assigning to each $a \in T$ a tuple $(\mathcal{F}, \mathcal{A}, \mathcal{B}, \mathcal{L}, \mathcal{R})$ of collections of s-types, $1 \leq s \leq k$. If the pair (Ω, Ξ) is locally consistent on $\mathfrak{T}$ then for every $a \in T$ we have that $\Xi(a) = prof_k^{\mathfrak{T}}(a)$.

PROOF. Let us first see that for every $a \in T$ the equality holds for the $\mathcal{L}$-, $\mathcal{B}$- and $\mathcal{R}$-components of $\Xi(a)$ and $prof_k^{\mathfrak{T}}(a)$. Let d be the maximal number of edges on a vertical path in $\mathfrak{T}$. Define $level(a)$ to

be d if a is the root and $level(b)-1$, where b is the parent of a, otherwise. This way $0 \leq level(a) \leq d$ for any $a \in T$. Define $posl(a)$ to be 0 is a the leftmost child of some node and $posl(b)+1$, where b is the previous sibling of a, otherwise. Similarly, define $posr(a)$ to be 0 is a the rightmost child of some node and $posr(b)+1$, where b is the next sibling of a, otherwise.

We proceed by induction on the level of a node. For the base of induction assume $level(a) = 0$. In this case a is a leaf. Then the equality $\Xi(a).\mathcal{B} = prof_k^{\mathfrak{T}}(a).\mathcal{B}$ follows from Condition (b) of Def. 1. Consider now the $\mathcal{L}$-components. We proceed by subinduction on $posl(a)$. If $pos(a) = 0$ then the equality $\Xi(a).\mathcal{L} = prof_k^{\mathfrak{T}}(a).\mathcal{L}$ follows from Condition (c). Otherwise, assume that for the previous sibling a' of a we have $\Xi(b).\mathcal{L} = prof_k^{\mathfrak{T}}(a').\mathcal{L}$. As it must be that $level(a') = 0$ it again follows from Condition (b) that $\Xi(a').\mathcal{B} = prof_k^{\mathfrak{T}}(a').\mathcal{B}$. The equality $\Xi(a).\mathcal{L} = prof_k^{\mathfrak{T}}(a).\mathcal{L}$ follows now from Condition (f)(i). The argument for the $\mathcal{R}$-components is strictly symmetric, by subinduction on $posr(a)$ and involves Condition (f)(ii).

Assume now that $level(a) = s$, for some $s > 0$ and that for any node b with $level(b) = s-1$ we have $\Xi(b).\mathcal{B} = prof_k^{\mathfrak{T}}(b).\mathcal{B}$, $\Xi(b).\mathcal{L} = prof_k^{\mathfrak{T}}(b).\mathcal{L}$, $\Xi(b).\mathcal{R} = prof_k^{\mathfrak{T}}(b).\mathcal{R}$ (the main inductive assumption). This inductive assumption in particular holds for any child of a. Again, we first consider the $\mathcal{B}$-components. If a is a leaf then the equality $\Xi(a).\mathcal{B} = prof_k^{\mathfrak{T}}(a).\mathcal{B}$ follows from Condition (b). Otherwise let a' be a child of a. The equality for the $\mathcal{B}$-components for a follows in this case from the inductive assumption for a' and Condition (e)(ii). So, all the nodes on level s have proper $\mathcal{B}$-components. For the $\mathcal{L}$- and $\mathcal{R}$-components we can now proceed as in the base of induction.

This finishes the part of the proof concerning the $\mathcal{B}$-, $\mathcal{L}$- and $\mathcal{R}$-components. It remains to show the equality for the $\mathcal{A}$-components. This is done by induction on $depth(a)$. If $depth(a) = 0$ (a is the root) then the equality for the $\mathcal{A}$-components follows from (a). Otherwise, let a' be the parent of a and assume that the equality $\Xi(a').\mathcal{A} = prof_k^{\mathfrak{T}}(a').\mathcal{A}$ holds. As we have already proved, for all nodes of $\mathfrak{T}$ this equality holds for the $\mathcal{L}$- and $\mathcal{R}$-components, so we can use Condition (e)(i) to get that $\Xi(a).\mathcal{A} = prof_k^{\mathfrak{T}}(a).\mathcal{A}$.

Now the equality of the $\mathcal{F}$-components follows from the fact that they are computed by the procedure `ComputeF` from Lemma 12. This finishes the proof. □

6.2 Size of models

In this section we show essentially optimal bounds for lengths of vertical and horizontal paths in 'minimal' models of normal form formulas, for all relevant navigational signatures.

What is crucial for the lower complexity bound in Thm. 8 is the ability to enforce doubly-exponentially long vertical paths. Let us see how to do it directly in $\mathrm{F}_1[\downarrow]$. We use unary predicates $N, P, P_0, \ldots, P_{n-1}, Q$. See the left part of Figure 7. The intended long path is the path of elements in N. Every element in N is going to have 2^n children marked by P, each of which has a *local position* in the range $[0, 2^n - 1]$ encoded by means of $P_0, \ldots, P_{n-1}$. Reading the truth-values of Q as binary digits we can assume that the collection of the P-children of a node in N encodes its *global position* in the tree in the range $[0, 2^{2^n} - 1]$ (the i-th bit of this global position is 1 iff at the element at local position i the value of Q is true). It is then possible to say that each node in n whose global position is smaller than $2^{2^n} - 1$ has a child in N with the global position greater by 1.

