UNARY NEGATION

BALDER TEN CATE [a] AND LUC SEGOUFIN [b]

[a] UC Santa Cruz
  e-mail address: btencate@ucsc.edu

[b] INRIA and ENS Cachan, LSV
  e-mail address: luc.segoufin@inria.fr

ABSTRACT. We study fragments of first-order logic and of least fixed point logic that allow only unary negation: negation of formulas with at most one free variable. These logics generalize many interesting known formalisms, including modal logic and the μ-calculus, as well as conjunctive queries and monadic Datalog. We show that satisfiability and finite satisfiability are decidable for both fragments, and we pinpoint the complexity of satisfiability, finite satisfiability, and model checking. We also show that the unary negation fragment of first-order logic is model-theoretically very well behaved. In particular, it enjoys Craig Interpolation and the Projective Beth Property.

1. INTRODUCTION

Vardi [46] raised the question "why is modal logic so robustly decidable?". His explanation centers around the fact that modal logic has the tree-model property. More precisely, modal logic enjoys a combination of three properties, namely (i) the *tree-model property* (if a sentence has a model, it has a model that is a tree), (ii) *translatability into tree automata* (each formula can be transformed into a tree automaton, or equivalently, an MSO formula, recognizing its tree models), and (iii) the *finite model property* (if a formula has a model, it also has a finite model). These three properties form a powerful explanation for the decidability of the satisfiability problem, and the finite satisfiability problem, for modal logic and many of its extensions such as the modal μ-calculus. The guarded fragment of first-order logic (GFO) was proposed by Andréka, van Benthem and Németi [1] as a large fragment of first-order logic that generalizes modal logic while essentially retaining these properties. It consists of FO formulas in which all quantifiers are "guarded" by atomic formulas. GFO has the tree-like model property (if a sentence has a model, it has a model of bounded tree width), it can be translated into tree automata (each formula can be

transformed into a tree automaton recognizing the tree decompositions of its models of bounded tree width) and it has the finite model property [1, 26].

In this paper we provide another, orthogonal generalization of modal logic that enjoys the same nice properties. We introduce UNFO, a fragment of FO in which *negation* is restricted to formulas having only one free variable. UNFO is incomparable in terms of expressive power to GFO. It generalizes modal logic, as well as other formalisms, such as conjunctive queries, that are not contained in GFO. We show that UNFO has the tree-like model property, is translatable into tree-like automata (in the sense described above), and has the finite model property. Hence UNFO, too, is robustly decidable.

We also introduce UNFP, which extends UNFO with least and greatest monadic fix-points, in the same way that the μ-calculus extends modal logic [32], and guarded fixpoint logic (GFP) extends GFO [28]. UNFP generalizes the μ-calculus but also monadic Datalog and remains incomparable with GFP. It still has the tree-like model property and can be translated into MSO, but it no longer has the finite model property. Nevertheless, we show that finite satisfiability for UNFP is decidable (note that the decidability of the analogous problem for GFP was only recently solved in [4]). More precisely, the satisfiability problem is 2ExpTime-complete, both for UNFO and for UNFP, both on arbitrary and finite structures.

We also study the model checking problem. In contrast with GFO, whose model checking problem is PTime-complete [9], we show that for UNFO it is complete for $\mathrm{P}^{\mathrm{NP}[O(\log^2 n)]}$, providing one of the few natural complete problems for that complexity class. For UNFP, model checking is hard for P^{NP} and contained in $\mathrm{NP}^{\mathrm{NP}} \cap \mathrm{coNP}^{\mathrm{NP}}$. The gap between the upper bound and the lower bound reflects a similar open problem for GFP and the μ-calculus where the model checking problem lies between PTime and $\mathrm{NP} \cap \mathrm{coNP}$ [9].

UNFO is not only computationally but also model-theoretically very well behaved. We characterize the expressive power of UNFO in terms of an appropriate notion of invariance, and we show that UNFO has Craig Interpolation as well as the Projective Beth Property. Note that Craig Interpolation fails for GFO [30]. On trees, UNFO and UNFP correspond to well-known existing formalisms.

Outline of the paper. In Section 2, we formally introduce UNFO and UNFP, and we review relevant background material on modal logics and computational complexity. In Section 3 we develop the model theory of UNFO and UNFP: we introduce an appropriate notion of bisimulations, we state a finite model property and a tree-like model property, we obtain model theoretic characterizations, and we prove Craig Interpolation and the Projective Beth Property for UNFO. In Section 4, we show that the satisfiability problem for UNFO, and for UNFP, is 2ExpTime-complete, both on arbitrary structures and on finite structures. In Section 5, we map out the complexity of the model checking problem, that is, the problem of evaluating a formula in a given finite structure. In Section 6, we study the expressive power of UNFO and UNFP on tree structures. We conclude in Section 7 with a comparison with other work, in particular on guarded negation logics.

This paper is the journal version of [17]. It contains new and simpler proofs for many of the results as well as new results, such as the characterization theorem in the finite case, cf. Theorem 3.8.

2. Preliminaries

We consider relational structures. A relational schema is a finite set of relation symbols fixing an arity to each relation. A *model*, or *structure*, M over a relational schema σ is a set $\mathrm{dom}(M)$, the *domain* of M, together with an interpretation R^M to each relation symbol R of σ as a relation over the domain of the arity given by the schema. A model is said to be *finite* if its domain is finite. We assume familiarity with first-order logic, FO, and least fixpoint logic, LFP, over relational structures. We use classical syntax and semantics for FO and LFP. In particular we write $M \models \phi(\overline{u})$ or $(M, \overline{u}) \models \phi(\overline{x})$ for the fact that the tuple $\overline{u}$ of elements of the model M makes the FO-formula, or LFP-formula, $\phi(\overline{x})$ true on M.

Given a structure M and a set $X \subseteq \mathrm{dom}(M)$ we denote by $M|X$ the substructure of M induced by X.

2.1. **UNFO and UNFP.** We define the *unary-negation fragment of first-order logic* (or UNFO, for short), as the fragment of FO given by the following grammar (where R is an arbitrary relation name from the underlying schema):

$$\phi ::= R(\overline{x}) \mid x = y \mid \phi \wedge \phi \mid \phi \vee \phi \mid \exists x \phi \mid \neg \phi(x)$$

where, in the last clause, ϕ has no free variables besides (possibly) x. Throughout this paper, we will keep using the notation $\phi(x)$ to indicate that a formula has at most one free variable. In other words, UNFO is the restriction of FO where negation is only allowed if the subformula has at most one free variable. In particular $x \neq y$ is not expressible in UNFO.

We say that a formula of UNFO is in UN-normal form if, in the syntax tree of the formula, every existential quantifier (except for the root of the syntax tree) is either directly below another existential quantification, or the subformula starting with that quantifier has at most one free variable. In other words, formulas in UN-normal form are existential positive formulas in prenex normal form where each atom is either a positive atom over the underlying schema or a possibly negated formula with at most one free variable in UN-normal form.

For instance the formula $\exists x \exists y (R(x, y) \wedge \exists z S(x, y, z))$ is not in UN-normal form. However the equivalent formula $\exists x \exists y \exists z \, (R(x, y) \wedge S(x, y, z))$ is in UN-normal form. Similarly the formula

$$\exists x (R(x) \wedge \exists y (R(y) \wedge \exists z (R(z) \wedge (\exists x S(x, y, z)))))$$

is not in UN-normal form, but each of the equivalent formulas

$$\exists x \exists y \exists z \exists x' R(x) \wedge R(y) \wedge R(z) \wedge S(x', y, z)$$

and

$$\exists x (R(x) \wedge \exists y \exists z \exists x (R(y) \wedge R(z) \wedge S(x, y, z)))$$

is.

Every formula of UNFO can be transformed into an equivalent formula in UN-normal form in linear time by "pulling out existential quantifiers" as soon as the corresponding subformula has more than one free variable, using the following two rewrite rules:

$$\phi \wedge \exists x \psi \equiv \exists x (\phi \wedge \psi) \text{ provided that } x \text{ does not occur free in } \phi$$
$$\phi \vee \exists x \psi \equiv \exists x (\phi \vee \psi) \text{ provided that } x \text{ does not occur free in } \phi$$

(together with safe renaming of variables where needed). For instance, starting with

$$\exists x(R(x) \wedge \exists y(R(y) \wedge \exists z(R(z) \wedge (\exists x S(x,y,z)))))$$

one could obtain:

$$\exists x(R(x) \wedge \exists y \exists z \exists x(R(y) \wedge R(z) \wedge S(x,y,z)))$$

Bringing a UNFO formula into UN-normal form may increase the number of variables occurring in the formula, because applying the above rewrite rules may require renaming bound variables. A formula of UNFO is said to be *of width k* if it can be put in UN-normal form using the above rules in such a way that the resulting formula uses at most k variables. The width of the above formula is therefore 3. We denote by UNFO^k the set of all UNFO formulas of width k.

In order to define *unary-negation fixpoint logic* (UNFP) we introduce extra unary predicates that will serve for computing unary fixpoints. We denote the unary predicates given by the relational schema using the letters $P, Q \ldots$ and the unary predicates serving for computing the fixpoints by $X, Y \ldots$. By $\mathrm{UNFO}(\overline{X})$ we mean UNFO defined over the schema extended with the unary predicates $\overline{X}$. In particular it allows formulas of the form $\neg\phi(x, \overline{X})$. UNFP is the extension of $\mathrm{UNFO}(\overline{X})$ by means of the following least fixpoint construction:

$$[\mathrm{LFP}_{X,x} \ \phi(X, \overline{X}, x)](y)$$

where X occurs only positively in ϕ. An analogous greatest fixed point operator is definable by dualization. Note that no first-order parameters (i.e., free variables in the body of ϕ other than x) are permitted.

Note that UNFP is a syntactic fragment of least fixpoint logic (LFP), i.e., the extension of full first-order logic with the least fixpoint operator. Therefore, we can simply refer to the literature on LFP for the semantics of these formulas (cf. for example [33]). However, we will discuss the semantics of the least fixpoint operator here in some detail, because our arguments later on will refer to it. Consider any UNFP formula of the form

$$[\mathrm{LFP}_{X,x} \ \phi(X, \overline{X}, x)](y)$$

and any structure $(M, \overline{S})$, where $\overline{S}$ is a collection of subsets of the domain of M that form the interpretation for $\overline{X}$. Since X occurs in ϕ only positively, $\phi(X, \overline{X}, x)](y)$ induces a monotone operation $\mathcal{O}_\phi$ on subsets of the domain of M, where $\mathcal{O}_\phi(A) = \{a \in \mathrm{dom}(M) \mid (M, \vec{S}, A) \models \phi(a)\}$. By the Knaster-Tarski fixpoint theorem, this monotone operation has a unique least-fixpoint. By definition, an element $b \in \mathrm{dom}(M)$ satisfies the formula $[\mathrm{LFP}_{X,x} \ \phi(X, \overline{X}, x)](y)$ in $(M, \vec{S})$ if and only if b belongs to this least fixpoint. The least fixpoint of the monotone operation $\mathcal{O}_\phi$ is known to be the intersection of all its pre-fixed points, i.e., $\bigcap\{A \subseteq \mathrm{dom}(M) \mid A \supseteq \mathcal{O}_\phi(A)\}$, and it can be equivalently characterized as $\mathcal{O}_\phi{}^\kappa(\emptyset)$, where $\kappa = |\mathrm{dom}(M)|$, $\mathcal{O}_\phi{}^0(\emptyset) = \emptyset$; for all successor ordinals $\lambda + 1$, $\mathcal{O}_\phi{}^{\lambda+1}(\emptyset) = \mathcal{O}_\phi(\mathcal{O}_\phi{}^\lambda(\emptyset))$; and for all limit ordinals $\lambda \leq \kappa$, $\mathcal{O}_\phi{}^\lambda(\emptyset) = \bigcup_{\lambda' < \lambda} \mathcal{O}_\phi{}^{\lambda'}(\emptyset)$.

The same definition of the UN-normal form applies to UNFP. As in the case of UNFO, we say that a UNFP formula *has width k* if, when put in UN-normal form, it uses at most k first-order variables. In other words, a formula of UNFP has width k if all the "$\mathrm{UNFO}(\overline{X})$-parts" of its subformulas have width k. We denote by UNFP^k the set of all UNFP formulas of width k.

The *negation depth* of a UNFO or UNFP formula will also be an important parameter. It is the maximal nesting depth of negations in its syntax tree.

Example 2.1. Two examples of UNFO formulas are $\exists yzu(R(x,y) \wedge R(y,z) \wedge R(z,u) \wedge R(u,x))$, which expresses the fact that x lies on a directed R-cycle of length 4, and its negation $\neg\exists yzu(R(x,y) \wedge R(y,z) \wedge R(z,u) \wedge R(u,x))$. It follows from known results [1] that neither can be expressed in the guarded fragment, and therefore, these examples show that UNFO can express properties that are not definable in the guarded fragment. On the other hand, we will see in Section 3.1 that the guarded-fragment formula $\forall xy(R(x,y) \to S(x,y))$ has no equivalent in UNFO, and therefore, the two logics are incomparable in expressive power.

A *conjunctive query* (CQ) is a query defined by a first-order formula of the form $\exists x_1 \cdots x_n \ \tau_1 \wedge \cdots \wedge \tau_l$, where each τ_i is a (positive) atomic formula. A *union of conjunctive queries* (UCQ) is a query defined by a finite disjunction of first-order formulas of the above form. Clearly, every UCQ is definable in UNFO. In fact, UNFO can naturally be viewed as the extension of the language of UCQs with unary negation. It is also worth noting that, in a similar way, all *monadic datalog* queries (i.e., datalog queries in which all IDB relations are unary [20]) are definable in UNFP. It was shown in [20] that query containment is decidable in 2ExpTime for monadic datalog. As the containment of two unary Datalog programs can be expressed in UNFP, the decidability of UNFP in 2ExpTime, cf. Theorem 4.5, generalizes this result. We also mention that the query containment problem was recently shown to be hard for 2Exptime [6], hence the lower bound of Theorem 4.5 also follows from this fact.

2.2. **Modal logic and bisimulation.** UNFO and UNFP can be viewed as extensions of modal logic and the μ-calculus, and, actually, of their *global two-way* extensions. As several of our proofs will make reductions to the modal logic case, we now review relevant definitions and results regarding *global two-way modal logic* and the *global two-way μ-calculus*.

We view a *Kripke structure* as a relational structure over a schema consisting of unary and binary relations. Modal logics are languages for describing properties of nodes in Kripke structures. Intuitively, modal formulas can navigate Kripke structures by traversing edges in a forward or backward direction.

We will use ML to denote the modal language with forward and backward modalities, and with the global modality, as defined by the following grammar.

$$\phi ::= P \mid \phi \wedge \phi \mid \neg\phi \mid \langle R \rangle \phi \mid \langle R^{-1} \rangle \phi \mid \mathbf{S}\phi$$

where P is a unary relation symbol (also called *proposition letter* in this setting), and R is a binary relation symbol (also called an *accessibility relation* in this context). Disjunction and the "box operators" $[R]$ and $[R^{-1}]$ are definable as duals of conjunction, $\langle R \rangle$ and $\langle R^{-1} \rangle$, respectively (for instance $[R]\phi$ is $\neg\langle R \rangle\neg\phi$).

The semantics of ML can be given via a translation into UNFO: For each ML formula ϕ we construct by induction, as explained in Figure 1, a UNFO formula $\phi^*(x)$ such that for each Kripke structure M and node a we have $M \models \phi^*(a)$ iff a has the property ϕ on M.

We refer to ML as *global two-way modal logic*, because it includes the global modal operator $\mathbf{S}$ and the inverse modal operators $\langle R^{-1} \rangle$ (and their duals). Traditionally, the basic modal logic is defined without those features and can only navigate by traversing an edge in the forward direction.

The *global two-way μ-calculus*, which we denote by ML_μ, is obtained by adding fixpoint variables and a least fixpoint operator to the language of ML: fixpoint variables are admitted as atomic formulas, and whenever ϕ is a formula of ML_μ in which a fixpoint variable X

$$
\begin{array}{rcl}
(P)^*(x) & = & P(x) \\
(\phi \wedge \psi)^*(x) & = & \phi^*(x) \wedge \psi^*(x) \\
(\neg\phi)^*(x) & = & \neg\phi^*(x) \\
(\langle R\rangle\phi)^*(x) & = & \exists y \; R(x,y) \wedge \phi^*(y) \\
(\langle R^{-1}\rangle\phi)^*(x) & = & \exists y \; R(y,x) \wedge \phi^*(y) \\
(\mathbf{S}\phi)^*(x) & = & \exists y \; \phi^*(y)
\end{array}
$$

Figure 1: Inductive translation of an ML-formula ϕ to an equivalent UNFO-formula $\phi^*(x)$

occurs only positively (under an even number of negations), then $\mu X\phi$ is again a valid formula of ML_μ, and it denotes the fixpoint of the monotone operation on sets defined by $\phi(X)$. An analogous greatest fixpoint operator is definable as the dual of the least fixpoint operator. Adding the rule

$$(\mu X\phi)^*(x) \;\; = \;\; [\mathrm{LFP}_{X,y} \; \phi^*(y)](x)$$

to the table of Figure 1 shows that ML_μ can be seen as a fragment of UNFP.