We employ the following abbreviations: $\lambda^{=}(x, y)$ in order to state that x and y have the same local position, $\lambda^{<}(x, y)$ to state that the local position of y is greater than the local position of x and $\lambda^{+1}(x, y)$ to state that the local position of y is one greater than the local position of x (addition modulo 2^n). All these abbreviations can be defined in the standard way using quantifier-free formulas of length polynomial in n. Formulas (6.1)–(6.6) take care of the basic shape of models (existence of

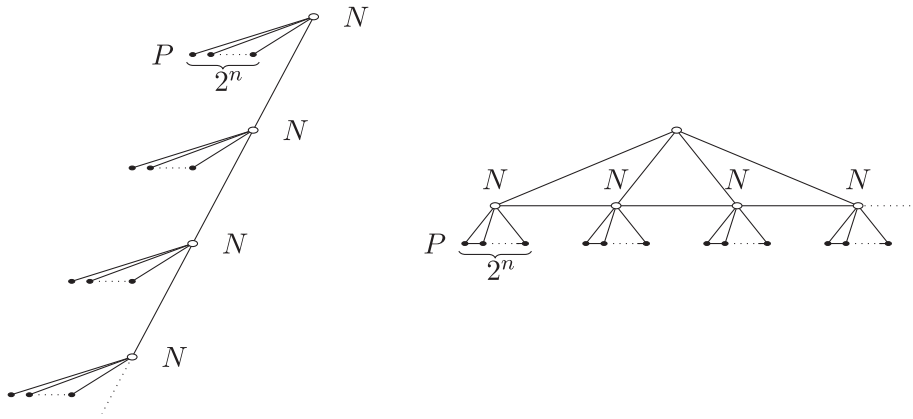

FIGURE 7 Enforcing doubly exponential vertical path in $\mathrm{F}_1[\downarrow]$ (left) and horizontal path in $\mathrm{F}_1[\downarrow_+, \rightarrow]$ or $\mathrm{F}_1[\downarrow, \rightarrow]$ (right).

N-successors, P-successors and exponentially many P-siblings with appropriate local positions):

$$\exists x(Nx \wedge \forall y(x{\downarrow}y \wedge Py \Rightarrow \neg Qy)) \tag{6.1}$$

$$\forall x(Nx \dot{\vee} Px) \tag{6.2}$$

$$\forall x(Nx \Rightarrow \exists y(x{\downarrow}y \wedge Py)) \tag{6.3}$$

$$\forall x(Px \Rightarrow \exists yz(z{\downarrow}x \wedge z{\downarrow}y \wedge Py \wedge \lambda^{+1}(x,y))) \tag{6.4}$$

$$\forall xyz(z{\downarrow}x \wedge z{\downarrow}y \wedge \lambda^{=}(x,y) \Rightarrow (Qx \leftrightarrow Qy)) \tag{6.5}$$

$$\forall x(Nx \wedge \exists y(x{\downarrow}y \wedge Py \wedge \neg Qy) \Rightarrow \exists y(x{\downarrow}y \wedge Ny)) \tag{6.6}$$

Let $\mu(x)$ abbreviate a formula stating that x is an element for which Q is false, and all its siblings with smaller local position have Q true. Now we can naturally encode $+1$ addition in our 2^{2^n}-global-position-counter:

$$\begin{aligned}\forall xyx'y'zt\big((z{\downarrow}x \wedge z{\downarrow}x' \wedge z{\downarrow}t \wedge t{\downarrow}y \wedge t \succ y' \wedge \\ Nx \wedge Nt \wedge Px \wedge Px' \wedge Py \wedge Py' \wedge \mu(x)\lambda^{=}(x,y) \wedge \lambda^{=}(x',y')) \Rightarrow \\ Q(y) \wedge \lambda^{<}(x',x) \Rightarrow \neg Qy' \wedge \lambda^{<}(x,x') \Rightarrow (Qy' \leftrightarrow Qx')\big)\end{aligned} \tag{6.7}$$

In an analogous way, assuming that $\rightarrow$ is available in the signature, that is in $\mathrm{F}_1[\downarrow, \rightarrow]$ or $\mathrm{F}_1[\downarrow_+, \rightarrow]$, we can enforce a doubly exponentially long horizontal path, like the path of the N in the right part of Figure 7.

It turns out that the presence of the successor relation(s) is crucial for enforcing doubly exponentially long vertical or horizontal paths. To show this let us prove two contraction lemmas.

LEMMA 17
Let σ be a signature with the navigational part containing $\rightarrow^+$. Let $\mathfrak{T}$ be a σ-tree and $a, b, a', b' \in T$

be such that b is a child of a, b' is a child of a', a' is a descendant of b, σ-$prof_k^{\mathfrak{T}}(a).\mathcal{B} = \sigma$-$prof_k^{\mathfrak{T}}(a').\mathcal{B}$ and σ-$prof_k^{\mathfrak{T}}(b).\mathcal{A} = \sigma$-$prof_k^{\mathfrak{T}}(b').\mathcal{A}$. Let $\mathfrak{T}'$ be the tree obtained from $\mathfrak{T}$ by replacing the subtree of a by the subtree of a', with the exception of the root of this subtree, which remains a. Then, for any node $c \in T'$ we have σ-$prof_k^{\mathfrak{T}}(c)$=σ-$prof_k^{\mathfrak{T}'}(c)$.

PROOF. We consider three cases. In the first two of them we analyse the profiles of the elements lying next to the cut made in our surgery, in the third one we systematically analyse the profiles of the remaining elements of $\mathfrak{T}'$.

(i) Assume first that c is a child of a in $\mathfrak{T}'$ (that is, it is a child of a' in $\mathfrak{T}$). Clearly the $\mathcal{L}$-, $\mathcal{B}$- and $\mathcal{R}$-component of $prof_k^{\mathfrak{T}}(c)$ are retained in $\mathfrak{T}'$, since the subtrees of c and its siblings are the same as in $\mathfrak{T}$. We need to see that also the $\mathcal{A}$-component is retained. Let $\pi = type^{\mathfrak{T}'}(c, a_1, \ldots, a_s) \in prof_k^{\mathfrak{T}'}(c).\mathcal{A}$. Let $\pi' = type^{\mathfrak{T}}(b, a_1, \ldots, a_s)$. As $\pi' \in prof_k^{\mathfrak{T}}(b).\mathcal{A}$, by the assumption of the Lemma we have that $\pi' \in prof_k^{\mathfrak{T}}(b').\mathcal{A}$. Let $b_1, \ldots, b_s$ be elements in position A to b' in $\mathfrak{T}$ such that $\pi' = type^{\mathfrak{T}}(b', b_1, \ldots, b_s)$. Observe that then $type^{\mathfrak{T}}(c, b_1, \ldots, b_s) = \pi$ and thus $\pi \in prof_k^{\mathfrak{T}}(c).\mathcal{A}$.

In the opposite direction assume that $\pi = type^{\mathfrak{T}}(c, a_1, \ldots, a_s) \in prof_k^{\mathfrak{T}}(c).\mathcal{A}$. Let $\pi' = type^{\mathfrak{T}}(b', a_1, \ldots, a_s)$. As $\pi' \in prof_k^{\mathfrak{T}}(b').\mathcal{A}$, by the assumption of the lemma we have that $\pi' \in prof_k^{\mathfrak{T}}(b).\mathcal{A}$. Let $b_1, \ldots, b_s$ be elements in position A to b in $\mathfrak{T}$ such that $\pi' = type^{\mathfrak{T}}(b, b_1, \ldots, b_s)$. Observe that then $type^{\mathfrak{T}'}(c, b_1, \ldots, b_s) = \pi$ and thus $\pi \in prof_k^{\mathfrak{T}'}(c).\mathcal{A}$.

(ii) Consider now the element a. Clearly, the $\mathcal{L}$-, $\mathcal{A}$- and $\mathcal{R}$-component of $prof_k^{\mathfrak{T}}(a)$ are retained in $\mathfrak{T}'$, since from the point of view of a, the only part of the tree that changes is its subtree, and this change may influence at most the $\mathcal{B}$-component of the profile of a. That this component also does not change follows straightforwardly from the assumption of the lemma that σ-$prof_k^{\mathfrak{T}}(a).\mathcal{B} = \sigma$-$prof_k^{\mathfrak{T}}(a').\mathcal{B}$, since in $\mathfrak{T}'$ the subtree of a is replaced by the subtree of a' (with the exception of the root which is still a).

(iii) If $c \neq a$ and c is not a child of a then note that c retains in $\mathfrak{T}'$ all its *direct neighbours* (i.e. the parent, the children, the next sibling and the previous sibling) from $\mathfrak{T}$. We will use the fact that each component of the profile of an element is determined by its 1-type (which is obviously retained from $\mathfrak{T}$) and some components of the profiles of its direct neighbours, as stated in Lemma 15. We will now systematically analyse the profiles of the elements of $\mathfrak{T}'$

(a) If c belongs to the subtree rooted at a child c' of a then the $\mathcal{L}$-, $\mathcal{B}$- and $\mathcal{R}$-component of $prof_k^{\mathfrak{T}}(c)$ are retained in $\mathfrak{T}'$, because the subtrees of c' and its siblings are exactly as in $\mathfrak{T}$. For the $\mathcal{A}$-component we proceed by induction on the depth of c in the subtree of c' using the fact that the $\mathcal{A}$-component of the profile of an element d is uniquely determined by its 1-type and by the $\mathcal{L}$-, $\mathcal{A}$- and $\mathcal{R}$-components of the profile of its parent (Lemma 15 (i)(a)).

(b) If c is a left sibling of a then the $\mathcal{L}$-, $\mathcal{B}$- and $\mathcal{A}$-component of $prof_k^{\mathfrak{T}}(c)$ are retained in $\mathfrak{T}'$, because, from the point of view of a only some elements in position R could change. For the $\mathcal{R}$-component we proceed by induction on the distance of c from a using the fact that the $\mathcal{R}$-component of the profile of an element d is uniquely determined by its 1-type and by the $\mathcal{R}$- and $\mathcal{B}$-components of the profile of its next sibling (Lemma 15 (ii)(b)).

(c) If c is a right sibling of a then we proceed symmetrically (using Lemma 15 (ii)(a)).

(d) If c is in the subtree of a sibling b of a then we proceed as in (a) using top-down induction on the distance from b to deal with the $\mathcal{A}$-component.

(e) If c is an ancestor of a then the $\mathcal{L}$-, $\mathcal{A}$- and $\mathcal{R}$-component of $prof_k^{\mathfrak{T}}(c)$ are retained in $\mathfrak{T}'$ since, from the point of view of c only some of its descendants has changed. For the the $\mathcal{B}$-component we proceed by induction on the distance of c from a using the fact that the $\mathcal{B}$-component of the profile

of an element d is uniquely determined by its 1-type and by the $\mathcal{L}$-, $\mathcal{B}$- and $\mathcal{R}$-components of the profile of its any child, in particular of the child on the vertical path to a (Lemma 15 (i)(b)).

(f) If c is a sibling of an ancestor of a then we proceed as in (b) or (c)

(g) Any remaining c is now in the subtree rooted at an element about which we already know that its profile is retained from $\mathfrak{T}$ and thus we can proceed as in (a) and (d) using top-down induction to deal with the $\mathcal{A}$-component. □

LEMMA 18

Let σ be a signature with the navigational part containing $\rightarrow^+$. Let $\mathfrak{T}$ be a σ-tree and $a, a' \in T$ be such that a' is a following sibling of a, $\sigma\text{-prof}_k^{\mathfrak{T}}(a).\mathcal{L} = \sigma\text{-prof}_k^{\mathfrak{T}}(a').\mathcal{L}$ and $\sigma\text{-prof}_k^{\mathfrak{T}}(a).\mathcal{R} = \sigma\text{-prof}_k^{\mathfrak{T}}(a').\mathcal{R}$. Let $\mathfrak{T}'$ be the tree obtained from $\mathfrak{T}$ by removing all the subtrees rooted at the elements lying on the horizontal path from a to a', including a and excluding a' (and thus making the next sibling of a' in $\mathfrak{T}$ the next sibling of a in $\mathfrak{T}'$). Then, for any node $c \in T'$ we have $\sigma\text{-prof}_k^{\mathfrak{T}}(c) = \sigma\text{-prof}_k^{\mathfrak{T}'}(c)$.