  We know from [21, 47] that the satisfiability problem for ML and for ML_μ, on arbitrary Kripke structures, is ExpTime-complete. Although ML has the finite model property [21], that is, every satisfiable ML-formula is satisfied in some finite Kripke structure, the same does not hold for ML_μ, and therefore the satisfiability problem for ML_μ on finite structures is not the same problem as the satisfiability problem for ML_μ on arbitrary structures. Nevertheless, it was shown in [11] that the satisfiability problem for ML_μ on finite Kripke structures is ExpTime-complete.

Theorem 2.2. [47, 11] Testing whether a formula of ML_μ is satisfiable is ExpTime-complete, both on arbitrary Kripke structures and on finite structures.

  We note that, while the two-way modal μ-calculus as defined in [47, 11] does not include the global modality $\mathbf{S}$, the results from [11] immediately extend to full ML_μ.

  Modal formulas are invariant for bisimulation [8]. Here, due to the backward modal operators and the global modal operator, we need global two-way bisimulations (see for example [38]). Given two Kripke structures M and N a *global two-way bisimulation* between M and N is a binary relation $Z \subseteq M \times N$ such that the following hold for every pair $(a,b) \in Z$ and every relation symbol R:

- $a \in P^M$ if and only if $b \in P^N$, for all unary relation symbols P.
- for every a' with $(a,a') \in R^M$ there is a b' such that $(b,b') \in R^N$ and $(a',b') \in Z$,
- for every b' with $(b,b') \in R^N$ there is an a' such that $(a,a') \in R^M$ and $(a',b') \in Z$,
- for every a' with $(a',a) \in R^M$ there is a b' such that $(b',b) \in R^N$ and $(a',b') \in Z$,
- for every b' with $(b',b) \in R^N$ there is an a' such that $(a',a) \in R^M$ and $(a',b') \in Z$,
- for every node a' of M there is a node b' of N such that $(a',b') \in Z$,
- for every node b' of N there is a node a' of M such that $(a',b') \in Z$,

We write $M \approx N$ if there is a global two-way bisimulation between M and N, and we write $(M,a) \approx (N,b)$ if the pair (a,b) belongs to a global two-way bisimulation between M and N. Recall that a homomorphism $h : M \to N$ is a map from the domain of M to the domain of N such that for all relation symbols R and tuples $(a_1,\ldots,a_n) \in R^M$, we have that $(h(a_1),\ldots,h(a_n)) \in R^N$. We say that M is a $\approx$-*cover* of N if there is a

homomorphism $h : M \to N$ such that $(M, a) \approx (N, h(a))$ for every element a of M. In addition, for $h(a') = a$, we say that (M, a') is a $\approx$-cover of (N, a).

One can equivalently view bisimulations as strategies for a player in a two-player game. In this game, the two players, called Abelard en Eloïse, maintain a pair (a, b) of elements, one in each structure. Intuitively, one can think of a pebble lying on each of these two selected nodes. At any time during the game, the pebbled nodes must satisfy the same unary predicates in the two structures. A move of Abelard consists of choosing one of the two pebbles (i.e., one of the two structures), and either sliding the pebble forward or backward along an edge belonging to some binary relation R, or moving it to an arbitrary position. Then Eloïse must respond in the other structure with a move mimicking Abelard's move, that is, either sliding her pebble along an edge belonging to the same relation R (and in the same direction), or moving it to an arbitrary position, depending on Abelard's move. If Eloïse cannot respond with a valid move, Abelard wins. It is easy to see that $(M, a) \approx (N, b)$ if and only if, starting in (a, b), Eloïse has a strategy that allows her to play forever without letting Abelard win. We also write $(M, a) \approx^l (N, b)$ if, starting in (a, b), Eloïse has a strategy that avoids losing in the first l rounds.

It is well known that ML_μ-formulas are invariant for global two-way bisimulations: if $(M, a) \approx (N, b)$ and if ϕ is a formula of ML_μ, then $(M, a) \models \phi$ if and only if $(N, b) \models \phi$. This basic fact of ML_μ has an important consequence: if a μ-calculus formula has a model, then it has a, possibly infinite, acyclic model, obtained by "unraveling" the original model along its paths while preserving bisimulation equivalence. If we restrict attention to finite Kripke structures, then, in general, acyclicity may not be achievable. For example, the one-element structure consisting in a self-loop is not bisimilar to any finite acyclic structure. However, over finite structures, a weaker form of acyclicity can be achieved. For a natural number l, a Kripke structure is called l-acyclic if its underlying graph contains no cycle of length less than l. We will make use of the following important result:

Theorem 2.3. [38] For all $l \in \mathbb{N}$, every finite Kripke structure has a finite l-acyclic $\approx$-cover.

This was used to show the following property, which is relevant for us as well. We write $(M, a) \equiv_{\mathrm{FO}^q} (N, b)$ if (M, a) and (N, b) satisfy the same first-order formulas of quantifier depth q.

Theorem 2.4. [38, Proposition 33] For each $q \in \mathbb{N}$ there is a $l \in \mathbb{N}$ such that whenever $(M, a) \approx^l (N, b)$, then (M, a) and (N, b) have $\approx$-covers (M', a') and (N', b'), respectively, such that $(M', a') \equiv_{\mathrm{FO}^q} (N', b')$. Moreover, if M and N are finite then M' and N' can be chosen to be finite as well.

2.3. **Overview of relevant oracle complexity classes.** We briefly review a number of complexity classes involving restricted access to an oracle, which turn out to be relevant for our present investigation, and may not be very well known. The reader interested to learn more about these classes would benefit from reading the literature cited below, as well as [42] and [22] that inspired us a lot.

The first class we use is denoted P^{NP}, also known as Δ_2^p. It consists of all problems that are computable by a Turing machine running in time polynomial in the size of its input, where the Turing machine, at any point during its computation, can ask yes/no queries to an NP oracle, and take the answers of the oracle into account in subsequent steps of the computation (including subsequent queries to the NP oracle). Analogously, one can

define the classes $\mathrm{NP}^{\mathrm{NP}}$ and $\mathrm{coNP}^{\mathrm{NP}}$, which are also known as Σ_2^p and Π_2^p, respectively. An example of a P^{NP}-complete problem is LEX(SAT), which takes as input a Boolean formula $\phi(x_1, \ldots x_n)$ and asks what is the value of x_n in the lexicographically maximal solution (where x_n is treated as the least significant bit in the ordering) [48].

A subclass of P^{NP} is $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$. It is defined in the same way as P^{NP}, except that the number of yes/no queries that can be asked to the NP oracle is bounded by $O(\log(n))$, where n is the size of the input. There is an equivalent characterization of this class, denoted $\mathrm{P}_{\|}^{\mathrm{NP}}$ in [13]. It consists of all problems computable using a Turing machine running in time polynomial in the size of its input, where the Turing machine can call the oracle only once (or a constant number of times, as this turns out not to make a difference), but in doing so it may ask the oracle several (polynomially many) yes/no questions in parallel [13]. In other words, the answer to a query cannot be used by the Turing machine in choosing which subsequent queries to ask to the oracle. A third equivalent characterization of $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ is as the class of problems that are PTime truth-table reducible to NP [13], that is, problems for which there is a PTime algorithm that, given an instance of the problem, produces a set $y_1, \cdots, y_n$ of inputs to some NP oracle, together with a Boolean formula $\phi(x_1, \cdots, x_n)$, such that the input is a yes-instance iff ϕ evaluates to true after replacing each x_i by 1 if y_i is accepted by the NP oracle and 0 otherwise. An example of a $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$-complete problem is the problem whether two graphs have the same chromatic number [48].

Finally, in between $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ and P^{NP} lies a hierarchy of classes $\mathrm{P}^{\mathrm{NP}[O(\log^i n)]}$ with $i > 1$. They are defined in the same way as $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ except that the number of queries to the oracle is bounded by $O(\log^i(n))$. Each class $\mathrm{P}^{\mathrm{NP}[O(\log^i n)]}$ can be equivalently characterized as the class of problems that can be solved in polynomial time allowing $O(\log^{i-1}(n))$ many rounds of parallel queries to an NP oracle [14].

There are few known natural complete problems for the classes $\mathrm{P}^{\mathrm{NP}[O(\log^i n)]}$. We introduce here a complete problem $\mathrm{LEX}_i(\mathrm{SAT})$ that we will make use of later on in our lower bound proofs. Recall that LEX(SAT) is the problem to decide, given a Boolean formula $\phi(x_1, \ldots, x_n)$, whether the value of x_n is 1 in the lexicographically maximal solution. Here, x_1 is treated as the most significant bit and x_n as the least significant bit in the ordering. Similarly, for $i \geq 1$, we define $\mathrm{LEX}_i(\mathrm{SAT})$ to be the problem of testing, given a Boolean formula $\phi(x_1, \ldots, x_n)$ and a number $k \leq \log^i(n)$, whether the value of x_k is 1 in the lexicographically maximal solution.

Theorem 2.5. $\mathrm{LEX}_i(\mathrm{SAT})$ is $\mathrm{P}^{\mathrm{NP}[O(\log^i n)]}$-complete.

Proof. The upper bound proof is immediate. In order to test whether the value of x_1 is 1 in the lexicographically maximal solution it is enough to ask the oracle whether the Boolean formula $\phi(1, x_2, \ldots, x_n)$ has a solution or not. Depending on the result we continue with $\phi(1, x_2, \ldots, x_n)$ or $\phi(0, x_2, \ldots, x_n)$ and with j calls to the oracle we learn this way the value of $x_1, \ldots, x_j$ in the lexicographically maximal solution.

The proof for the lower bound is a straightforward adaptation of the proof in [48] that LEX(SAT) is P^{NP}-complete. Let A be any problem in $\mathrm{P}^{\mathrm{NP}[O(\log^i n)]}$, let M be a deterministic polynomial-time Turing machine accepting A using an oracle for a problem $B \in \mathrm{NP}$, and let M' be a non-deterministic polynomial-time Turing machine accepting B. Let $f(n) = O(\log^i(n))$ be a function bounding the number of oracle queries asked by M on an input of size n.

Recall the textbook proof of NP-hardness of propositional satisfiability (cf., for example, [49]), which is based on efficiently constructing a Boolean formula describing runs of a given non-deterministic polynomial-time Turing machine. By the same construction, we can efficiently compute a Boolean formula

$$H^n(\overline{x}, \overline{u}, \overline{v}_1, \ldots, \overline{v}_{f(n)}, z_1, \ldots, z_{f(n)}, z)$$

(in fact, a conjunction of clauses with 3 literals per clause), parametrized by a natural number n, whose satisfying assignments describe the runs of M on input words of length n, where

(1) $\overline{x}$ describes an input word of length at most n
(2) $\overline{u}$ describes the sequence of configurations of M during the run (including tape content, head position and state in each configuration),
(3) $\overline{v}_j$ describes the j-th query asked to the oracle,
(4) z_j describes the answer of the j-th query asked to the oracle (where $z = 0$ means "no" and $z = 1$ means "yes"),
(5) z describes the result of the entire computation of M (where $z = 0$ means "reject" and $z = 1$ means "accept").

The formula $H^n(\overline{x}, \overline{u}, \overline{v}_1, \ldots, \overline{v}_{f(n)}, z_1, \ldots, z_{f(n)}, z)$ does *not* enforce that each z_j is the *correct* answer to the oracle query $\overline{v}_j$. Thus, the formula may have several satisfying assignments with the same values of $\overline{x}$ (one for each possible sequence of answers that the oracle may give).

In the same way, we can efficiently compute a Boolean formula

$$G^n(\overline{v}, \overline{y}, z')$$

(in fact, a conjunction of clauses with 3 literals per clause), whose satisfying assignments describe all (not necessarily accepting) runs of M' on input words of length n, where

(1) $\overline{v}$ describes an input word of length at most n
(2) $\overline{y}$ describes the sequence of configurations of M' during the run (including tape content, head position and state in each configuration),
(3) z' describes the result of the entire computation of M (where $z = 0$ means "reject" and $z = 1$ means "accept").

Since M' is non-deterministic, the formula $G^n(\overline{v}, \overline{y}, z')$ may have many satisfying assignments with the same values of $\overline{v}$.

Finally, for each input word w given as a bitstring of length n, we define ϕ_w to be the Boolean formula

$$\phi_w = H^n(w, \overline{u}, \overline{v}_1, \ldots, \overline{v}_{f(n)}, z_1, \ldots, z_{f(n)}, z) \wedge \bigwedge_{j=1\ldots f(n)} G^n(\overline{v}_j, \overline{y}_j, z_j)$$

Observe that ϕ_w only asserts that the z_j is the result of *some* run of the non-deterministic Turing machine M' on input $\overline{v}_j$. It does not require that $z_j = 1$ when M' has an accepting run on input $\overline{v}_j$. Consequently, not every satisfying assignment of ϕ_w describes the correct computation of M on input $\overline{x}$. However, it is easy to see that the lexicographically maximal satisfying assignment *does* describe the correct computation, due to the fact that it makes $z_j = 1$ whenever possible (given the already obtained values for z_ℓ for $\ell < j$). Thus, we have $z = 1$ in the lexicographically maximal solution of ϕ_w if and only if $w \in A$.

We order the variables in the formula ϕ_w so that $z_1, \ldots, z_{f(n)}, z$ come first (and in this order). By construction, z is then the $f(n)+$1st variable of ϕ_w. Let m be the total number of

variables occurring in ϕ_n (which is bounded by some polynomial in n). If $f(n) + 1 \leq \log^i m$, then (ϕ_w, m) is a valid input for the $\mathrm{LEX}_i(\mathrm{SAT})$ problem, and we are done. Otherwise, we extend ϕ_w with additional dummy variables, which serve no role other than making sure that $f(n) + 1 \leq \log^i m$. It is easy to see that this can always be done. $\square$

3. MODEL THEORY

In this section we give many key definitions, we show results about the expressive power of UNFO and UNFP, and we show that UNFO has Craig Interpolation and the Projective Beth Property.

3.1. **UN-bisimulations, the finite model property, and the tree-like model property.** We define a game that captures model indistinguishability, and we use it to characterize the expressive power of UNFO and UNFP. The game is as follows: the two players maintain a single pair (a, b) of elements from the two structures. A move of Abelard consists of choosing a set X of points in one of the two structures. Then Eloise responds with a homomorphism h from the set X into a set of points in the other structure, where the homomorphism maps a to b (respectively b to a) if a (respectively b) belongs to the set X. Finally, Abelard picks a pair $(u, h(u))$ (respectively $(h(u), u)$) and the players continue with that pair. The game is parametrized by the size of the sets chosen by Abelard in each round.

Equivalently, we can present the game in terms of a back-and-forth system:

Definition 3.1. Let M, N be two structures. A UN-bisimulation (resp. a UN-bisimulation of width $k \geq 1$) is a binary relation $Z \subseteq M \times N$ such that the following hold for every pair $(a, b) \in Z$:

- [**Forward property**] For every finite set $X \subseteq \mathrm{dom}(M)$ (resp. with $|X| \leq k$) there is a partial homomorphism $h : M \to N$ whose domain is X, such that $h(a) = b$ if $a \in X$, and such that every pair $(a', b') \in h$ belongs to Z.
- [**Backward property**] Likewise in the other direction, where $X \subseteq \mathrm{dom}(N)$.

We write $M \approx_{\mathrm{UN}} N$ if there is a non-empty UN-bisimulation between M and N, and we write $M \approx_{\mathrm{UN}^k} N$ if there is a non-empty UN-bisimulation of width k between M and N.

It is not difficult to see that the existence of a UN-bisimulation implies indistinguishability by UNFP sentences, and that the (weaker) existence of a UN-bisimulation of width k implies indistinguishability in UNFP^k.

Proposition 3.1. *For any $k \geq 1$, if $M \approx_{\mathrm{UN}^k} N$ then M and N satisfy the same sentences of UNFP^k. In particular, if $M \approx_{\mathrm{UN}} N$ then M and N satisfy the same sentences of UNFP.*

Proof. The second claim follows immediately from the first one, because $M \approx_{\mathrm{UN}} N$ implies $M \approx_{\mathrm{UN}^k} N$ for all $k \geq 1$.