PROOF. Let b be the next sibling of a' in $\mathfrak{T}$ and e their parent. As in the previous proof we first see that the profiles of a, b and e, that is the elements lying next to the cut, do not change and then propagate our analysis to the remaining elements.

(i) For a it is clear that the $\mathcal{L}$-, $\mathcal{B}$- and $\mathcal{A}$-components of its profile do not change. For the $\mathcal{R}$-component we naturally use the assumption of the lemma that $\sigma\text{-prof}_k^{\mathfrak{T}}(a).\mathcal{R} = \sigma\text{-prof}_k^{\mathfrak{T}}(a').\mathcal{R}$ (note that it implies, in particular, that the 1-types of a and a' are identical) and the fact that what a can see in position R in $\mathfrak{T}'$ is exactly what a' can see in position R in $\mathfrak{T}$.

(ii) The case of b is symmetric.

(iii) For e, the $\mathcal{L}$-, $\mathcal{R}$- and $\mathcal{A}$-components of its profile clearly do not change. For the $\mathcal{B}$-component take any $\pi \in \text{prof}_k^{\mathfrak{T}}(e).\mathcal{B}$ and assume $\pi = \text{type}^{\mathfrak{T}}(e, a_1, \ldots, a_s)$, for $a_1, \ldots, a_s$ in position B to e. W.l.o.g. assume that $a_1, \ldots, a_t$ are the elements lying in position L or B to a or being a itself, and that $a_{t+1}, \ldots, a_t$ are the elements in position R to a. Since $\pi' = \text{type}^{\mathfrak{T}}(a, a_{t+1}, \ldots, a_s)$ belongs to $\text{prof}_k^{\mathfrak{T}}(a).\mathcal{R}$, by the assumption of the Lemma we have that $\pi' \in \text{prof}_k^{\mathfrak{T}}(a').\mathcal{R}$. Let $b_{t+1}, \ldots, b_s$ be elements in position R to a' in $\mathfrak{T}$ such that $\pi' = \text{type}^{\mathfrak{T}}(a', b_{t+1}, \ldots, b_s)$. Now $\text{type}^{\mathfrak{T}'}(e, a_1, \ldots, a_t, b_{t+1}, \ldots, b_s) = \pi$ and thus $\pi \in \text{prof}_k^{\mathfrak{T}'}(e).\mathcal{B}$. In the opposite direction we proceed similarly.

(iv) For the remaining elements of $\mathfrak{T}'$ we argue analogously as in case (iii) of the proof of Lemma 17, that is we use the fact that those elements retain their neighbours from $\mathfrak{T}$ and use Lemma 15 to propagate the equalities of the profiles of computed in $\mathfrak{T}$ and $\mathfrak{T}'$ towards the other parts of $\mathfrak{T}'$. □

With the help of the above lemmas we can now easily get essentially optimal upper bounds on the lengths of paths.

THEOREM 9

There are a fixed doubly exponential function $\mathfrak{g}$ and a singly exponential function $\mathfrak{f}$ such that

(i) For any navigational signature $\sigma_{nav} \subseteq \{\downarrow, \downarrow_+, \leftarrow, \rightarrow^+\}$ every satisfiable $\text{F}_1[\sigma_{nav}]$ formula φ has a model in which horizontal and vertical paths have length bounded from above by $\mathfrak{g}(\|\varphi\|)$.

(ii) Any satisfiable formula φ in $\text{F}_1[\downarrow, \downarrow_+, \rightarrow^+]$ has a model in which horizontal paths have length bounded from above by $\mathfrak{f}(\|\varphi\|)$ (and vertical paths are bounded by $\mathfrak{g}(\|\varphi\|)$).

(iii) Any satisfiable formula in $\text{F}_1[\downarrow_+, \rightarrow, \rightarrow^+]$ has a model in which the length of vertical paths is bounded from above by $\mathfrak{f}(\|\varphi\|)$ (and horizontal paths are bounded by $\mathfrak{g}(\|\varphi\|)$).

(iv) Any satisfiable formula in $\text{F}_1[\downarrow_+, \rightarrow^+]$ has a model in which vertical paths and horizontal paths have length bounded from above by $\mathfrak{f}(\|\varphi\|)$.

PROOF. Let us take a normal form formula φ over a signature $\sigma = \sigma_0 \cup \sigma_{nav}$ and denote by k its width. Let $\mathfrak{T} \models \varphi$. First, until there are elements $a, a' \in T$ meeting the assumptions of Lemma 18 replace $\mathfrak{T}$ by $\mathfrak{T}'$ as in this lemma. Let $\mathfrak{T}^*$ be the tree eventually obtained. Clearly every horizontal path in $\mathfrak{T}^*$ contains elements of distinct k-σ-profiles. By Lemma 13 the number of such profiles is bounded by $\mathfrak{g}^*(|\sigma_0|, k)$. As $k \leq |\varphi|$ and as we may assume that σ_0 consists only of the unary relations appearing in φ, also $|\sigma_0| \leq \|\varphi\|$ we get that $\mathfrak{T}^*$ has paths bounded by $\mathfrak{g}^*(\|\varphi\|, \|\varphi\|)$, doubly exponentially in $\|\varphi\|$.

Further, take $\mathfrak{T} := \mathfrak{T}^*$ and as long as there are elements $a, a', b, b' \in T$ meeting the assumptions of Lemma 17 replace $\mathfrak{T}$ by $\mathfrak{T}'$ as in this lemma. Let $\mathfrak{T}^\dagger$ be the tree eventually obtained. Take a vertical path in $\mathfrak{T}^\dagger$ and split it into segments consisting of two consecutive elements each (possibly with the exception of the last segment, which may consists of a single element if the number of elements on the path is odd). Clearly, every two pairs have different combination of k-σ-profiles, since otherwise a further contraction step would be possible. The number of such combinations is bounded by $(\mathfrak{g}^*(\|\varphi\|, \|\varphi\|))^2$, doubly exponentially in $\|\varphi\|$. As horizontal paths in $\mathfrak{T}^\dagger$ are also horizontal paths in $\mathfrak{T}^*$ we have that $\mathfrak{T}^\dagger$ is a witness to (i), where as $\mathfrak{g}(\|\varphi\|)$ we take $2\mathfrak{g}^*(\|\varphi\|, \|\varphi\|)^2 + 1$ (two elements in each pair plus possibly the last element on the path if their number is odd).

Next, note that if $\rightarrow \notin \sigma_{nav}$ then, if $\mathfrak{T}^* \models a \rightarrow^+ a'$ and a, a' have the same 1-type then σ-$prof_k^{\mathfrak{T}^*}(a).\mathcal{L} \subseteq \sigma$-$prof_k^{\mathfrak{T}^*}(a').\mathcal{L}$ and σ-$prof_k^{\mathfrak{T}^*}(a).\mathcal{R} \supseteq \sigma$-$prof_k^{\mathfrak{T}^*}(a').\mathcal{R}$. Thus, when moving along a horizontal path from left to right through the elements of the same 1-type, the $\mathcal{L}$-components of the profiles of elements either stay unchanged or grow, and the $\mathcal{R}$-components either stay unchanged or diminish, but in each step at least one of these must change since otherwise a contraction step as in Lemma 18 would be possible. As the number of 1-types and the size of $\mathcal{L}$- and $\mathcal{R}$-components is bounded exponentially in $\|\varphi\|$ (cf. Lemma 13) we conclude that the horizontal paths in $\mathfrak{T}^*$ are bounded exponentially in $\|\varphi\|$. This justifies (ii).

Reasoning similarly as in the above paragraph, but using the $\mathcal{A}$- and $\mathcal{B}$-components and Lemma 17 we can show that if $\downarrow \notin \sigma_{nav}$ then the vertical paths in $\mathfrak{T}^\dagger$ are bounded exponentially in φ. Take a vertical path in $\mathfrak{T}^\dagger$ and split it into segment of size two. If there are segments $\langle a, b\rangle$ and $\langle a', b'\rangle$ such that a' is a descendant of b, the 1-types of a and a' are equal, and the 1-types of b and b' are equal then $prof_k^{\mathfrak{T}^*}(b).\mathcal{A} \subseteq \sigma$-$prof_k^{\mathfrak{T}^*}(b').\mathcal{A}$ and $prof_k^{\mathfrak{T}^*}(a).\mathcal{B} \supseteq \sigma$-$prof_k^{\mathfrak{T}^*}(s').\mathcal{B}$, but at least one of the above inclusions must be strict since otherwise a contraction step as in Lemma 17 would be possible. Since the sizes of the components are bounded exponentially in $\|\varphi\|$, there are exponentially many segments for any fixed pair of 1-types of its elements. As the number of 1-types is also bounded exponentially we get (iii).

Finally, if none of $\downarrow$, $\rightarrow$ belongs to σ_{nav}, then by the arguments above, $\mathfrak{T}^\dagger$ has exponentially bounded horizontal and vertical paths, which proves (iv). □

6.3 Complexity

Using Theorem 9 one could establish the optimal upper complexity bounds for satisfiability of F_1 over trees for any navigational signature. We concentrate on the case of the signature $\{\downarrow_+, \rightarrow, \rightarrow^+\}$ which will allow us to complete the picture concerning the complexity of satisfiability, and then roughly explain how similar approach can be used to directly prove the upper bound in Theorem 8, which we have already proved by a reduction to the unary negation fragment.

THEOREM 10
Let $\{\downarrow_+\} \subseteq \sigma_{nav} \subseteq \{\downarrow_+, \rightarrow, \rightarrow^+\}$. Then the satisfiability problem for $F_1[\sigma_{nav}]$ is EXPSPACE-complete.

PROOF. The lower bound for $F_1[\downarrow_+]$ is inherited from $FO^2[\downarrow_+]$, [2], which in turn refers to EXPSPACE-hardness of the so-called one-way two-variable guarded fragment, [15].

To justify the upper bound for $F_1[\downarrow_+, \rightarrow, \rightarrow^+]$ we propose a nondeterministic algorithm working in exponential space checking if a given normal form formula φ is satisfiable. As by Savitch theorem NEXPSPACE=EXPSPACE, the result follows.

Let k be the width of φ. Our algorithm attempts to construct a model $\mathfrak{T}$ together with functions Ω and Ξ assigning to each node $a \in T$ a 1-type and, respectively, a tuple $(\mathcal{F}, \mathcal{A}, \mathcal{B}, \mathcal{L}, \mathcal{R})$ of sets of s-types for various $s \leq k$, intended to be the 1-type and, respectively, the k-profile of a in $\mathfrak{T}$. For each constructed tuple $(\mathcal{F}, \mathcal{A}, \mathcal{B}, \mathcal{L}, \mathcal{R})$ it immediately checks if $\mathcal{F} = \mathtt{fulltype}(\mathcal{A}, \mathcal{B}, \mathcal{L}, \mathcal{R})$ and rejects if it is not the case. Additionally, the algorithm stores for each node a its position $hcount(a)$ in the horizontal path of its siblings and its position $vcount(a)$ on the vertical path from the root to a.

The algorithm starts with constructing the root ϵ of $\mathfrak{T}$, that is by guessing the values $\Omega(\epsilon)$, $\Xi(\epsilon)$ and setting $hcount(\epsilon) := 0$ and $vcount(\epsilon) := 0$. It then verifies that the values of Ω and Ξ on ϵ respect Conditions (a), (c) and (d) of Def. 1.

Then the algorithm works in a depth-first manner, that is, being at a node a it first goes down to the leftmost child of a (or just decides that a is a leaf), analyses the subtree of a, marks a as "visited", then goes right to the next sibling of a, proceeds a, and so on; when it decides that the rightmost child in a horizontal path of siblings is reached it goes up.