The proof of the first claim is by induction on the nesting of fixpoints and existential quantification in the formula. We assume without loss of generality that all formulas are in UN-normal form. It is convenient to state the induction hypothesis for UNFO^k-formulas $\phi(x)$ in one free first-order variable and several free monadic second-order variables. The induction hypothesis then becomes: for all formulas $\phi(x, \overline{Y})$ of width k, for all UN-bisimulations Z of width k between $(M, \overline{P})$ and $(N, \overline{Q})$, and for all pairs $(a, b) \in Z$, we

have $(M, \overline{P}, a) \models \phi$ iff $(N, \overline{Q}, b) \models \phi$. We show only the important cases of the inductive step. Let M, N be two structures, Z be a UN-bisimulation of width k between M and N, $\overline{P}$ and $\overline{Q}$ be valuations of $\overline{Y}$ respectively on M and N, and $(a, b) \in Z$.

- $\phi(x, \overline{Y})$ starts with an existential quantifier. Then, by definition of UN-normal form, ϕ starts with a block of existential quantifications, followed by a Boolean combination of atomic formulas or formulas with at most one free first-order variable, i.e. again of the form $\psi(y, \overline{Y})$. Let $x_1, \ldots, x_n$ be the initial existentially quantified variables of ϕ. In particular $n \leq k$.

  First, suppose $(M, \overline{P}, a) \models \phi$. Let $X = \{a_1, \ldots, a_n\}$ be the quantified elements of M witnessing the truth of ϕ. By the definition of UN-bisimulation, there is a homomorphism $h : M|X \to N$ such that $h(a) = b$ (if a is in the domain of h) and such that $\{(a_i, h(a_i)) \mid i \leq n\} \subseteq Z$. By induction hypothesis, a subformula $\psi(y, \overline{Y})$ of ϕ is true on $(M, \overline{P}, a_i)$ iff it is true on $(N, \overline{Q}, h(a_i))$. Hence the assignment that sends $x_1, \ldots, x_n$ to $h(a_1), \ldots, h(a_n)$ makes ϕ true on $(N, \overline{Q}, b)$. The opposite direction, from $(N, \overline{Q}, b) \models \phi$ to $(M, \overline{P}, a) \models \phi$, is symmetric.

- $\phi(x, \overline{Y})$ is any Boolean combination of formulas of the form $\psi(y, \overline{Y})$, the result is immediate from the induction hypothesis.

- $\phi(x, \overline{Y})$ is of the form $[\mathrm{LFP}_{X,y} \, \psi(X, \overline{Y}, y)](x)$. We proceed by induction on the the fixpoint iterations. Let $\mathcal{O}_{\phi,(M,\overline{P})}$ and $\mathcal{O}_{\phi,(N,\overline{Q})}$ be be the monotone set-operations induced by ϕ on subsets of the domain of $(M, \overline{P})$ and $(N, \overline{Q})$, respectively, and let $\kappa = \max\{|M|, |N|\}$. Recall that the least fixpoint of $\mathcal{O}_{\phi,(M,\overline{P})}$ is equal to $\mathcal{O}_{\phi,(M,\overline{P})}{}^{\kappa}(\emptyset)$, and similarly for the least fixpoint of $\mathcal{O}_{\phi,(N,\overline{Q})}$. A straightforward transfinite induction shows that, for all ordinals λ, and for all $(a, b) \in Z$, $a \in \mathcal{O}_{\phi,(M,\overline{P})}{}^{\lambda}(\emptyset)$ if and only if $b \in \mathcal{O}_{\phi,(N,\overline{Q})}{}^{\lambda}(\emptyset)$. We conclude that $(M, \overline{P}, a) \models [\mathrm{LFP}_{X,y} \, \psi(X, \overline{Y}, y)](x)$ if and only if $(N, \overline{Q}, b) \models [\mathrm{LFP}_{X,y} \, \psi(X, \overline{Y}, y)](x)$. $\square$

Note that it is crucial, here, that we have defined width in terms of the UN-normal form. For example, if R is a binary relation, then the existence of a cyclic directed R-path of length k (i.e., a sequence of not necessarily distinct nodes $a_1, \ldots, a_k$ with $R(a_i, a_{i+1})$ and $R(a_k, a_1)$) can be expressed in UNFO using only 3 variables, by means of a careful reuse of variables, but the formula in question would not be in UN-normal form. Indeed, the existence of a cyclic directed R-path of length k, for $k > 3$, is not preserved by UN-bisimulations of width $k - 1$.

A similar invariance property holds for formulas with free variables. For simplicity, we only state a version of the result without reference to the width of formulas.

Definition 3.2. Let M and N be structures with the same signature. A *UN-homomorphism* $h : M \to N$ is a homomorphism with the property that $(M, a) \approx_{\mathrm{UN}} (N, h(a))$ for all $a \in \mathrm{dom}(M)$. We write $(M, \overline{a}) \to_{\mathrm{UN}} (N, \overline{b})$ if there is a UN-homomorphism $h : M \to N$ such that $h(\overline{a}) = \overline{b}$.

Proposition 3.2. *If $(M, \overline{a}) \to_{\mathrm{UN}} (N, \overline{b})$ and $M \models \phi(\overline{a})$ then $N \models \phi(\overline{b})$, for all UNFP-formulas $\phi(\overline{x})$.*

Proof. Follows from Proposition 3.1, together with the fact that positive existential formulas are preserved by homomorphisms (note that every UNFP-formula $\phi(\overline{x})$ can be viewed as a positive existential formula built from atomic formulas and from UNFP-formulas in one free variable). $\square$

From the invariance for UN-bisimulation it follows by a standard infinite unraveling argument that UNFP has the tree-like model property. A more involved partial unraveling, using back-edges in order to keep the structure finite, can also be used to show that UNFO has the finite model property. We only state the results without giving the details of these constructions, as it turns out that both results will follow from the material presented in Section 4.

Theorem 3.3. Every satisfiable UNFO formula has a finite model.

Theorem 3.4. Every satisfiable UNFP formula of width k has a model of tree-width $k-1$.

Note, that UNFP does *not* have the finite model property. This follows from the fact that UNFP contains the two-way μ-calculus which is known to lack the finite model property [44]. Indeed, if $\max(x)$ is shorthand for $\neg\exists y\ E(x,y)$, then the formula

$$\exists x \max(x) \vee \exists x \neg[\text{LFP}_{X,y}\ \neg\exists z (E(z,y) \wedge \neg X(z))](x)$$

expresses the property that either there exists a maximal element or there is an infinite backward path. This formula is therefore obviously false in the infinite structure $(\mathbb{N}, suc)$. However it holds on any finite structure as if a finite structure has no maximal elements, it must contain a cycle, and hence an infinite backward path. The negation of this sentence is satisfiable, by $(\mathbb{N}, suc)$, but has no finite model.

3.2. **Characterizations.** We have seen in Proposition 3.1 that UNFO sentences are first-order formulas that are preserved under $\approx_{\text{UN}}$-equivalence. It turns out that the converse is also true. Indeed, in the same way that bisimulation-invariance characterizes modal logic [8, 41] and guarded bisimulation-invariance characterizes the guarded fragment of FO [1, 39], we will see that $\approx_{\text{UN}}$-invariance characterizes UNFO. We show two variants of this result depending on whether we consider finite or infinite structures. It turns out that the proof for the finite case also works for the infinite case. However we give an independent proof for the infinite case as it is simpler and introduces techniques that will be useful later when considering Craig Interpolation.

We say that a FO sentence is $\approx_{UN}$-*invariant* if for all structures M and N such that $M \approx_{\text{UN}} N$, we have $M \models \phi$ iff $N \models \phi$. The notion of $\approx_{\text{UN}^k}$-invariance is defined similarly.

Theorem 3.5. A sentence of FO is equivalent to a formula of UNFO iff it is $\approx_{\text{UN}}$-invariant.

For all $k \geq 1$, a sentence of FO is equivalent to a formula of UNFO^k iff it is $\approx_{\text{UN}^k}$-invariant.

Before proving Theorem 3.5 we state and prove the following useful lemma. We write $(M, a) \equiv_{\text{UNFO}} (N, b)$ if (M, a) and (N, b) satisfy the same UNFO-formulas. We define $(M, a) \equiv_{\text{UNFO}^k} (N, b)$ similarly. This lemma makes use of the classical notion of ω-saturation. The actual definition is not needed here and the interested reader is referred to [29]. For our purpose it is enough to know the following two key properties.

(1) If a (possibly infinite) set of first-order formulas is satisfiable, then it is satisfied by a model that is ω-saturated.
(2) Let M be ω-saturated, let $a_1, \ldots, a_m \in \text{dom}(M)$, and let $T(x_1, \ldots, x_n)$ is an infinite set of first-order formulas with free variables $x_1, \ldots, x_n$ and using $a_1, \ldots, a_m$ as parameters. If every finite subset T' of T is realized in M (meaning that $(M, \bar{b}, \bar{a}) \models T(\bar{x})$ for some $\bar{b} = b_1, \ldots, b_n \in \text{dom}(M)$), then the entire set T is realized in M.

Lemma 3.6. For all ω-saturated structures M and N with elements a and b, respectively, the following hold.

(1) The relation $\{(a,b) \mid (M,a) \equiv_{\mathrm{UNFO}} (N,b)\}$ is a UN-bisimulation.

(2) The relation $\{(a,b) \mid (M,a) \equiv_{\mathrm{UNFO}^k} (N,b)\}$ is a UN-bisimulation of width k ($k \geq 1$).

Proof. We prove the second claim. The proof of the first claim is similar. Let $Z = \{(a,b) \mid (M,a) \equiv_{\mathrm{UNFO}^k} (N,b)\}$. We show that Z satisfies the forward property, the proof of the backward property is analogous.

Suppose $(c,d) \in Z$ and let $X \subseteq \mathrm{dom}(M)$ with $|X| \leq k$. We can distinguish two cases: either $c \in X$ or $c \notin X$. We will consider the first case (the second case is simpler). Thus, let $X = \{c, c_1, \ldots, c_n\}$ ($n < k$). Let $T[x_1, \ldots, x_n]$ be the set of all formulas $\phi(x, x_1, \ldots, x_n)$ that are positive Boolean combinations of atomic formulas or unary formulas of UNFO^k and that are true in $(M, c, c_1, \ldots, c_n)$. We view T as an n-type with one parameter.

Notice that by construction of T, for each finite subset T' of T the formula $\exists x_1 \ldots x_n(\bigwedge T')$ is in UNFO^k and is satisfied by (M, c). By hypothesis this formula is therefore also satisfied by (N, d). Since N is ω-saturated (and treating T as an n-type with parameter d), it follows that the entire set $T[x, x_1, \ldots, x_n]$ is realized in N under an assignment g that sends x to d. This implies that the function h sending c to d and c_i to $g(x_i)$ is a homomorphism such that, for all i, c_i and $h(c_i)$ satisfy the same formulas of UNFO^k and therefore $(c_i, h(c_i)) \in Z$ by definition of Z. $\square$

Proof of Theorem 3.5. One direction follows from Proposition 3.1. For the other direction, we only give the proof for the case of UNFO^k, the argument for full UNFO being identical.

Let ϕ be any $\approx_{\mathrm{UN}^k}$-invariant FO sentence. We want to show that ϕ is equivalent to a UNFO^k-sentence.

We first show that whenever two structures agree on all sentences of UNFO^k, they agree on ϕ. Suppose M and N satisfy the same sentences of UNFO^k. Without loss of generality we can assume that M and N are ω-saturated. Define $Z \subseteq M \times N$ as the set of all pairs (a,b) such that (M,a) and (N,b) satisfy the same UNFO^k-formulas. By Lemma 3.6, Z is a UN-bisimulation of width k. We claim that Z is non-empty. Let a be any element of M, and let $\Sigma(x)$ be the set of all UNFO^k-formulas with one free variable, true for a on M. Notice that for every finite subset Σ' of Σ, the formula $\exists x \bigwedge \Sigma'$ is a sentence of UNFO^k that is satisfied by M. Hence, by hypothesis, it is also satisfied in N. Therefore, by ω-saturation, the entire set $\Sigma(x)$ is realized by an element b in N, and hence $(a,b) \in Z$, which implies that Z is non-empty. This implies that $M \approx_{\mathrm{UN}^k} N$. Assume now that $M \models \phi$. As ϕ is $\approx_{\mathrm{UN}^k}$-invariant and $M \approx_{\mathrm{UN}^k} N$, this implies that $N \models \phi$. By symmetry we get $M \models \phi$ iff $N \models \phi$ as desired.

The rest of the proof is a well known argument using Compactness: If ϕ is not satisfiable then ϕ is equivalent to the UNFO^1 sentence "false". Otherwise let $M \models \phi$ and let Θ be the set of all UNFO^k sentences θ such that $M \models \theta$. We show that $\Theta \models \phi$ (i.e. any model of Θ is a model of ϕ). If this were not the case then we have a structure N such that $N \models \Theta \wedge \neg\phi$. But because Θ contains each UNFO^k sentence or its negation we have $M \approx_{\mathrm{UN}^k} N$ and M, N disagree on ϕ. This contradict the claim of the previous paragraph.

By compactness, there is a finite subset Θ' of Θ such that $\Theta' \models \phi$. By construction, this implies that ϕ is equivalent to the conjunction of all the sentences in Θ'. $\square$

Before we turn to the finite variant of Theorem 3.5, we remark that a similar characterization can be obtained for formulas with free variables, using UN-homomorphisms instead of UN-bisimulations.

Theorem 3.7. A formula of FO with free variables is equivalent to a formula of UNFO iff it is preserved under UN-homomorphisms.

Proof. One direction is provided by Proposition 3.2. For the other direction, let $\phi(\overline{x})$ be a FO formula preserved under UN-homomorphisms. As for the proof of Theorem 3.5, using a standard Compactness argument (cf. [18, Lemma 3.2.1]), it is enough to show that, for structures M, N with tuples $\overline{a}, \overline{b}$, if every UNFO-formula true in $(M, \overline{a})$ is true in $(N, \overline{b})$, then also $M \models \phi(\overline{a})$ implies $N \models \phi(\overline{b})$. Without loss of generality we may assume that M and N are ω-saturated. As ϕ is preserved under UN-homomorphisms, the result is now a direct consequence of the following claim.

> If M, N are ω-saturated structures, and $\overline{a}, \overline{b}$ tuples of elements such that every UNFO-formula true in $(M, \overline{a})$ is true in $(N, \overline{b})$, then there is a UN-homomorphism from an elementary substructure of $(M, \overline{a})$ to an elementary substructure of $(N, \overline{b})$ that maps $\overline{a}$ to $\overline{b}$.

In what follows, we prove the above claim. First of all, note that every equality statement satisfied in $\overline{a}$ is satisfied in $\overline{b}$, which makes it meaningful to speak about functions mapping $\overline{a}$ to $\overline{b}$.

We need the notion of a *potential homomorphism* from a structure M to a structure N. It is a non-empty collection $\mathcal{F}$ of finite partial homomorphisms $f : M \to N$, which satisfies the following extension property: for all $f \in \mathcal{F}$ and for all $a \in \text{dom}(M)$, there is an $f' \in \mathcal{F}$ which extends f and whose domain includes a. By a *potential UN-homomorphism* we will mean a potential homomorphism whose finite partial homomorphisms preserve the UN-bisimilarity type of each node. It is not hard to see that if M, N are countable structures and $\mathcal{F}$ is a potential homomorphism from M to N, then there is a homomorphism $h : M \to N$, which can be defined as the limit of a sequence of finite partial homomorphisms belonging to $\mathcal{F}$. In particular, if $\mathcal{F}$ is a potential UN-homomorphism, then h is a UN-homomorphism.

Now, let M and N be structures as described by the statement of the Lemma. A straightforward variation of the proof of Lemma 3.6 shows that there is a potential UN-homomorphism $\mathcal{F}$ from M to N mapping $\overline{a}$ to $\overline{b}$.

Next, we take the model pair $(M, N, \overline{a}, \overline{b})$ expanded with the maximal UN-bisimulation relation Z between M and N (i.e., the binary relation containing all pairs (a', b') such that $(M, a') \approx_{\text{UN}} (N, b')$), plus infinitely many additional relations that represent the potential UN-homomorphism $\mathcal{F}$ (for each $k \leq 1$, we use a new $2k$-ary relation R_k to represent all finite partial homomorphisms defined on k elements). We then apply to downward Löwenheim-Skolem theorem to obtain a similar situation $(M', N', \overline{a}, \overline{b}, \ldots)$, but where M' and N' are countable elementary substructures of M and N. Since we added the Z and R_k relations before applying the Löwenheim-Skolem theorem, we still have a potential UN-homomorphism from M' to N' mapping $\overline{a}$ to $\overline{b}$. It then follows by the earlier remark that there is a UN-homomorphism from M' to N' mapping $\overline{a}$ to $\overline{b}$. $\square$

Finally, we consider the case of finite structures. We say that a FO sentence is $\approx_{UN}$-*invariant on finite structures* if for all finite structures M and N such that $M \approx_{\text{UN}} N$, we have $M \models \phi$ iff $N \models \phi$. The notion of $\approx_{UN^k}$-*invariance on finite structure* is defined in the same way. We prove that UNFO^k is also the $\approx_{\text{UN}^k}$-invariant fragment of FO on finite

structures. The proof of this result constitutes, at the same time, also an alternative proof of the second part of Theorem 3.5. However it relies in a crucial way on the parameter k and therefore does not yield a characterization of UNFO in terms of $\approx_{\mathrm{UN}}$-invariance. In particular, it remains open whether UNFO is the $\approx_{\mathrm{UN}}$-invariant fragment of FO on finite structures. For simplicity, we state the result for formula with at most one free variable.

Theorem 3.8. Fix a finite schema σ, and let m be the maximal arity of the relations in σ. For all $k \geq m$, a formula of FO with at most one free variable is equivalent over finite σ-structures to a sentence of UNFO^k iff it is $\approx_{\mathrm{UN}^k}$-invariant on finite σ-structures.

Proof. Recall that we write $(M, a) \equiv_{\mathrm{FO}_q} (N, b)$ if M and N satisfy the same FO sentences of quantifier depth q. Recall the definition of UN-normal form, and observe that, in UNFO formulas that are in UN-normal form and that have at most one free variable, every existential quantifier must be either directly below another existential quantifier, or, otherwise, the subformula starting with that quantifier has at most one free variable. In the latter case, we call the existential quantifier in question a *leading existential quantifier*. We say that a formula of UNFO^k in UN-normal form with at most one free variable has *block depth* q if the nesting depth of its leading quantifiers is less than q. We denote by UNFO^k_q the fragment of UNFO^k consisting of formulas in UN-normal form with at most one free variable that have block depth q, and we will write $(M, a) \equiv_{\mathrm{UNFO}^k_q} (N, b)$ if M and N satisfy the same formulas of UNFO^k_q. One can show by a straightforward induction on k that, for every fixed finite schema, $\equiv_{\mathrm{UNFO}^k_q}$ is an equivalence relation of finite index (that is, there are only finitely many equivalence classes) and, consequently, each equivalence class can be described using a single UNFO^k_q-formula.

The basic proof strategy is as follows: we will show that for every $q \geq 0$ there exists a $l \geq 0$ such that whenever $(M, a) \equiv_{\mathrm{UNFO}^k_l} (N, b)$, then there exist structures (M^*, a^*) and (N^*, b^*) such that $(M, a) \approx_{\mathrm{UN}^k} (M^*, a^*)$, $(M^*, a^*) \equiv_{\mathrm{FO}^q} (N^*, b^*)$, and $(N^*, b^*) \approx_{\mathrm{UN}^k} (N, b)$. In other words, using the terminology of [38], "the equivalence relation $\equiv_{\mathrm{UNFO}^k_l}$ can be lifted to $\equiv_{\mathrm{FO}^q}$ modulo $\approx_{\mathrm{UN}^k}$". This implies the theorem: starting with a $\approx_{\mathrm{UN}^k}$-invariant FO formula ϕ in one free variable of quantifier depth q, we obtain that ϕ is $\equiv_{\mathrm{UNFO}^k_l}$-invariant. Therefore ϕ is equivalent to the disjunction of all (finitely many, up to equivalence) formulas describing an equivalence class of $\equiv_{\mathrm{UNFO}^k_l}$ containing a model of ϕ.

The construction of (M^*, a^*) and (N^*, b^*) makes use of Theorem 2.4. We first need to introduce some auxiliary definitions that will allow us to construct a Kripke structure from an arbitrary structure, and vice versa.

Let M be any structure over a signature σ. In what follows, by a *k-neighborhood* of M, we will mean a pair (K, h), where K is a structure with domain $\{1, \ldots, k\}$ over σ and h is a homomorphism from K to M. Intuitively, one can think of a k-neighborhood as a realization in M of some positive existential description of size k. From σ and k we construct the signature σ_k containing a unary predicate P_K for each structure K over σ with domain $\{1, \ldots, k\}$ and a binary relation R_i for each $1 \leq i \leq k$. Given a structure M over σ, we associate to it a structure G_M, called *the graph of* M, over σ_k, as follows. The nodes of G_M are the elements of M plus the k-neighborhoods of M. Each node of G_M that is a k-neighborhood (K, h) is labeled by the unary predicate P_K. There is an R_i-edge in G_M from a k-neighborhood (K, h) of M to an element $a \in M$ if and only if $h(i) = a$.

Conversely, we can transform a structure G over the signature σ_k into a structure $\widehat{G}$ over σ as follows. The universe of $\widehat{G}$ consists of the nodes of G that do not satisfy any unary

predicate. A tuple of such elements $\bar{a} = a_1 \ldots a_n$ (with $n \leq m \leq k$) belongs to R iff there is a node u of G satisfying P_K, for some K, and a tuple $\bar{b} = b_1 \ldots b_n$ of elements of K such that $K \models R(\bar{b})$ and such that for each $i \leq n$, G contains the edge $R_{b_i}(u, a_i)$.

We start by establishing a useful property linking M to its graph G_M.

Claim 3.3. For all structures (M, a) and (N, b) and for all $l \in \mathbb{N}$, if $(M, a) \equiv_{\mathrm{UNFO}_l^k} (N, b)$ then $(G_M, a) \approx^l (G_N, b)$.

Proof. Let Z_0 be the set of all pairs of elements (a, b) such that $(M, a) \equiv_{\mathrm{UNFO}_0^k} (N, b)$, and, for $\ell > 0$, let Z_ℓ be the set of all pairs of elements (a, b) such that $(M, a) \equiv_{\mathrm{UNFO}_\ell^k} (N, b)$, together with the set of all pairs $((K, h), (K, g))$ such that, for all $i \leq k$, $(M, h(i)) \equiv_{\mathrm{UNFO}_{\ell-1}^k} (N, g(i))$. We show, by induction on ℓ, that from a pair (u, v) in Z_ℓ, Eloïse can survive ℓ steps of the global two-way bisimulation game. Therefore, if (a, b) is such that $(M, a) \equiv_{\mathrm{UNFO}_l^k} (N, b)$, Z_l witnesses the fact that $(G_M, a) \approx^l (G_N, b)$ and the lemma is proved.

The base of the induction, where $\ell = 0$, is trivial. For the induction step, recall that Abelard can choose a structure, and subsequently, play one of three types of moves: moving the pebble forward along an edge, moving the pebble backward along an edge, and moving the pebble to an arbitrary node. We will assume that Abelard chooses G_M (the case where Abelard chooses G_N is symmetric) and that Abelard either (i) moves the pebble (forward or backward) along an edge, or (ii) moves the pebble to an arbitrary node. We treat the two cases separately.

As a convenient notation, for each structure K over σ with universe $\{1, \ldots, k\}$, we define $\alpha_K(x_1, \ldots, x_k)$ to be the quantifier-free formula that is the conjunction of all atomic formulas true in K, using free variables $x_1, \ldots, x_k$ for the elements $1, \ldots, k$. Moreover we say that τ is the UNFO_ℓ^k-type of an element a of M if it is the conjunction of all the UNFO_ℓ^k formula $\varphi(x)$ such that $M \models \varphi(a)$. Modulo logical equivalence there are only finitely many formulas in UNFO_ℓ^k hence each UNFO_ℓ^k-type is equivalent to a formula of UNFO_ℓ^k.

(i) Let $(u, v) \in Z_\ell$. By construction, (u, v) is either of the form (a, b) with $a \in \mathrm{dom}(M)$ and $b \in \mathrm{dom}(N)$, or it is of the form $((K, h), (K, g))$. First, suppose that $(u, v) = (a, b)$, where $a \in M$ and $b \in N$. Let u' be a node of G_M reachable from a by an edge. By construction of G_M, u' must be of the form (K, h) and is connected with an edge of label i to a, i.e. $h(i) = a$. For all i, let τ_i be the $\mathrm{UNFO}_{\ell-1}^k$-type of $h(i)$. By definition, the formula $\beta(x_i)$ defined as $\exists x_1 \ldots x_{i-1} x_{i+1} \ldots x_k \, \alpha_K \wedge \bigwedge_{j \neq i} \tau_j(x_j)$ is a formula of UNFO_ℓ^k and by construction we have $(M, a) \models \beta(x_i)$. Therefore $(N, b) \models \beta(x_i)$. Let g be the valuation corresponding to the initial block of existential quantifications. By construction g is a homomorphism from K to N and, for each i, $(h(i), g(i)) \in Z_{\ell-1}$. Hence $((K, h), (K, g)) \in Z_{\ell-1}$.

Next, suppose $(u, v) = ((K, h), (K, g))$ and a there is a node u' connected to u via an edge. By construction u' must be an element a of G_M connected with an edge of label i to (K, h), i.e. $h(i) = a$. By definition, for $b = g(i)$ we have $(a, b) \in Z_{\ell-1}$.

(ii) Let $(u, v) \in Z_\ell$ and let u' be any element of G_M. First, consider the case where u' is an element of M, and let $\tau(x)$ be its $\mathrm{UNFO}_{\ell-1}^k$-type. Since $(M, a) \equiv_{\mathrm{UNFO}_\ell^k} (N, b)$ and $(M, a) \models \exists x \, \tau(x)$, we have that $(N, b) \models \exists x \, \tau(x)$. Let v' be the witnessing element of N. Then $(N, u) \equiv_{\mathrm{UNFO}_{\ell-1}^k} (N, v)$, and hence, $(u', v') \in Z_{\ell-1}$.

Next, suppose that u' was of the form (K, h). For each $i \leq k$, let $\tau_i(x)$ be the $\mathrm{UNFO}_{\ell-1}^k$-type of $h(i)$. By definition, the formula β defined as $\exists x_1 \ldots x_k \, \alpha_K \wedge \bigwedge_i \tau_i(x_i)$ is a formula

of UNFO$_\ell^k$ and by construction we have $(M, a) \models \beta$. Therefore $(N, b) \models \beta$. Let g be the valuation corresponding to the initial block of existential quantifications in β. By construction g is a homomorphism from K to N and, for each i, $(h(i), g(i)) \in Z_{\ell-1}$. Hence $((K, h), (K, g)) \in Z_{\ell-1}$. $\qquad\square$

Fix some $q \in \mathbb{N}$ and let l be the number given by Theorem 2.4 for $q+1$. Let now (M, a) and (N, b) be such that $(M, a) \equiv_{\mathrm{UNFO}_l^k} (N, b)$. By Claim 3.3 we have $(G_M, a) \approx^l (G_N, b)$. By Theorem 2.4 there exists (G_M', a') and (G_N', b') such that $(G_M', a') \approx (G_M, a)$, $(G_N', b') \approx (G_N, b)$ and $(G_M', a) \equiv_{\mathrm{FO}^{q+1}} (G_N', b)$. Let M^* be $\widehat{G}_M'$ and N^* be $\widehat{G}_N'$. The following two claims show that (M^*, a') and (N^*, b') have the desired properties.

Claim 3.4. $(M^*, a') \equiv_{\mathrm{FO}^q} (N^*, b')$.

Proof. We lift the winning strategy in the Ehrenfeucht-Fraïssé game given by $(G_M', a') \equiv_{\mathrm{FO}^{q+1}} (G_N', b')$ to a winning strategy for the Ehrenfeucht-Fraïssé game $(M^*, a') \equiv_{\mathrm{FO}^q} (N^*, b')$: When, at stage i of the game, Spoiler chooses an element a_i in M^* Duplicator responds with the element b_i provided by his strategy in the game between G_M' and G_N', and vice versa if Spoiler chooses an element in N^*. We show that $M^* \models R(a_{i_1} \cdots a_{i_\kappa})$ implies $N^* \models R(b_{i_1} \cdots b_{i_\kappa})$. Let (K, h) be any k-neighborhood of M such that $R(a_{i_1}, \ldots, a_{i_\kappa})$ is the h-image of an atomic fact of K. Assuming the node $u = (K, h)$ is played in the game between G_M' and G_N', the extra move in the winning strategy of Duplicator guarantees the existence of $v = (K, g)$ in G_M' that is linked to $\bar{b}$ the same way u is linked to $\bar{a}$. This implies the desired property. $\qquad\square$

Claim 3.5. $(M^*, a') \approx_{\mathrm{UN}^k} (M, a)$ and $(N^*, b') \approx_{\mathrm{UN}^k} (N, b)$.

Proof. We only treat the case for M, the other being identical.

By Theorem 2.4, we have that (G_M', a') is a $\approx$-cover of (G_M, a). That is, there is a homomorphism $h : G_M' \to G_M$ such that $h(a') = a$ and such that $(G_M', c) \approx (G_M, h(c))$ for all $c \in \mathrm{dom}(G_M')$.

Let $Z = \{(c, h(c)) \mid c \in \mathrm{dom}(M^*)\}$. Note that, for all $c \in \mathrm{dom}(M^*)$ we have that $h(c) \in \mathrm{dom}(M)$, due to the fact that c and $h(c)$ agree on all unary predicates. We will show that, in fact, Z is a UN-bisimulation of width k between M^* and M. Note also that $(a', a) \in Z$.

We first prove that Z is the graph of a homomorphism. Let $\bar{a}$ be such that $M^* \models R(\bar{a})$. By construction of M^*, this implies that there is a node u in G_M' of label P_K and elements $b_1, \ldots, b_l$ in K such that $K \models R(b_1, \ldots, b_l)$ and $G_M' \models R_{b_j}(u, a_j)$ for all j. As h is a homomorphism, $h(u)$ has the same label as u and is connected to $h(\bar{a})$ the same way u is connected to $\bar{a}$. By construction of G_M, this implies $M \models R(h(\bar{a}))$.

Since we have just shown that Z is the graph of a homomorphism, it remains only to show that Z satisfies the backward property of the definition of UN-bisimulations.

Consider $(c, d) \in Z$, that is, $d = h(c)$. Let Y be a subset of elements of M of size less than k. We assume that $d \in Y$. The case where $d \notin Y$ is handled in the same way. Let K be the structure with universe $\{1, \ldots, k\}$ which is isomorphic to the restriction of M to Y and let g be the corresponding isomorphism. Without loss of generality we assume that $g(1) = b$. By definition of G_M there is a node $v = (K, g)$ that is connected to all the elements of Y via edges of appropriate label. Since $(G_M', c) \approx (G_M, d)$, we can find a node u, whose label correspond to K, connected to c via an edge of label 1, and such that $h(u) = v$. Since $(G_M', u) \approx (G_M, v)$, we can further find in G_M' a set of nodes X, connected

to u in the same way that Y are connected to v, such that $h(X) = Y$. It follows by the construction of M^* that the map from Y to X is a partial homomorphism from M to M^*. $\square$

This concludes the proof of Theorem 3.8: it follows from Claim 3.4 and Claim 3.5 that every $\approx_{\mathrm{UN}^k}$-invariant FO formula ϕ in one free variable of quantifier depth q, is $\equiv_{\mathrm{UNFO}_l^k}$-invariant for suitably large l, and, therefore (as explained earlier), is equivalent to a UNFO_l^k-formula. The analogous result for sentences follows immediately. $\square$

3.3. Craig Interpolation and Beth Property.

We conclude the list of nice model-theoretic properties of UNFO by showing that it has Craig Interpolation and the Projective Beth Property. In fact, we can show strong versions of these results, which take into account also the width of formulas. This is remarkable, given that both Craig Interpolation and the Beth Property fail for the k-variable fragment of first-order logic, for all $k > 1$. Moreover, the results presented in this section hold both on arbitrary structures and on finite structures. Model-theoretic proofs of Craig Interpolation typically use amalgamation constructions [18], and the proof we give here is essentially based on an amalgamation construction called *zigzag-products* that was introduced, in the context of modal logic, in [36].

For all UNFO-formulas $\phi(\overline{x}), \psi(\overline{x})$, we write $\phi \models \psi$ to express that the first-order formula $\forall \overline{x}(\phi \rightarrow \psi)$ (which is not necessarily a UNFO-formula) is valid, i.e. holds on all models.

It turns out that it is enough to consider finite models for testing whether $\phi \models \psi$. This is a consequence of the following remark and Theorem 3.3:

Remark 3.9. For all UNFO-formulas $\phi(\overline{x}), \psi(\overline{x})$, $\phi \models \psi$ holds (resp. holds on finite structures) if and only if the UNFO sentence

$$\exists \overline{x}(\phi \wedge \bigwedge_{1 \leq i \leq n} P_i(x_i)) \wedge \neg \exists \overline{x}(\psi \wedge \bigwedge_{1 \leq i \leq n} P_i(x_i))$$

is not satisfiable (resp. does not have a finite model), where $\overline{x} = x_1, \ldots, x_n$ and $P_1, \ldots, P_n$ are fresh unary predicates.

Remark 3.9 implies that all the results we prove for sentences apply to entailment between formulas with free variables as well. In particular, since UNFO has the finite model property by Theorem 3.3, we have that $\phi \models \psi$ holds on arbitrary structures if and only if holds on finite structures. Consequently, Craig Interpolation, and therefore also Beth Property, hold on arbitrary structures if and only if they hold over finite structures. In the remaining part of this section we only state the results for arbitrary structures, but, as we have just argued, they also hold over finite structures.