At any moment the algorithm stores the whole vertical path from the root ϵ to the current node. When making a step down from a node a to a new node a' the algorithm guesses $\Omega(a')$, $\Xi(a')$, sets $hcount(a') := 0$ and $vcount(a') := vcount(a) + 1$, verifies that the values of Ω and Ξ on a and a' respect Condition (e) of Def. 1, and that their values on a' respect condition (c) of Def. 1.

When making a step right from a node a, which is a child of a node b, to a new node a', the algorithm guesses $\Omega(a')$, $\Xi(a')$, sets $hcount(a') := hcount(a) + 1$ and $vcount(a') := vcount(a)$ and verifies that the values of Ω and Ξ on a and a' respect Condition (f) of Def. 1, and that their values on b and a' respect Condition (e) of Def. 1.

When the algorithm nondeterministically decides that the current node a is a leaf, it verifies that the values of Ω and Ξ on a respect Condition (b) of Def. 1. When the algorithm nondeterministically decides that the current node a is the rightmost child on a horizontal path, it verifies that the values of Ω and Ξ on a respect Condition (d) of Def. 1.

The algorithm rejects if the value of $hcount$ at any node exceeds $\mathfrak{g}(|\varphi|)$ or the value of $vcount$ at any node exceeds $\mathfrak{f}(|\varphi|)$ or if the values of Ξ on any node, treated as a profile, is not compatible with φ (cf. Lemma 14). It accepts when it returns back to the root without noticing any violation of the local consistency conditions or φ-compatibility.

That the algorithm uses only exponential space should be clear: it stores a single vertical path of length bounded exponentially by $\mathfrak{f}(|\varphi|)$ plus possibly one sibling of the currently inspected node. The values of the counters and the functions Ω and Ξ stored at each node are also of exponential size.

Let us finally explain the correctness of the algorithm. Assume that φ is satisfiable. By Thm. 9 (iii) φ has a model $\mathfrak{T}$ with vertical paths bounded by $\mathfrak{f}(|\varphi|)$ and horizontal paths bounded by $\mathfrak{g}(|\varphi|)$. An accepting run of the algorithm can be then naturally constructed by making all the guesses in accordance with $\mathfrak{T}$. In the opposite direction, if the algorithm has an accepting run, then we can naturally extract from this run a tree $\mathfrak{T}$ in which the 1-types of nodes are as given by the function Ω. Since during the run the local consistency of the pair (Ω, Ξ) is checked, it follows by 16 that for each $a \in T$ we have $\Xi(a) = prof_k^{\mathfrak{T}}(a)$. That $\mathfrak{T} \models \varphi$ follows then by Lemma 14, since the algorithm verifies at each node a that $\Xi(a)$ is compatible with φ. □

As promised, we shall finally briefly explain how to reprove the upper bound in Thm. 8 using the technique we have developed. Recall that Thm. 9 (i) says that every satisfiable formula has a model in which the length of vertical and horizontal paths is bounded from above doubly exponentially by the function $\mathfrak{g}$. Our examples illustrated in Figure 7 demonstrate that in the case of $\sigma_{nav} = \{\downarrow, \downarrow_+, \rightarrow, \rightarrow^+\}$ we indeed need to take into account models with at least doubly exponential paths, which thus have triply exponentially many nodes. This means that to fit in 2-ExpTime we cannot walk through the whole model. Instead we propose an algorithm for an alternating machine with exponentially bounded space. This suffices for our purposes, since by the well-known result by Chandra et al. [7] AExpSpace=2-ExpTime, i.e. any algorithm working in alternating exponential space can be turned into an algorithm working in doubly exponential time.

Given a normal form $F_1[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$ formula φ, the algorithm attempts to construct a single walk through a tree being a model of φ from the root to a leaf, at each node making a universal choice whether to go down (to the leftmost child of the current node) or to go right (to the next sibling of the current node).

The other details are as in the procedure from the the proof of Thm. 10: the algorithm operates on similar data structures, i.e. at each nodes it guesses the values of Ω and Ξ and appropriately updates the counters *vcount* and *hcount*. At each step it also guesses if the current node is the rightmost child or a leaf, checks if the values of the counters do not exceed $\mathfrak{g}(\|\varphi\|)$, and verifies the local consistency conditions from 16 and φ-compatibility conditions from Lemma 5.

Arguments similar to those from the proof of Thm 10 ensure that φ has a model iff the algorithm has an accepting run.

7 Conclusions

In this paper we investigated the one-dimensional fragment of first-order logic, F_1, over words and trees and collated our results with the results on a few important formalisms for speaking about those classes of structures.

Regarding expressivity, all the considered formalisms (CoreXPath, GF^2, FO^2, C^2, UNFO, F_1) are equiexpressive over words, while over trees it depends on the navigational signature: over XML trees (child, descendant, next sibling, following sibling) again all the logics are equiexpressive, but over unordered trees (only child and descendant) they differ in the expressivity, with F_1 being as expressive as C^2 but more expressive than UNFO and FO^2.

Concerning the complexity of the satisfiability problem, the picture is presented in Table 1. Column $\{\rightarrow, \rightarrow^+\}$ concerns the case of words. The remaining columns show the results for the case of trees. We have chosen the four most interesting navigational signatures (XML trees, unordered trees with both child and descendant, and unordered trees accessible by only descendant or only child). In the case of CoreXPath we assume that both downward and upward modalities (and both left and right modalities in the case of XML trees) are present in each of the considered variations

For convenience, below we recall the references to the results in the table. The PSpace result for UTL is proved in [10]. For CoreXPath the ExpTime-results follow from [22], while the PSpace-completeness for $\{\downarrow\}$ is proved in [5]; the argument for PSpace-completeness in the case of $\{\downarrow_+\}$ is sketched in the Appendix. NExpTime-completeness of FO^2 over words is shown in [10]; this holds also for GF^2, as in the case of words every pair of elements is guarded by $\rightarrow^+$ and thus any FO^2 formula can be easily translated into GF^2. GF^2 and FO^2 over trees are thoroughly examined in [2]. NExpTime-completeness of C^2 over words is shown in [8]. C^2 over trees is investigated in [3] where ExpSpace-completeness for signatures containing $\downarrow_+$ is proved; the signature $\{\downarrow\}$ is not

TABLE 1 Complexity over words and trees. Results in bold are proved in this paper. We have not found the results in grey in the literature but they can be easily derived using the existing techniques (see the Appendix).