Theorem 3.10. UNFO^k has Craig Interpolation: for all $k \geq 1$ and for every pair of UNFO^k-formulas $\phi(\overline{x}), \psi(\overline{x})$ in the same free variables such that $\phi \models \psi$, there is a UNFO^k-formula $\chi(\overline{x})$ over the common vocabulary of ϕ and ψ such that $\phi \models \chi$ and $\chi \models \psi$.

Proof. The proof is by contradiction. Suppose that there is no interpolant in UNFO^k. We will show that (the first-order formula) $\phi \wedge \neg \psi$ is satisfiable.

Let σ be the vocabulary of ϕ, and τ the vocabulary of ψ, and let $\overline{x} = x_1 \ldots x_n$. By $(M, \overline{a}) \equiv_{\mathrm{UNFO}^k}^{\sigma} (N, \overline{b})$ we will mean that the tuple $\overline{a}$ in M and the tuple $\overline{b}$ in N satisfy the same UNFO^k-formulas using only relation symbols in σ, and by $(M, \overline{a}) \Rightarrow_{\mathrm{UNFO}^k}^{\sigma} (N, \overline{b})$ we

will mean that every UNFO^k-formula using only relation symbols in σ that is satisfied by the tuple $\bar{a}$ of M is satisfied also by the tuple $\bar{b}$ of N.

Claim 3.6. There are $(M, \bar{a}) \models \phi$ and $(N, \bar{b}) \models \neg\psi$ such that $(M, \bar{a}) \Rrightarrow^{\sigma \cap \tau}_{\mathrm{UNFO}^k} (N, \bar{b})$.

Proof of Claim 3.6: A classic argument involving two applications of Compactness (cf. [18, Lemma 3.2.1]). Let $\Phi(\bar{x})$ be the set of all UNFO^k-formulas in the joint vocabulary that are valid consequences of $\phi(\bar{x})$. The first-order theory $\Phi(\bar{x}) \cup \{\neg\psi(\bar{x})\}$ is consistent, because, if this were not the case, then by Compactness, some finite conjunction of formulas in Φ would be an interpolant, which we have assumed is not the case. Hence, let $(N, \bar{b}) \models \Phi(\bar{x}) \cup \{\neg\psi(\bar{x})\}$.

Next, let $\Gamma(\bar{x})$ be the set of all UNFO^k-formulas in the joint vocabulary that are *false* in $(N, \bar{b})$. Then the first-order theory $\{\neg\gamma \mid \gamma \in \Gamma(\bar{x})\} \cup \{\phi(\bar{x})\}$ is consistent, for, if it were not, then by Compactness, there would be $\gamma_1, \ldots, \gamma_m \in \Gamma$ such that $\bigwedge_i \neg\gamma_i \models \neg\phi(\bar{x})$ and hence $\bigvee_i \gamma_i \in \Phi(\bar{x})$, which contradicts $(N, \bar{b}) \models \Phi(\bar{x})$. Let $(M, \bar{a}) \models \{\neg\gamma \mid \gamma \in \Gamma(\bar{x})\} \cup \{\phi(\bar{x})\}$. By construction, all UNFO^k formulas in $\sigma \cap \tau$ true of $\bar{a}$ in M are true of $\bar{b}$ in N. In other words, $(M, \bar{a}) \Rrightarrow^{\sigma \cap \tau}_{\mathrm{UNFO}^k} (N, \bar{b})$. □

The following Claim 3.7 is a strengthening of Claim 3.6. Recall the definition of UN-homomorphisms. Just as we parametrized UN-bisimulations by a width k and a signature, we can parametrize UN-homomorphisms by a width and a signature. We write $(M, x) \approx^{\sigma \cap \tau}_{\mathrm{UN}^k} (N, y)$ if there is a UN-bisimulation of width k, with respect to the signature $\sigma \cap \tau$, between (M, x) and (N, y). We write $(M, \bar{a}) \to^{\sigma \cap \tau}_{\mathrm{UN}^k} (N, \bar{b})$ if, for every set $X \subseteq M$ with $|X| \leq k$, there is a partial homomorphism (with respect to the signature $\sigma \cap \tau$) $h : M \to N$ with domain X, such that (i) $h(a_i) = b_i$ for all $a_i \in \bar{a} \cap X$, and (ii) $(M, x) \approx^{\sigma \cap \tau}_{\mathrm{UN}^k} (N, h(x))$ for all $x \in X$. In particular this implies that $(M, \bar{a}) \Rrightarrow^{\sigma \cap \tau}_{\mathrm{UNFO}^k} (N, \bar{b})$.

Claim 3.7. There are $(M, \bar{a}) \models \phi$ and $(N, \bar{b}) \models \neg\psi$ such that $(M, \bar{a}) \to^{\sigma \cap \tau}_{\mathrm{UN}^k} (N, \bar{b})$.

Proof of Claim 3.7: We may assume without loss of generality that the models M and N provided by Claim 3.6 are ω-saturated, and therefore, by Lemma 3.6 we have that, whenever $(M, a) \equiv^{\sigma \cap \tau}_{\mathrm{UNFO}^k} (N, b)$, then $(M, a) \approx^{\sigma \cap \tau}_{\mathrm{UN}^k} (N, b)$.

Consider now a set $X = \{u_1, \ldots, u_\kappa\} \subseteq M$ with $\kappa \leq k$. We assume that X contains no element of $\bar{a}$. If this were not the case we simply remove those elements and proceed as below. Let $Y = X \cup \bar{a}$. We view Y as the sequence $(v_1, \ldots, v_\ell)$ where for $j \leq \kappa$, $v_j = u_j$ and for $j > \kappa$, $v_j = a_j$. For each element u_i of X, let T_i be the set of UNFO^k formulas $\psi(x)$ such that $M \models \psi(u_i)$. Let α be the conjunction of all atoms $R(x_{i_1}, \ldots, x_{i_l})$ such that $M \models R(v_{i_1}, \ldots, v_{i_l})$. For each finite subset T_i' of T_i, the formula $\exists x_1, \ldots x_\kappa \, \alpha \wedge \bigwedge_i T_i'(x_i)$ is in UNFO^k and is satisfied by $(M, \bar{a})$. Because $(M, \bar{a}) \Rrightarrow^{\sigma \cap \tau}_{\mathrm{UNFO}^k} (N, \bar{b})$ it is also satisfied by $(N, \bar{b})$. By ω-saturation, this implies that $\{\alpha(x_1, \ldots, x_i)\} \cup \bigcup_{i \leq \kappa} T_i(x_i)$ is realized in $(N, \bar{b})$. The witnessing tuple $\bar{c}$ provides the desired homomorphism from X to N.

Altogether this shows that $(M, \bar{a}) \to^{\sigma \cap \tau}_{\mathrm{UN}^k} (N, \bar{b})$. □

We now construct a new $\sigma \cup \tau$-structure K out of the σ-structure M and the τ-structure N provided by Claim 3.7. This new structure K will contain a tuple satisfying $\phi \wedge \neg\psi$. Essentially, K will be the substructure of the cartesian product of M and N containing pairs of elements that are UN-bisimilar in the joint language. Relations in the joint vocabulary are interpreted as usual in cartesian products, while relations outside of the joint vocabulary are copied from the respective structure. The precise definition of K is as follows:

- The domain of K is the set of all pairs $(a, b) \in M \times N$ such that $(M, a) \approx^{\sigma \cap \tau}_{\mathrm{UN}^k} (N, b)$.
- For each $R \in \sigma \cap \tau$ of arity m, R^K contains all tuples $(\langle a_1, b_1 \rangle, \ldots, \langle a_m, b_m \rangle)$ in the domain of K such that $(a_1, \ldots, a_m) \in R^M$ and $(b_1, \ldots, b_m) \in R^N$.
- For each $R \in \sigma \setminus \tau$ of arity m R^K contains all tuples $(\langle a_1, b_1 \rangle, \ldots, \langle a_m, b_m \rangle)$ in the domain of K such that $(a_1, \ldots, a_m) \in R^M$.
- Analogously for all relation symbols $R \in \tau \setminus \sigma$.

Claim 3.8. For all elements $\langle a, b \rangle$ of K, we have

$$(K, \langle a, b \rangle) \approx^{\sigma}_{\mathrm{UN}^k} (M, a) \quad \text{and} \quad (K, \langle a, b \rangle) \approx^{\tau}_{\mathrm{UN}^k} (N, b) .$$

Proof of Claim 3.8: We will show that $(K, \langle a, b \rangle) \approx^{\sigma}_{\mathrm{UN}^k} (M, a)$. The proof of the other half of the claim is analogous.

Let Z be the graph of the natural projection from K onto M, i.e., $Z = \{(\langle u, v \rangle, u) \mid \langle u, v \rangle \in K\}$. Clearly, $(\langle a, b \rangle, a) \in Z$. Therefore, it suffices to show that Z is a UN-bisimulation of width k for the signature σ.

Consider any pair $(\langle u, v \rangle, u) \in Z$, and let X be any subset of the domain of K, with $|X| \leq k$. Let h be the natural projection from K onto M, restricted to X. It is clear from the definition of K that h is a partial homomorphism with respect to all relations in σ (those that belong to $\sigma \cap \tau$ as well as those that belong to $\sigma \setminus \tau$). Furthermore, it is clear that $h(\langle a, b \rangle) = a$ if $\langle a, b \rangle \in X$. Therefore, the forward property of the definition of UN-bisimulations is satisfied.

Next, consider any pair $(\langle u, v \rangle, u) \in Z$, and let X be any subset of the domain of M, with $|X| \leq k$. Since $(M, u) \approx^{\sigma \cap \tau}_{\mathrm{UN}^k} (N, v)$, there is a partial homomorphism $f : M \to N$ whose domain is X, such that $(M, x) \approx^{\sigma \cap \tau}_{\mathrm{UN}^k} (N, h(x))$ for all $x \in X$, and such that $h(u) = v$ if $u \in X$. Now, define $h : M \to K$ to be the map that sends every $x \in X$ to $\langle x, f(x) \rangle$. Clearly, this is a well-defined partial map from M to K, with domain X, having the property that $h(u) = (u, v)$ if $u \in X$. Therefore, it only remains to show that h is a partial *homomorphism*. That h preserves all relations in $\sigma \setminus \tau$ is immediate from the definition of K. That h preserves all relations in $\sigma \cap \tau$ follows from the construction of h and of K: if $(x_1, \ldots, x_m) \in R^M$, then, since f is a partial homomorphism, we have that $(f(x_1), \ldots, f(x_m)) \in R^N$. Hence, by the definition of K, we have that $(h(x_1), \ldots, h(x_m)) \in R^K$. $\square$

We will use the notation $\langle \bar{a}, \bar{b} \rangle$ as a convenient shorthand for $(\langle a_1, b_1 \rangle, \ldots, \langle a_n, b_n \rangle)$.

Claim 3.9. $(K, \langle \bar{a}, \bar{b} \rangle) \models \phi \wedge \neg \psi$

Proof of Claim 3.9: We may assume that ϕ is in UN-normal form. Let ϕ be of the form $\exists \bar{z} \phi'$, where ϕ' is built up from atomic formulas, equalities and formulas in one free variable, using conjunction and disjunction. Since $(M, \bar{a}) \models \phi$, there is a tuple $\bar{a}'$ witnessing $(M, \bar{a}') \models \phi'$ (where $\bar{a}$ and $\bar{a}'$ are appropriately related, depending on which variables are quantified in ϕ). The fact that $(M, \bar{a}) \to^{\sigma \cap \tau}_{\mathrm{UN}^k} (N, \bar{b})$ gives us that there is a partial homomorphism $h : M \to N$ whose domain is $\bar{a}'$, such that $h(a_i) = b_i$ for all $a_i \in \bar{a} \cap \bar{a}'$, and such that $(M, x) \approx^{\sigma \cap \tau}_{\mathrm{UN}^k} (N, h(x))$ for all $x \in \bar{a}'$. Now, let us use $\langle \bar{a}', \bar{b}' \rangle$ again as a convenient shorthand for the pairwise product of $\bar{a}'$ and $\bar{b}'$. Then it follows from the definition of K that all atomic formulas in signature τ that are true of $\bar{a}'$ in M are true of $\langle \bar{a}', \bar{b}' \rangle$ in K (for relations in $\tau \setminus \sigma$, this is immediate, and for relations in $\sigma \cap \tau$, this follows from the fact that the atomic formula in question is true also of $\bar{b}'$ in N). Hence (by induction on the conjunctions and

disjunction in ϕ'), we have that $(K, \langle \overline{a}', \overline{b}' \rangle) \models \phi'$. In fact, it can be seen that $\langle \overline{a}', \overline{b}' \rangle$ provides a witness showing that $(K, \langle \overline{a}, \overline{b} \rangle) \models \phi$.

Next, consider ψ. We may again assume that ψ is in UN-normal form. Recall that the definition of K implies that the natural projections from K onto N is a homomorphism preserving all relations in τ (both those in $\sigma \cap \tau$ and those in $\tau \setminus \sigma$). It follows by a straightforward induction that if $(K, \langle \overline{a}, \overline{b} \rangle) \models \psi$, then also $(N, \overline{a}) \models \psi$, which is not the case. $\qquad\square$

To summarize: on the basis of the assumption that there is no Craig interpolant for ϕ and ψ, we were able to show that $\phi \wedge \neg\psi$ is satisfiable, and hence, $\phi \to \psi$ is not a valid implication. This concludes the proof of our Craig interpolation theorem. $\qquad\square$

As usual, Craig Interpolation implies Beth Property. Let Σ be a UNFO-theory in a signature σ and let $R \in \sigma$ and $\tau \subseteq \sigma$. We say that Σ *implicitly defines* R in terms of τ if for all τ-structures M and for all σ-expansions M_1, M_2 of M satisfying Σ, we have that $R^{M_1} = R^{M_2}$. We say that a formula $\phi(\overline{x})$ in signature τ is an *explicit definition* of R relative to Σ if $\Sigma \models \forall \overline{x} \, (R\overline{x} \leftrightarrow \phi(\overline{x}))$. Note that the formula $\forall \overline{x} \, (R\overline{x} \leftrightarrow \phi(\overline{x}))$ is itself not necessarily a UNFO-formula, but this is irrelevant.

Theorem 3.11. UNFO has the Projective Beth Property: whenever a UNFO-theory Σ in a signature σ implicitly defines a k-relation R in terms of a signature $\tau \subseteq \sigma$, then there is a UNFO-formula in signature τ that is an explicit definition of R relative to Σ. Moreover, if Σ belongs to UNFO^k ($k \geq 1$), then the explicit definition can be found in UNFO^k as well.

Proof. The argument is standard from Craig Interpolation. We give it here for the sake of completeness. Suppose Σ implicitly defines R in terms of τ. We may assume $R \notin \tau$ because otherwise it is trivial. Furthermore, by Compactness, we may assume Σ to be finite. Let Σ' be a copy of Σ in which all relations S outside of τ (including R) have been replaced by new disjoint copies S'. Then the fact that Σ implicitly defines R in terms of τ implies that following first-order implication is valid:

$$\bigwedge \Sigma \wedge \bigwedge \Sigma' \models \forall \overline{x}(R\overline{x} \to R'\overline{x})$$

We can rewrite this into an equivalent implication of UNFO-formulas:

$$\bigwedge \Sigma \wedge R\overline{x} \models \bigwedge \Sigma' \to R\overline{x}$$

Let $\chi(\overline{x})$ be the interpolant given by Theorem 3.10 for this UNFO-implication. It is now straightforward to verify that χ is indeed an explicit definition of R relative to Σ. $\qquad\square$

4. Satisfiability

In this section, we show that the satisfiability problem for UNFP and for UNFO is 2ExpTime-complete, both on arbitrary structures and on finite structures. The lower bound holds already over structures with relations of bounded arity, and, in particular, over finite trees. Note that this is in contrast with GFO whose complexity drops from 2ExpTime-complete to ExpTime-complete when the arity of relations is bounded [27]. The upper bound is obtained by a reduction to the two-way modal μ-calculus and Theorem 2.2. Given a formula φ of UNFP we construct in exponential time a formula φ^* in the μ-calculus such that φ has a (finite) model iff φ^* has a (finite) model. The correctness of the construction in the

finite case builds on Theorem 2.3. The same reduction to the two-way modal μ-calculus allows us to prove the finite model property of UNFO and the tree-like model property of UNFP, i.e., Theorem 3.3 and Theorem 3.4.

We describe the reduction from φ to φ^* in two parts. In the first one we consider only a special case of UNFP formulas that we call *simple*. Those are, intuitively, formulas of the global two-way μ-calculus with navigation through arbitrary relations instead of just binary relations. The construction of φ^* is then polynomial. In a second part we show how the general case reduces to this one (with an exponential blow-up).

4.1. **Simple UNFP formulas.** We first consider a fragment of UNFP, which we call *simple* and denote it by sUNFP. It is a common fragment of UNFP and GFP, which embeds the global two-way μ-calculus. The syntax of sUNFP is given by the following grammar (recall that we use the notation $\phi(x)$ to indicate that a formula has no free first-order variables besides possibly x, but may contain some monadic second-order free variables):

$$\phi(x) ::= P(x) \mid X(x) \mid \phi(x) \wedge \phi(x) \mid \phi(x) \vee \phi(x) \mid \neg\phi(x) \mid [\mathrm{LFP}_{X,y}\phi(y)](x) \mid$$

$$\exists y_1 \ldots y_n (R(y_1 \ldots y_n) \wedge y_i = x \wedge \bigwedge_{j \in \{1 \ldots n\} \setminus \{i\}} \phi(y_j)) \mid \exists x \phi(x)$$

Note that all formulas generated by this inductive definition have at most one free (first-order) variable. We denote by sUNFO the first-order (that is, fixpoint-free) fragment of sUNFP.

We need the following notions. A *fact* of a structure M is an expression $R(a_1, \ldots, a_n)$ where $(a_1, \ldots, a_n) \in R^M$. The *incidence graph* $inc(M)$ of a structure M is the bi-partite graph containing facts of M and elements of M, and with an edge between a fact and an element if the element occurs in the fact. We say that a structure M is l-acyclic, for $l \geq 1$, if (i) $inc(M)$ has no cycle of length less than $2l$, and (ii) no element of M occurs twice in the same fact. We call a structure acyclic, if it is l-acyclic for all l (i.e., the incidence graph is acyclic and no element occurs in the same fact twice).

Based on a simple coding of relations of arbitrary arity using binary relations we can transform a simple formula into a formula of the μ-calculus and, using the results of Section 2.2, obtain:

Proposition 4.1.
(1) *The satisfiability problem for* sUNFP *is* ExpTime-*complete, both on arbitrary structures and on finite structures.*
(2) *If a* sUNFP *formula has a model, it has an acyclic model*
(3) *If a* sUNFP *formula has a finite model, then it has a l-acyclic finite model, $\forall l \geq 1$.*
(4) sUNFO *has the finite model property.*

Proof. (1) The ExpTime lower bound for sUNFP follows immediately from the fact that sUNFP subsumes the two-way modal μ-calculus ML_μ whose satisfiability problem is Exp-Time-hard (cf. Theorem 2.2). For the ExpTime upper bounds, we give a polynomial time translation from sUNFP to ML_μ, which preserves (un)satisfiability on arbitrary structures and on finite structures.

Let ϕ be any sUNFP formula. Let l be the maximal arity of the relation symbols occurring in ϕ, and let $p_1, \ldots, p_l$ be fresh proposition letters. For each unary predicate P, we introduce a fresh proposition letter p_P. Furthermore, we associate to each relation

symbol R a proposition letter p_R. Intuitively, the idea of the translation is that, in our Kripke models, we will encode an R-tuple by a "gadget" consisting of a node satisfying p_R that has n successors (where n is the arity of the relation R) satisfying p_1, ..., p_n, respectively, each of which has as a successor the corresponding element of the R-tuple. This is spelled out below in more detail. The syntactic translation $[\cdot]^*$ from sUNFP to ML_μ is then as follows:

$$[P(x)]^* = p_P$$
$$[X(x)]^* = X$$
$$[\phi(x) \wedge \psi(x)]^* = [\phi(x)]^* \wedge [\psi(x)]^*$$
$$[\phi(x) \vee \psi(x)]^* = [\phi(x)]^* \vee [\psi(x)]^*$$
$$[\neg\phi(x)]^* = \neg[\phi(x)]^*$$
$$[[\mathrm{LFP}_{X,y}\phi(y)](x)]^* = \mu X \ [\phi(y)]^*$$
$$[\exists y_1 \ldots y_n (R(y_1 \ldots y_n) \wedge y_i = x \wedge \bigwedge_{j \neq i} \phi_j(y_j))]^* = \Diamond^-(p_i \wedge \Diamond^-(p_R \wedge \bigwedge_{j \neq i} \Diamond(p_j \wedge \Diamond[\phi_j(y_j)]^*)))$$
$$[\exists x \phi(x)]^* = \mathbf{S}[\phi(x)]^*$$

For any structure M, we denote by M^* the Kripke model obtained as follows: every atomic fact $R(a_1 \ldots a_n)$ of M gets replaced by a substructure consisting of a node labeled p_R which has n children, labeled $p_1, \ldots p_n$, each of which has as its single child a_i. Then it is clear from the construction that $(M, a) \models \phi(x)$ if and only if $(M^*, a) \models [\phi(x)]^*$. Conversely, for every Kripke model M, we define a structure M_* as follows: the elements of M_* are the elements of M. A fact $R(a_1, \ldots, a_n)$ is inserted in M_* whenever in M there is a node labeled p_R and n (not necessarily distinct) children of p_R, labeled $p_1, \ldots p_n$, each of which has a_i as a child.

It is clear from the construction that $(M, a) \models [\phi(x)]^*$ if and only if $(M_*, a) \models \phi(x)$.

Altogether this shows that the translation $[\cdot]^*$ preserves (un)satisfiability, both on arbitrary structures and on finite structures.

(3) If ϕ has a finite model then by the construction above $[\phi]^*$ has a finite Kripke model. By Theorem 2.3 this implies that $[\phi]^*$ has a model that is $4l$-acyclic. Now notice that the transformation of M into M_* described above preserves l-acyclicity up to a factor of 4, and therefore, ϕ has a model that is is l-acyclic.

(2) is proved in the same way as (3), using the fact that if $[\phi]^*$ has a model then it has an acyclic one and that the transformation of M into M_* preserves acyclicity.

(4) For a sUNFO formula ϕ, $[\phi]^*$ does not contain any fixpoint and is therefore in ML. Recall from Section 2.2 that ML has the finite model property. This implies that UNFO also has the finite model property as the transformation of M into M_* preserves finiteness. $\square$

4.2. **Arbitrary UNFP-formulas.** We now attack the satisfiability problem for arbitrary UNFP formulas. We say that a disjunction $\psi_1 \vee \psi_2$ is unary if $\psi_1 \vee \psi_2$ has at most one free variable. We remind the reader that we use the notation $\psi(y)$ to express that ψ has *at most* one free first-order variable y.

Lemma 4.1. Every UNFP sentence is equivalent to a UNFP sentence in UN-normal form that uses only unary disjunction, and, more precisely, a sentence generated by the following

grammar:

$$\chi(y) ::= \exists \overline{z} \ \psi(y, \overline{z}) \mid \neg\chi(y) \mid \chi_1(y) \vee \chi_2(y) \mid [\text{LFP}_{X,z} \ \chi(z)](y) \tag{4.1}$$

where $\psi(y, \overline{z})$ is of the form:

$$\exists \overline{z}\big(\tau(\overline{z}) \wedge z_i = y \wedge \bigwedge_{j \in \{1 \ldots n\} \setminus \{i\}} \phi_j(z_j)\big) \qquad \text{or} \qquad \exists \overline{z}\big(\tau(\overline{z}) \wedge \bigwedge_{j \in \{1 \ldots n\}} \phi_j(z_j)\big) \tag{4.2}$$

where $\overline{z} = z_1 \ldots z_n$, $i \leq n$, and $\tau(\overline{z})$ is a conjunction of relational atomic formulas with no equalities, and ϕ_j is generated by the grammar (4.1).

Proof. Concerning (4.1). Clearly we may assume that ϕ is a sentence as all free variables can be existentially quantified without affecting the satisfiability and the fact that ϕ is in UN-normal form. It is also easy to see that non-unary occurrences of disjunction can be eliminated at the cost of a possible exponential blowup, using the fact that disjunction commutes with conjunction and with the existential quantifiers. This is an exponential transformation, reminiscent of the transformation of propositional formula into disjunctive normal form, and it does not affect the width of the formula, nor the size of other parameters that we will define later. This remark will be important during the complexity analysis as the width will appear in the exponent of the complexity of some forthcoming transformations.

The form (4.2) is straightforward to obtain by eliminating equalities by identifying the respective quantified variables. $\qquad \square$

In what follows let ϕ be any UNFP formula of the form described in Lemma 4.1, for which we want to test satisfiability.

We denote by SUBF_ϕ the set of all subformulas $\psi(y)$ of ϕ that have one free first-order variable. For any subformula of ϕ of the form

$$\exists \overline{z}(\tau(\overline{z}) \wedge z_i = y \wedge \bigwedge_{j \in \{1 \ldots n\} \setminus \{i\}} \phi_j(z_j)) \qquad \text{or} \qquad \exists \overline{z}(\tau(\overline{z}) \wedge \bigwedge_{j \in \{1 \ldots n\}} \phi_j(z_j)) \ ,$$

we call $\tau(\overline{z})$ a *neighborhood type*. We denote the set of neighborhood types in ϕ by NTYPES_ϕ. In the sequel we will sometime view a neighborhood type $\tau(\overline{z})$ as a structure whose universe is the set of variables of $\overline{z}$ and whose relations are the atoms of τ.

We now consider structures that are "stitched together" from copies of the neighborhood types in NTYPES_ϕ.

Definition 4.2. Consider the signature that contains an n-ary relation symbol R_τ for each neighborhood type $\tau(z_1, \ldots, z_n) \in \text{NTYPES}_\phi$. A structure in this new signature is called a *stitch diagram*. Each stitch diagram M gives rise to a *stitching* $M^\varkappa$, which is the structure (with the same domain of M) in the original signature obtained by replacing each R_τ-tuple with a copy of the neighborhood type τ (viewed as a structure) for all $\tau \in \text{NTYPES}_\phi$.

At this point, our basic strategy for reducing UNFP to sUNFP should be clear: we will produce an sUNFP sentence to describe stitch diagrams whose stitchings satisfy the desired UNFP sentence. In the rest of this section, we work out the details of this strategy.

It is important to realize that, even if a stitch diagram M does not contain an atomic fact $R_\tau(\overline{a})$, it may still be the case that $M^\varkappa \models \tau(\overline{a})$. In this case we say that the fact $R_\tau(\overline{a})$ is *implicit* in M. For example, this could happen if $M \models R_{\tau'}(\overline{a})$ and τ is contained in τ'. The following claim gives us a handle on when this phenomenon may occur. For any $\tau \in \text{NTYPES}_\phi$, we denote by $|\tau|$ the number of atomic formulas in τ. We write $N \subseteq M$ if N is a not-necessarily-induced substructure of M.

Lemma 4.3. If $R_\tau(\overline{a})$ is implicit in a stitch diagram M then there is an $N \subseteq M$ containing at most $|\tau|$ many facts, such that $R_\tau(\overline{a})$ is already implicit in N. Moreover N is connected whenever τ is.

Proof. We need at most one fact of M to account for each atom in $\tau(\overline{a})$. $\qquad\square$

Let $l = \max_{\tau \in \mathrm{NTYPES}_\phi} |\tau|$. We will restrict attention to stitch diagrams M that are l-acyclic. By Item (3) of Proposition 4.1 this is without loss of generality. This implies that every $N \subseteq M$ containing at most l facts is acyclic. The importance of the above claim, then, shows in two facts: (i) intuitively, there are finitely many reasons why a fact may be implicit in M, and (ii) each of these reasons is acyclic, and hence can be described in sUNFP as we will see.

Lemma 4.4. Let $\psi(y, \overline{X})$ be any subformula of ϕ with at most one free first-order variable. By induction on the structure of $\psi(y)$ we can construct a sUNFP formula $\psi'(y)$ such that, for all l-acyclic stitch diagrams M, all $a \in M$, and all sets $\overline{S}$ of elements of M, $M \models \psi'(a, \overline{S})$ iff $M^{\mathsf{x}} \models \psi(a, \overline{S})$.

Proof. The inductive translation commutes with all Boolean operators and with the LFP operator. Fix now any $\tau(\overline{z}) \in \mathrm{NTYPES}_\phi$ with $\overline{z} = z_1, \ldots, z_n$, fix an $i \leq n$, and fix a sequence of formulas $\psi_1, \ldots, \psi_{i-1}, \psi_{i+1}, \ldots, \psi_n \in \mathrm{SUBF}_\phi$ and assume ψ is of the form:

$$\psi(y) := \exists \overline{z}(\tau \wedge z_i = y \wedge \bigwedge_{j \in \{1\ldots n\} \setminus \{i\}} \psi_j(z_j)).$$

(The argument if ψ is of the form $\exists \overline{z}(\tau \wedge \bigwedge_{j \in \{i\ldots n\}} \psi_j(z_j))$ is similar. Note that these two cases also account for the base of the induction, if we let $n = 1$).

By induction we already have constructed sUNFP formulas $\psi'_1, \ldots, \psi'_{i-1}, \psi'_{i+1}, \ldots, \psi'_n$ corresponding to $\psi_1, \ldots, \psi_{i-1}, \psi_{i+1}, \ldots, \psi_n$.

We are interested in detecting in M how a node in M^{x} may come to satisfy ψ. We will construct a sUNFP formula that lists all the cases in M that make this happen. It clearly suffices to consider one connected component of τ at a time. Hence by Lemma 4.3 it only depends on a small neighborhood of x in M. The formula will then be essentially a long disjunction, where each disjunct corresponds to the description of a small neighborhood of M in which τ is implicitly satisfied by a tuple of nodes satisfying in addition the formulas ψ_j. Note that since we assume M to be l-acyclic, these small substructures are all acyclic, which will make it possible to describe them by an (existential) formula of sUNFP.

More precisely, consider any connected acyclic stitch diagram N containing at most l facts, and any homomorphism $h : \tau \to N^{\mathsf{x}}$. We now construct an sUNFP formula $\chi_{\psi,N,h}(y)$ that describes N (existentially positively) from the point of view of $h(z_i)$, and expressing also that each $h(z_j)$ satisfies ψ_j.

We shall make use of the following property of acyclic structures. If N is acyclic, i.e. $inc(N)$ is acyclic, then there is a tree $T(N)$ such that each of its nodes is labeled with a fact of N satisfying the following properties: (i) each atom of N is the label of exactly one node of $T(N)$, (ii) if a node u of $T(N)$ is the parent in $T(N)$ of a node v then their label share at most one element of N, (iii) if two nodes u and v of $T(N)$ share an element then either they are siblings of one is the parent of the other. In other words, we view N as a tree $T(N)$ rooted with an atom containing $h(z_i)$ and the formula describes that tree from top to bottom. We construct the desired sUNFP formula by induction on the number of nodes in $T(N)$.

When $T(N)$ contains only one node, whose label is $R_\tau(a_1, \cdots, a_m)$ and, assuming $h(z_i) = a_{i_0}$, the desired formula is then

$$\exists \overline{y} \ R_\tau(\overline{y}) \wedge y_{i_0} = y \wedge \bigwedge_{j \neq i, h(z_j) = a_{\alpha_j}} \psi'_j(y_{\alpha_j}).$$

Assume now that $T(N)$ is a tree whose root element is $u = R_\tau(a_1, \cdots, a_m)$ and with several subtrees $T(N_1), T(N_2) \cdots$. By property (ii) of $T(N)$, for all $j > 0$, the label of u and the elements of N_j share at most one element, say a_{β_j}. For $j > 0$, let h_j be the restriction of h to the elements of N_j. Finally assume that $h(z_i) = a_{i_0}$. The desired formula $\chi_{\psi, N, h}(y)$ is then:

$$\exists \overline{y} \Big(R_\tau(\overline{y}) \wedge y_{i_0} = y \wedge \bigwedge_{j \neq i, h(z_j) = a_{\alpha_j}} \psi'_j(y_{\alpha_j}) \ \wedge \ \bigwedge_j \chi_{\psi, N_j, h_j}(y_{\beta_j}) \Big)$$

Finally $\psi'(y)$ is the disjunction, for each N and h as above, of the formulas $\chi_{\psi, N, h}(y)$.

It follows from the construction that, for all l-acyclic stitch diagrams M, $M \models \psi'$ if and only if $M^\times \models \psi$. $\square$

It follows from Lemma 4.4 that if the constructed sUNFO formula ψ' is satisfiable (in the finite or in the infinite), by Theorem 2.3 it has a l-acyclic model (in the infinite this model is actually acyclic) and therefore ψ is satisfiable. Conversely, it is easy to construct, from a (finite) model M of ψ, a (finite) model M' of ψ'. Take for M' the structure whose domain is the domain of M and whose relation R_τ contains all the tuples of M satisfying τ. Altogether this shows that ψ is satisfiable (on finite structures) if and only if ψ' is. By Proposition 4.1 this implies that UNFP is decidable. A careful analysis of the complexity of the above translation actually yields:

Theorem 4.5. The satisfiability problem for UNFP is in 2ExpTime, both on arbitrary structures and on finite structures.