	$\{\rightarrow, \rightarrow^+\}$	$\{\downarrow, \downarrow_+, \rightarrow, \rightarrow^+\}$	$\{\downarrow, \downarrow_+\}$	$\{\downarrow_+\}$	$\{\downarrow\}$
UTL / CoreXPath	PSPACE	EXPTIME	EXPTIME	grayPSPACE	PSPACE
GF^2	NEXPTIME	EXPSPACE	EXPSPACE	EXPSPACE	EXPTIME
FO^2	NEXPTIME	EXPSPACE	EXPSPACE	EXPSPACE	NEXPTIME
C^2	NEXPTIME	EXPSPACE	EXPSPACE	EXPSPACE	grayNEXPTIME
UNFO	**NEXPTIME**	2-EXPTIME	2-EXPTIME	**EXPSPACE**	2-EXPTIME
F_1	**NEXPTIME**	**2-EXPTIME**	**2-EXPTIME**	**EXPSPACE**	**2-EXPTIME**

studied there, and we sketch an argument for NEXPTIME-completeness in this case in the Appendix. Finally, 2-EXPTIME-results for UNFO are proved in [27].

What is probably interesting to note is that in the case of the two-variable logics over trees, it is the signature $\{\downarrow\}$ which is easier than the other signatures. In the case of the multi-variable logics F_1 and UNFO this signature is equally hard as our full navigational signature, but a complexity drop can be observed this time for the signature $\{\downarrow_+\}$.

One more interesting issue that we have not investigated in detail in this paper is succinctness. Our work implies that F_1 is exponentially more succinct than FO^2 over XML trees and more generally over trees whose signatures contain $\downarrow$. This follows from the fact that every satisfiable $FO^2[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$ formula has a model whose paths are bounded exponentially in its size [2] (this holds also for C^2 [3]) while already in $F_1[\downarrow]$ we can enforce models with doubly exponentially long paths (Section 6.2 of this paper).

The above argument does not work in the case of words, since we have shown that in F_1 at most exponentially large models can be enforced, and this indeed can be also done in FO^2. Nevertheless, we suspect that also in this case F_1 is more succinct, which is suggested by the examples presented in the Introduction.

Acknowledgements

This work was supported by the project 'Theory of computational logics' funded by the Academy of Finland [grants 324435 and 328987 to AK], the project 'Explaining AI via logic (XAILOG)' funded by the Academy of Finland [grant 345612], and the project 'A quest for new computer logics' funded by Polish National Science Centre [grant 2016/21/B/ST6/01444 to EK].

References

[1] H. Andréka, I. Németi and J. van Benthem. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, **27**, 217–274, 1998.

[2] S. Benaim, M. Benedikt, W. Charatonik, E. Kieronski, R. Lenhardt, F. Mazowiecki and J. Worrell. Complexity of two-variable logic on finite trees. *ACM Transactions on Computational Logic*, **17**, 32:1–32:38, 2016.

[3] B. Bednarczyk, W. Charatonik and E. Kieronski. Extending two-variable logic on trees. In *The 26th EACSL Annual Conference on Computer Science Logic, CSL 2017*, 20–24 August 2017,

Stockholm, Sweden, pp. 11:1–11:20, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.
[4] M. Bojańczyk, C. David, A. Muscholl, T. Schwentick and L. Segoufin. Two-variable logic on data words. *ACM Transactions on Computational Logic*, **12**, 27, 2011.
[5] M. Benedikt, W. Fan and F. Geerts. Xpath satisfiability in the presence of dtds. *Journal of the ACM*, **55**, 8:1–8:79, 2008.
[6] W. Charatonik, E. Kieronski and F. Mazowiecki. Satisfiability of the two-variable fragment of first-order logic over trees. *CoRR*, abs/1304.7204, 2013.
[7] A. K. Chandra, D. Kozen and L. J. Stockmeyer. Alternation. *Journal of the ACM*, **28**, 114–133, 1981.
[8] W. Charatonik and P. Witkowski. Two-variable logic with counting and a linear order. *Logical Methods in Computer Science*, **12**, 1–31, 2016.
[9] W. Charatonik and P. Witkowski. Two-variable logic with counting and trees. *ACM Transactions on Computational Logic*, **17**, 31:1–31:27, 2016.
[10] K. Etessami, M. Y. Vardi and T. Wilke. First-order logic with two variables and unary temporal logic. *Inf. Comput.*, **179**, 279–295, 2002.
[11] D. M. Gabbay, I. Hodkinson and M. Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects,* vol. 1, 653 pp. Oxford Science Publications, 1994.
[12] E. Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, **64**, 1719–1742, 1999.
[13] L. Hella and A. Kuusisto. One-dimensional fragment of first-order logic. *Advances in Modal Logic*, **10**, 274–293, 2014.
[14] H. Kamp. *Tense Logic and the Theory of Linear Order*. PhD Thesis, University of California, Los Angeles, 1968.
[15] E. Kieronski. On the complexity of the two-variable guarded fragment with transitive guards. *Information and Computation*, **204**, 1663–1703, 2006.
[16] E. Kieronski. One-dimensional logic over words. In *The 25th EACSL Annual Conference on Computer Science Logic, CSL 2016*, 29 August to 1 September 2016, Marseille, France, pp. 38:1–38:15, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.
[17] E. Kieronski and A. Kuusisto. Complexity and expressivity of uniform one-dimensional fragment with equality. *Mathematical Foundations of Computer Science 2014*, 365–376, 2014.
[18] E. Kieronski and A. Kuusisto. Uniform one-dimensional fragments with one equivalence relation. In *Computer Science Logic*. Volume 41 of LIPIcs, pp. 597–615, 2015.
[19] E. Kieronski and A. Kuusisto. One-dimensional logic over trees. In *42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017*, 21–25 August 2017, Aalborg, Denmark, pp. 64:1–64:13, 2017.
[20] A. Krebs, K. Lodaya, P. K. Pandya and H. Straubing. Two-variable logics with some betweenness relations: expressiveness, satisfiability and membership. *Logical Methods in Computer Science*, **16**, 16:1–16:41, 2020.
[21] A. Kuusisto. On the uniform one-dimensional fragment. In *Proceedings of International Workshop on Description Logic*, CEUR-WS.org, 2016.
[22] M. Marx. Xpath with conditional axis relations. *Advances in Database Technology - EDBT*, **2004**, 477–494, 2004.
[23] M. Marx and M. de Rijke. Semantic characterization of navigational XPath. *ACM SIGMOD Record*, **34**, 41–46, 2004.
[24] M. Otto. Two variable first-order logic over ordered domains. *Journal of Symbolic Logic*, **66**, 685–702, 2001.