Proof. Assume first that the input UNFP formula ϕ satisfies the simplifying assumptions of Step 1. Recall that we denote by l the maximal number of conjuncts in neighborhood types, and by k the width of the formula. Note that the formula $\chi_{\psi, N, h}(y)$ is only polynomially long in the length of ψ, but for any given ψ, the number of possible structures N and homomorphisms h can be exponential. More precisely, each structure N to be considered has at most l facts and domain size at most $k \cdot l$. Moreover the number of possible atomic relations is $|\text{NTYPES}_\phi| = O(|\phi|)$ and each relation has arity at most k and cannot contain twice the same element. There are at most $|\phi|^{O(k \cdot l)}$ many such structures, up to isomorphism. For each such structure N, since the domain size is at most $l \cdot k$, the number of homomorphisms $h : \tau \to N^\times$ is at most $(l \cdot k)^k = |\phi|^{O(k)}$. All in all, the number of disjuncts occurring in ψ' is bounded by $|\phi|^{O(k \cdot l)}$. Lets now consider the size of one such disjunct: as it contains formulas of the form ψ'_i for smaller formulas, we obtain by induction that the size of the sUNFP formula ϕ' is bounded by $|\phi|^{O(r \cdot k \cdot l)}$ where r is the nesting depth of existential blocks.

Consider now the general case of a UNFP formula θ. Putting θ in UN-normal form is linear time. The transformation of θ into a formula ϕ satisfying the simplifying assumption is exponential in time but produces a formula whose parameters k, l and r are only polynomial (actually linear) in the size of θ. Hence, from the previous paragraph, it follows that the size of the resulting sUNFP formula ψ' can be bounded by $(2^{|\theta|})^{|\theta|^c}$ for some constant c, that is exponential in $|\theta|$.

Hence, we obtain by Proposition 4.1 that the satisfiability problem is in 2-ExpTime, both on arbitrary structures and on finite structures. □

We conclude this section by proving Theorem 3.3 (Finite Model Property of UNFO) and Theorem 3.4 (Tree-like Model Property of UNFP) using the reductions described above.

Proof of Theorem 3.3. Recall the construction of a sUNFP formula from a UNFP formula described above, and observe that, when starting with a formula of UNFO, we actually obtain a formula of sUNFO. Consider now a satisfiable UNFO formula ϕ. By construction the resulting sUNFO formula ϕ' is satisfiable. By Item (4) of Proposition 4.1 ϕ' has a finite model N. By construction $N^{\times}$ is a finite model of ϕ. □

Proof of Theorem 3.4. Similarly, consider a satisfiable UNFP formula ϕ. By construction the resulting sUNFP formula ϕ' is satisfiable. By Item (2) of Proposition 4.1 ϕ' has an acyclic model N. By construction this implies that $N^{\times}$ is a model of ϕ that has tree-width at most $k - 1$, where k is the width of ϕ'. □

4.3. **Lower bounds and restricted fragments.** The complexity result of Theorem 4.5 is tight:

Proposition 4.2. *There is a fixed finite signature such that the satisfiability problem for UNFO is 2ExpTime-hard, both on arbitrary structures and on finite structures.*

Proof. Fix an alternating 2^n-space bounded Turing machine M whose word problem is 2-ExpTime-hard. We may assume that the Turing machine runs in double exponential time (e.g., by maintaining a counter). Let w be a word in the input alphabet of M. We construct a formula ϕ_w that is satisfiable if and only if M accepts w. Moreover, if ϕ_w is satisfiable, then in fact it is satisfied in some finite tree structure. In this way, we show that the lower bound holds not only for arbitrary structures, but also for finite trees and for any class in-between. The formula ϕ_w describes an (alternating) run of M starting in the initial state with w on the tape, and ending in a final configuration.

The run is encoded as a finite tree whose nodes labeled by a unary predicate C represent configurations of the Turing machine, and where a child-edge between two C-nodes indicates that the second node represents a successor configuration of the first. Each C-node is also labeled by a unary predicate Q_i indicating the state of the Turing machine in that configuration. In addition, each C-node is the root of a subtree uniformly of height n, in which each non-leaf node has a child satisfying P_0 and a child satisfying P_1 (and no children satisfying both). We can associate to every leaf node of this subtree a number between 0 and 2^n, determined by whether each of its n ancestors (the node itself included) satisfies P_0 or P_1. Thus, each leaf node of the subtree represents a tape cell, and, using further unary predicates, we can encode at each tape cell the current content of that tape cell, and whether the head is currently located there. See Figure 2. All in all, the schema of the structure consists of a binary relation R, unary relations C, P_0, P_1, a unary relation Q_i for each state of the Turing machine, a unary relation for each element of the alphabet, and a unary relation H to represent the head position.

The construction of the formula ϕ_w is based on the above encoding of runs as structures. More precisely, ϕ_w is the conjunction of
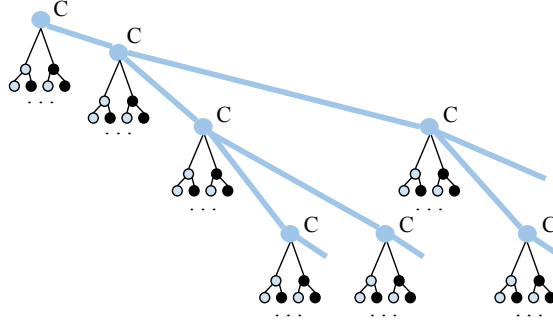
Figure 2: Encoding of a run of the alternating Turing machine

- A formula ϕ_1 expressing that the root node is labeled C, and that each C-node is the root of a subtree uniformly of height n in which every non-leaf node has a child satisfying P_0 and a child satisfying P_1 (and no children satisfying both).
- A formula ϕ_2 expressing that, whenever two leaf nodes represent the same tape cell in the same configuration, then they agree on all unary predicates (note that, in UNFO, we cannot force that there is only one node representing each tape cell in a given configuration, because we cannot express inequality).
- A formula ϕ_3 expressing that, whenever two leaf nodes represent the same tape cell in the successor configuration, and this tape cell is not the head position, then the two nodes agree on all unary predicates.
- A formula ϕ_4 encoding the transition function of the Turing machine (i.e., whenever, in some configuration, the Turing machine is in a $\exists$-state ($\forall$-state), and its head reading a particular letter, then for some (respectively, for every) possible transition there exists a corresponding successor configuration).
- A formula ϕ_5 expressing that, in the initial (root) configuration, the tape content is w and the Turing machine is in the initial head position and state; and all final configurations (i.e., C-nodes without C-successors) are accepting configurations.

We omit a detailed definition of the formulas in question, which is tedious but not difficult. For example, if we use $x \uparrow^n \downarrow^m y$ as a shorthand for a UNFO formula stating that there is a path going n steps up in a tree from x and then m steps down reaching y, if we use $\mathrm{leaf}(x)$ as a shorthand for $\neg \exists y R(x,y)$, and if we use $P_1^{\uparrow i}(x)$ as a shorthand for a UNFO formula that x has an i step ancestor satisfying P_1, then ϕ_2 can be expressed as follows (for every unary predicate A):

$$\neg \exists xy(\mathrm{leaf}(x) \wedge \mathrm{leaf}(y) \wedge x \uparrow^n \downarrow^n y \wedge \bigwedge_{i=0\ldots n-1} (P_1^{\uparrow i}(x) \leftrightarrow P_1^{\uparrow i}(y)) \wedge A(x) \wedge \neg A(y))$$

For ϕ_3 and ϕ_4 we make use of the following UNFO formula expressing the fact that x and y denote the same tape position in successive configurations:

$$\mathrm{leaf}(x) \wedge \mathrm{leaf}(y) \wedge (x \uparrow^{n+1} \downarrow^{n+2} y) \wedge \bigwedge_{i}(P_1^{\uparrow i}(x) \leftrightarrow P_1^{\uparrow i}(y))$$

The rest of the construction is straightforward. $\square$

The above proof actually established the lower bound for arbitrary structures, for finite trees, and on any class in-between. Moreover, the proof only uses formulas that have negation depth 2. For formulas of negation depth 1, the satisfiability problem turns out to have a lower complexity.

Theorem 4.3. *The satisfiability problem for* UNFO *formulas of negation depth 1 is* $\mathrm{NP}^{\mathrm{NP}}$-*complete (even for formulas containing unary predicates only).*

The proof of Theorem 4.3 is postponed until Section 5 as it makes use of results regarding the complexity of model checking which are obtained there.

5. Model Checking

In this section we study the complexity of the model-checking problem for UNFO and UNFP. The model-checking problem takes as input a structure M and a sentence ϕ, and it asks whether ϕ is true in M. We focus here on the *combined complexity* of the model-checking problem, where the input consists of a sentence and a structure. It was already observed in [16] that the model checking problem for UNFO (there called CRA(mon$\neg$)) is in P^{NP}. Here, we show that the problem is in fact $\mathrm{P}^{\mathrm{NP}[O(\log^2 n)]}$-complete, and that the model checking problem for UNFP is in $\mathrm{NP}^{\mathrm{NP}} \cap \mathrm{coNP}^{\mathrm{NP}}$. We refer to the reader to Section 2.3 for the definition of the relevant complexity classes.

5.1. **Model checking for UNFO.** We start by showing that the model checking problem for UNFO is $\mathrm{P}^{\mathrm{NP}[O(\log^2 n)]}$-complete. We then show that bounding the nesting of negations lowers the complexity to $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$-complete and that allowing subformula-sharing increases the complexity to P^{NP}-complete.

In order to give a better intuition about the upper bound algorithms we start by presenting a naive evaluation algorithm. Recall that the *negation depth* of a UNFO formula is the maximal nesting depth of negations in its syntactic tree. Let $\phi(x)$ be a formula of UNFO of negation depth l and M a model. Testing for an element $u \in \mathrm{dom}(M)$, whether $M \models \phi(u)$, can be done by induction on l using the following bottom-up strategy.

If $l = 0$ then ϕ is existential and we can test whether ϕ holds using one call to an NP-oracle (the oracle guesses witnesses fror the existentially quantified variables and then checks whether the remaining quantifier-free part of the formula holds).

If $l \neq 0$, then, for all elements v of M, and all maximal subformulas $\psi(y)$ of ϕ that have negation depth $l - 1$, we test whether $M \models \psi(v)$ using the induction on l. All these tests can be performed independently of each other, and hence (by induction hypothesis) we need only $l - 1$ parallel calls to the NP-oracle. Based on the outcomes of these calls, we compute a new structure M' by expanding M with new unary predicates recording for each element of M which of the maximal subformulas $\psi(y)$ of negation depth $l - 1$ hold at that element. We also transform ϕ into ϕ' replacing $\neg\psi(y)$ with the appropriate newly introduced unary predicate. It remains to evaluate ϕ' on M' using the same algorithm as in the base case where $l = 0$.

Altogether this yields an algorithm that makes l parallel calls to an NP-oracle. When l is constant this implies that the total process can be made in $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ and this gives the upper bound of Theorem 5.2 below. When l is at most $\log |\phi|$ this implies that the total process can be made in $\mathrm{P}^{\mathrm{NP}[O(\log^2 n)]}$ (recall the discussion in Section 2.3) and the upper bound of Theorem 5.1 essentially reduces the general case to this case.

Theorem 5.1. The model checking problem for UNFO is $P^{NP[O(\log^2 n)]}$-complete.

Proof. For the **lower bound**, we give a reduction from $\text{LEX}_2(\text{SAT})$: *given a Boolean formula $\phi(x_1....x_n)$, is $x_{\lceil \log^2(n) \rceil}$ true in the lexicographically maximal satisfying assignment?*

Let $\phi(x_1, \ldots, x_n)$ be a given Boolean formula, and let $d = \log(n)$ (we may assume that n is a power of 2). We are interested in knowing whether x_{d^2} is set to 1 in the lexicographically maximal satisfying assignment of ϕ. For this we construct, in time polynomial in n, a model M_n and a formula θ_n such that $M_n \models \theta_n$ iff x_{d^2} is set to 1 in the lexicographically maximal satisfying assignment of ϕ.

Let M_n be the structure containing n elements, $a_1, \ldots, a_n$, together with two elements, $1, 0$. The elements $1, 0$ represents truth and falsity and are distinguished from the elements $a_1, \ldots, a_n$ using a unary predicate Q that holds only for $0, 1$. Each of the elements $a_1, \ldots, a_n$ of M_n represents a bit-string of length d encoded using d unary predicates, $P_1, \cdots, P_d$, with the intended meaning that $P_i(a_j)$ holds in M_n iff the i-th bit of the bit-string represented by a_j is true. M_n is such that $a_1, \cdots, a_n$ code all possible bit-strings of length d.

Below, as a suggestive notation, we will write x_i for variables intended to range over truth and falsity, and y_i for variables intended to range over the n elements of M_n that represent length-d bit-strings. Hence $\exists x \, \psi$ should be understood as $\exists x \, Q(x) \wedge \psi$ and $\exists y \, \psi$ as $\exists y \, \neg Q(y) \wedge \psi$.

By $\widehat{\phi}$ we denote the UNFO formula obtained from ϕ by replacing, for $k \leq d$ and $j \leq d$, the variable $x_{(k-1)d+j}$ by $P_j(y_k)$. Note that the free variables of $\widehat{\phi}$ are $y_1, \ldots, y_d$ and $x_{d^2+1}, \ldots, x_n$. As $y_1, \ldots, y_d$ will range over bit strings of length d, we are interested in the last bit of y_d, i.e. $P_d(y_d)$.

We define, by induction on i, a formula $\chi_i(y)$ that is true for an element a_j if the length-d bit-string represented by a_j describes the bits $x_{(i-1)d+1} \cdots x_{(i-1)d+d}$ of the lexicographically maximal satisfying assignment of ϕ. It is convenient to define simultaneously another formula, $\psi_i(y)$, which is true for an element a_j if the length-d bit-string represented by a_j describes the bits $x_{(j-1)d+1} \cdots x_{(j-1)d+d}$ in the *some* satisfying assignment whose prefix up to $x_{(j-1)d}$ is the same as the lexicographically maximal satisfying assignment of ϕ.

$$\psi_i(y) = \exists y_1 \ldots y_d \, x_{d^2+1} \ldots x_n \left(\widehat{\phi} \wedge y_i = y \wedge \bigwedge_{j<i} \chi_j(y_j) \right)$$

$$\chi_i(y) = \psi_i(y) \wedge \neg \exists y'(y' > y \wedge \psi_i(y'))$$

where $y' > y$ is shorthand for a formula expressing that the bit-string denoted by y' is lexicographically greater than the bit-string denoted by y:

$$\bigvee_{i<d} \left(\left(\bigwedge_{j<i} P_j(y) \leftrightarrow P_j(y') \right) \wedge \neg P_i(y) \wedge P_i(y') \right).$$

Finally, take

$$\theta_n = \exists y_1 \ldots y_d \, x_{d^2+1} \ldots x_n \left(\widehat{\phi} \wedge \bigwedge_{i \leq d} (\chi_i(y_i)) \wedge P_d(y_d) \right).$$

Then θ_n is true in M_n if and only if x_{d^2} is true in the lexicographically maximal satisfying assignment of ϕ as required. Moreover θ_n is indeed a formula of UNFO. Finally notice that each χ_i has a size exponential in i but they are used only for $i \leq d = \log n$. Hence θ_n has polynomial size and can be computed in time polynomial in n.
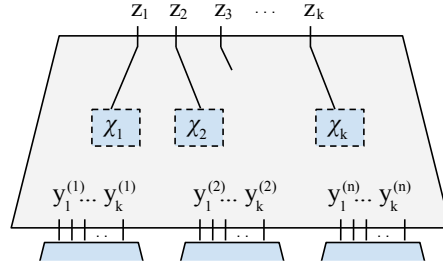
Figure 3: A block in a TB-tree with n children

We now turn to the **upper bound**. Recall the algorithm presented in the preamble of this section. If the syntactic tree of the formula would be a balanced binary tree then its negation depth l would be bounded by the log of the size of the formula and we would be done. The idea is to evaluate the formula by first making its syntactic tree balanced. This trick has already been applied in the context of branching time model checking using "Tree Block Satisfaction" [42]. Instead of redoing the trick, we reduce the model checking problem of UNFO to this one.

We first present "Tree Block Satisfaction", called $\text{TB(SAT)}_{1 \times M}$ in [42]. In fact, for present purposes, it suffices to consider a restricted version of the problem $\text{TB(SAT)}_{1 \times M}$, which we will refer to simply as TB(SAT) in what follows. We will describe here this restricted version. A TB-tree of width $k \geq 1$ is a tree consisting of blocks, where each block is, intuitively, a kind of Boolean circuit having k output gates and having k input gates for each of its children. See Figure 3. The ith output of a block is defined in terms of the input gates by means of an existentially quantified Boolean formula χ_i of the form

$$\exists \bar{b}_1 c_1 \ldots \bar{b}_m c_m \bar{d} \ \left(c_1 = \mathsf{input}_{i_1}(\bar{b}_1) \wedge \cdots \wedge c_m = \mathsf{input}_{i_m}(\bar{b}_m) \right) \wedge \psi \right)$$

where each $\bar{b}_j$ is a tuple of $\log k$ Boolean variables, encoding a number $[\![\bar{b}_j]\!]$ from 1 to k, and $\mathsf{input}_{i_j}(\bar{b}_j)$ represents the value of the $[\![\bar{b}_j]\!]$-th output bit of the i_j-th child block (which is denoted by $y^{(i_j)}_{[\![\bar{b}_j]\!]}$ in Figure 3) and ψ is a Boolean formula using any of the existentially quantified Boolean variables.