[25] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, **32**, 733–749, 1985.
[26] D. Scott. A decision method for validity of sentences in two variables. *Journal Symbolic Logic*, **27**, 477, 1962.
[27] L. Segoufin and B. ten Cate. Unary negation. *Logical Methods in Computer Science*, **9**, 1–46, 2013.
[28] L. J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory and Logic*. PhD Thesis, MIT, Cambridge, Massachusetts, USA, 1974.

A Missing complexities

THEOREM 11
The satisfiability problem for CoreXPath[$\downarrow_+$] is PSPACE-complete.

PROOF. (Sketch) The lower bound can be shown in a standard fashion by a reduction from the QBF problem; alternatively one can use PSPACE-hardness of the modal logic K4.

To get the upper bound we show that every satisfiable formula has a model of depth and degree bounded polynomially in its length. This can be done by the following standard selection process. We extend the language by box modalities [$\downarrow_+$], [$\uparrow^+$], with their standard semantics: $[\cdot]\varphi := \neg\langle\cdot\rangle\neg\varphi$. For a given input formula, using de Morgan laws, we push all the negations down to the propositional variables. Let φ be the NNF result of this process, and let $SF(\varphi)$ be the set of its subformulas.

Take now a tree $\mathfrak{T}$ and its node c such that $\mathfrak{T}, c \models \varphi$. We first take care of the length of vertical path. For each $\langle\downarrow_+\rangle\psi \in SF(\varphi)$ mark all minimal nodes a such that $\mathfrak{T}, a \models \psi$. Similarly, for each $\langle\uparrow^+\rangle\psi \in SF(\varphi)$ mark all maximal nodes a such that $\mathfrak{T}, a \models \psi$. Mark also the element c. Let $\mathfrak{T}^*$ be the result of removing from $\mathfrak{T}$ all the unmarked elements and rebuilding the structure of the tree on the marked ones, so that the relation $\downarrow_+$ from $\mathfrak{T}$ is respected. By the structural induction we can now show that for any $\psi \in SF(\varphi)$ and any node $a \in T^*$, if $\mathfrak{T}, a \models \psi$ then $\mathfrak{T}^*, a \models \psi$; in particular $\mathfrak{T}^*, c \models \varphi$. Since the size of $SF(\varphi)$ is linear in $\|\varphi\|$ it follows that the paths of $\mathfrak{T}^*$ are also bounded linearly in $\|\varphi\|$.

Next we take care of the degree of nodes. Proceeding in breadth-first manner we repeat for all nodes a of $\mathfrak{T}^*$: for every $\langle\downarrow_+\rangle\psi$, if ψ holds at an descendant of a then mark one such descendant; mark also c if it is a descendant of a. Remove all the subtrees rooted at the children of a which do not contain any marked node. After this process the resulting tree has the degree of nodes and the length of the vertical paths bounded linearly in $\|\varphi\|$.

Finally, we can check the existence of models with linearly bounded length of paths and degree by guessing their nodes in a depth-first manner. A natural decision procedure can be designed to work in NPSPACE=PSPACE. □

THEOREM 12
The satisfiability problem for $C^2[\downarrow]$ is NEXPTIME-complete.

PROOF. (Sketch) The lower bound is inherited from monadic FO^2 (with no navigational predicates). The upper bound can be proved by an adaptation of the upper bound proof for $FO^2[\downarrow, \rightarrow, \rightarrow^+]$ in [2]. It will work even in the richer scenario of $C^2[\downarrow, \rightarrow, \rightarrow^+]$. We first convert the input formula into

Scott-type normal form from [3]:

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

where $\bowtie_i \in \{\leq, \geq\}$, each C_i is a natural number, and $\chi(x,y)$ and all the $\chi_i(x,y)$ are quantifier-free. Denote $C = \max\{C_i\}_{i=1,\ldots,m}$.

We then mostly repeat the construction from the proof of Thm. 4.1 from [2]. We start from a model of φ with exponentially bounded horizontal and vertical paths as guaranteed by Thm. 18 in [3]. Then, the only real modification of the proof from [2] is that when selecting the set of *protected witnesses* W_1 we choose C representatives of each 1-type (or all of them if there are less than C of them) rather than just one. Similarly, when selecting *incomparable witnesses* for the elements of W_1 we also add to the set W_2 C incomparable witnesses for each element and conjunct of type $\forall\exists$ (or all of them if there are less than C of them). Since the number of 1-types and the value of C are bounded exponentially in the size of φ it follows that the size of $W = W_1 \cup W_2$ is also bounded exponentially.

We then proceed as in [2] to prove that there exists a model of φ with exponentially many non-isomorphic subtrees. Such models can be represented as DAGs of exponential size, which can be then naturally used to test satisfiability in NEXPTIME by guessing such a representation and verifying that it indeed encodes a model of φ. □

Received 6 October 2021