TB(SAT) is then the problem, given a TB-tree and a truth assignment for all inputs of the blocks that are leaves of the TB-tree, whether the first output bit of the root block evaluates to true. It was shown in [42, corollary 3.4] that TB(SAT) can be decided in $\mathrm{P}^{\mathrm{NP}[O(\log^2 n)]}$.

Given a formula ϕ of UNFO and a structure N, we construct in polynomial time a TB-tree that is a yes instance of TB(SAT) iff $N \models \phi$. The reduction is simple and reflects the naive evaluation of ϕ on N.

The construction of the TB-tree is by induction on ϕ. The width k of the TB-tree is the size of the domain of N. At each step of the induction the block of the root of the TB-tree is associated to a subformula of ϕ having one free variable, and is such that its output gate i is set to 1 iff the i-th element of N makes the associated subformula true. Altogether the shape of the TB-tree resemble the one of the syntactic tree of the formula. In the QBF formulas constructed below we denote by $\bar{b}$ a vector of $\log k$ variables and $[\![\bar{b}]\!] = i$ is a shorthand for the Boolean formula stating that $\bar{b}$ represents the binary encoding of i.

Let $\phi(x)$ be any formula in one free variable.

Case 1: $\phi(x)$ is of the form $\neg\psi(x)$. By induction hypothesis, we have a TB-tree for $\psi(x)$ such that the outputs from the root block indicate which elements of N satisfy $\psi(x)$. We extend this TB-tree with one extra block on top, in which the i-th output gate is defined by the formula
$$\exists \bar{b}, c \ (c = \mathsf{input}(\bar{b}) \wedge [\![\bar{b}]\!] = i \wedge c = 0).$$
This formula sets its output to 1 iff its input i is set to 0, hence preserves the inductive hypothesis as required.

Case 2: $\phi(x)$ is built up from atomic formulas and formulas in one free variable using conjunction and disjunction and existential quantification. Let $y_1, \ldots, y_n$ be the set of all existentially quantified variables in ϕ. Let $\psi_1(z_1) \ldots, \psi_m(z_m)$ be the maximal sub-formulas in one free variable occurring in ϕ, where $z_1 \ldots z_m$ are among $x, y_1, \ldots, y_n$. We may assume without loss of generality that $z_1, \ldots, z_m$ are the first m variables from the sequence $y_1, \ldots, y_n$ (in particular that $z_1, \ldots, z_m$ are distinct variables) because if not, one can always introduce another existentially quantifier variable y_{n+1} and replace $\psi_j(z_j)$ by $(\psi_j(y_{n+1}) \wedge y_{n+1} = z_j)$. By induction, we have a TB-block T_j for each $\psi_j(y_j)$ such that the outputs from the root block of T_j indicate which elements of N satisfy $\psi_j(y_j)$. The TB-tree for ϕ will consist of a new root block whose children are the roots of each of the T_j. The definition of the i-th output gate is
$$\exists \bar{b}_1, c_1, \ldots, \bar{b}_m, c_m, \bar{b}_{m+1}, \ldots, \bar{b}_n \ \big(\bigwedge_{j \leq m} (c_j = \mathsf{input}_j(\bar{b}_j)) \wedge \chi_N \big)$$

where χ_N is obtained from ϕ by replacing each subformula of the form $\psi_l(y_l)$ by c_l, each subformula of the form $x = y_l$ by $\bar{b}_l = i$, each subformula of the form $y_l = y_{l'}$ by $\bar{b}_l = \bar{b}_{l'}$ and each subformula of the form $R(y_{l_1}, \ldots, y_{l_\kappa})$ by a Boolean formula listing all tuples in the relation R^N:
$$\bigvee_{([\![\bar{d}_1]\!], \ldots, [\![\bar{d}_\kappa]\!]) \in R^N} (\bar{b}_{l_1} = \bar{d}_1 \wedge \ldots \wedge \bar{b}_{l_\kappa} = \bar{d}_\kappa)$$

Note that Case 2 covers the base case when $\phi(x)$ has no subformula with one free variable.

It is now easy to check that the TB-tree constructed by the above induction has the desired property and can be computed in polynomial time. This concludes the proof of the Theorem. $\square$

As mentioned earlier, restricting the nesting of negations gives a lower complexity.

Theorem 5.2. For all $l > 0$, the complexity of the model-checking problem for UNFO formula of negation depth bounded by l is $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$-complete. The lower bound holds even for a fixed structure.

Proof. For the **lower bound** we use the equivalent characterization of $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ as the class of problems that are PTime truth-table reducible to NP. As 3-colorability is NP-complete, every problem in $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ is PTime truth-table reducible to 3-colorability. Recall from Section 2.3 that a PTime truth-table reduction from a given problem to 3-colorability is a PTime algorithm that, given an instance of the problem, produces a set $y_1, \cdots, y_n$ of inputs to 3-colorability, together with a Boolean formula $\phi(x_1, \cdots, x_n)$, such

that the input is a yes-instance iff ϕ evaluates to true after replacing each x_i by 1 if y_i is 3-colorable and 0 otherwise.

We show that we can reduce any problem that is PTime truth-table reducible to 3-colorability to the model checking problem for UNFO formula of negation depth 1. For this, fix a problem P that is PTime truth-table reducible to 3-colorability. Let w be an input string for P. We construct from w, in time polynomial in $|w|$, an instance M and a formula ϕ_w such that $M \models \phi_w$ iff $w \in P$. As M will be independent of w and ϕ_w will be in UNFO and with nesting depth 1, this will show $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$-hardness.

By assumption, there is a polynomial time algorithm that produces from w a tuple $\langle G_1, \cdots, G_n, C(x_1, \ldots, x_n) \rangle$ such that $w \in P$ iff $C(x_1, \ldots, x_n)$ evaluates to 1 after replacing each of the x_i by 1 iff G_i is 3 colorable.

For each graph G, let q_G be the *canonical conjunctive query of G*, that is, the existentially quantified conjunction of relational atoms, where there is an existentially quantified variable for each node of G, and a relational atom for each edge of G. Note that, since q_G does not use negation, it belongs to UNFO. Let M be the structure representing a clique of size 3. It is well known that $M \models q_G$ iff G is 3-colorable. Let ϕ_w be the formula constructed from C by replacing each occurrence of x_i with the sentence q_{G_i}. It is immediate to verify that the resulting formula is in UNFO, has negation depth 1 and satisfies the properties required for the reduction. Moreover ϕ_w can be computed in time polynomial in $|w|$ as this was the case for $\langle G_1, \cdots, G_n, C(x_1, \ldots, x_n) \rangle$. This completes the proof for the lower-bound.

The **upper bound** is obtained using the naive evaluation of the formula presented in the preamble of this section as l is treated as a constant. $\qquad\square$

We now consider formalism equivalent in expressive power to UNFO but with a more succinct syntax that allows the sharing of subformulas. Let us denote by UNFO(let) the extension of UNFO with Boolean variables $b_1, b_2, \ldots$ (ranging over truth values) that can be used as atomic formulas, and with a new construct let $b = \phi$ in ψ, where b is a Boolean variable, ϕ is a sentence, i.e., a formula without free first-order variables but possibly with free Boolean variables (excluding b itself), and ψ is a formula that may use all the Boolean variables, including b. We only consider UNFO(let) formulas without free Boolean variables, i.e., in which each occurring Boolean variable is bound by a let operator. The semantics of UNFO(let) formulas is as expected: when the valuation of the Boolean free variables of ϕ is known, we can derive a valuation for b using $b = \phi$ and then we can evaluate ψ.

Example 5.3. A typical formula of UNFO(let) looks like this:

$$\mathsf{let}\ b = \psi\ \mathsf{in}\ \exists x_1, \ldots x_n\ \big(b = 0 \wedge \psi_1(\overline{x})\big) \vee \big(b = 1 \wedge \psi_2(\overline{x})\big)$$

It is equivalent to the UNFO formula:

$$\exists x_1, \ldots x_n\ \big(\neg\psi \wedge \psi_1(\overline{x})\big) \vee \big(\psi \wedge \psi_2(\overline{x})\big)$$

The UNFO(let) model checking problem is the problem of evaluating a given UNFO(let) formula without free Boolean variables in a given structure.

Theorem 5.4. The UNFO(let) model-checking problem is P^{NP}-complete. The lower bound holds even for a fixed structure.

Proof. For the **upper bound**, we use a simple bottom-up evaluation strategy. Let ϕ be any UNFO(let) formula. We may assume without loss of generality that each let-operator in ϕ binds a different Boolean variable. We assign a rank to each Boolean variable occurring in

ϕ: a Boolean variable b has rank 0 if its definition (i.e., the sentence to which it is bound by its let-operator in ϕ) does not contain any Boolean variables (free or bound), and b has rank $k+1$ if its definition only contains Boolean variables of rank at most k. Using Theorem 5.1 repeatedly as an oracle, we can compute in polynomial time a truth value for each Boolean variable. Finally, by replacing all Boolean variables in ϕ by their truth value and applying Theorem 5.1 once more, we find out whether ϕ is true in the structure. Altogether this yields a P^{NP} algorithm.

For the **lower bound**, we make use of the problem LEX(SAT): *given a satisfiable Boolean formula $\phi(x_1, \ldots, x_n)$, test if the value of x_n is 1 in the lexicographically maximal solution* (cf. Section 2.3). Given a satisfiable Boolean formula $\phi(x_1, \ldots, x_n)$ we construct in time polynomial in n a model M and a UNFO(let) formula ψ such that $M \models \psi$ iff x_n is 1 in the lexicographically maximal solution of ϕ.

The idea of the reduction will be to construct the lexicographically maximal solution $B_1 \cdots B_n$ of ϕ bit by bit from B_1 to B_n using the following algorithm: B_1 is true iff $\phi(\top, x_1, \ldots, x_n)$ is satisfiable and if $B_1 \cdots B_i$ have already been computed then B_{i+1} is true iff $\phi(B_1, \cdots, B_i, \top, x_{i+1}, \cdots, x_n)$ is satisfiable.

Specifically, our reduction uses a fixed structure M with two elements, one of which is labeled by a unary predicate T and intuitively represents *true* while the other element represents *falsity*. We let $\widehat{\phi}$ be the UNFO formula obtained from ϕ by replacing each positive occurrence of the variable x_i in ϕ by the formula $T(x_i)$ and each occurrence of $\neg x_i$ in ϕ by the formula $\neg T(x_i)$.

We construct by induction sentences $\psi_1, \ldots, \psi_n$, where each ψ_i is true in M if and only if B_i is 1. By definition $B_1 = 1$ iff ϕ has an satisfiable assignment setting x_1 to true. This can be expressed using the formula:

$$\psi_1 := \exists x_1 \ldots x_n \ (\widehat{\phi}(x_1, \ldots . x_n) \wedge T(x_1)) \ ,$$

By definition the following UNFO formula, obtained by induction, expresses that B_i is 1:

$$\psi_i := \exists x_1 \ldots x_n \ (\widehat{\phi}(x_1, \ldots, x_n) \wedge (\bigwedge_{j<i} T(x_j) \leftrightarrow \psi_j) \wedge T(x_i)).$$

Notice that the size of ψ_i is exponential in i and therefore, even though ψ_n does express the truth value of B_n, this would not give a polynomial time reduction. However, using the let construction of UNFO(let) we can derive the same formulas more succinctly.

Let ψ_i' be the formula (with $b_1, \ldots, b_{i-1}$ as free Boolean variables):

$$\exists x_1, \ldots x_n (\widehat{\phi}(x_1, \ldots, x_n) \wedge (\bigwedge_{j<i} T(x_j) \leftrightarrow b_j) \wedge T(x_i)) \ .$$

Then we set ψ be the UNFO(let) formula:

$$\psi := \mathsf{let} \quad b_1 = \psi_1' \quad \mathsf{in}$$
$$\mathsf{let} \qquad b_2 = \psi_2' \quad \mathsf{in}$$
$$\ldots$$
$$\mathsf{let} \quad b_{n-1} = \psi_{n-1}' \quad \mathsf{in} \quad \psi_n'$$

Then notice that ψ has all the desired properties: it is in UNFO(let), it can be computed in time polynomial in n and it verifies that $M \models \psi$ iff $B_n = 1$. This concludes the proof. $\square$

Satisfiability for UNFO *formulas of negation depth 1.* We are now also in a position to give the proof of Theorem 4.3, which states that testing satisfiability for UNFO formulas of negation depth 1 is NP^{NP}-complete.

Proof of Theorem 4.3. Let ψ be any UNFO formula of negation depth 1, and let χ be the first-order formula obtained by bringing ϕ into prenex normal form. Note that χ is, in general, no longer a UNFO formula. Since ψ does not contain nested negations or universal quantifiers, χ is of the form $\exists \overline{x} \forall \overline{y} \chi'$, where χ' is a quantifier-free formula, which we may assume to be in negation-normal form. In addition, we know that the variables $\overline{y}$ can only occur in negative atomic subformulas of χ. In other words, all positive atomic subformulas of χ use only variables in $\overline{x}$. We claim that χ, and hence, ψ, has a "polynomial size model property": suppose $M \models \forall \overline{y} \phi(\overline{a}, \overline{y})$. Let M' be the submodel of M containing only the elements $\overline{a}$ and containing only the (polynomially many) facts about $\overline{a}$ that occur positively in $\phi(\overline{a}, \overline{y})$. It is easy to see that M' still satisfies $\forall \overline{y} \phi(\overline{a}, \overline{y})$. The existence of a model of ϕ can now be tested by guessing a structure of polynomial size and then applying the model checking procedure to verify whether it is a model or not. Using Theorem 5.2, this implies that the satisfiability problem for UNFO-formulas of negation depth 1 is in NP^{NP}.

Hardness is by reduction from the problem of evaluating QBF formulas of the form

$$\phi := \exists \overline{x} \forall \overline{y} \psi$$

with $\overline{x} = x_1 \ldots x_n$ and $\overline{y} = y_1 \ldots y_m$ which is known to be complete for NP^{NP}. We will construct in PTime a UNFO formula of negation depth 1 that is satisfiable if and only if ϕ evaluates to true. Take the vocabulary consisting of unary predicates T and F plus unary predicates $P_1, \ldots, P_n$ corresponding to the existentially quantified variable $\overline{x}$. The UNFO formula is defined as the conjunction

$$\exists x. T(x) \wedge \exists x. F(x) \wedge \neg \exists x. (T(x) \wedge F(x))$$

$$\wedge \bigwedge_{i \leq n} \exists x (P_i(x) \wedge (T(x) \vee F(x))) \wedge \neg (\exists x. (P_i(x) \wedge T(x)) \wedge \exists x (P_i(x) \wedge F(x)))$$

$$\wedge \neg \exists \overline{y} (\bigwedge_i (T(y_i) \vee F(y_i)) \wedge -\phi)$$

where $-\phi$ is obtained by negating ϕ, then pushing the negations down to the atoms, and then replacing y_i by $T(y_i)$, replacing $\neg y_i$ by $F(y_i)$, replacing x_i by $\exists x (P_i(x) \wedge T(x))$, and replacing $\neg x_i$ by $\exists x (P_i(x) \wedge F(x))$.

It is straightforward to verify that this UNFO formula is satisfiable if and only if ϕ evaluates to true, and that it has negation depth 1. □

5.2. **Model checking for UNFP.** We now turn to the complexity of the model checking of UNFP. It is convenient, at this point, to treat the greatest-fixpoint operator (GFP) as a primitive operator, instead of as a defined connective. This way we can use dualization in order to assume without loss of generality that all formulas of UNFP are such that *every occurrence of a fixpoint operator in ϕ is a positive occurrence (i.e., lies under an even number of negations)*. We also assume without loss of generality that each fixpoint operator binds a different variable, so that we can speak of *the fixpoint definition* of a variable X, by which we mean the formula directly below the fixpoint operator binding X. The *dependency graph* of fixpoint variables in ϕ is the directed graph whose nodes are the fixpoint variables in ϕ and where there is an edge from X to Y if Y occurs (free or bound)

inside the fixpoint definition of X. We say that a UNFP formula is *alternation free* if the dependency graph does not have a cycle containing both a least fixpoint variable and a greatest fixpoint variable [37].

We first consider the alternation-free fragment.

Theorem 5.5. The model-checking problem for the alternation-free fragment of UNFP is P^{NP}-complete.

Proof. Recall that we assume the formulas are such that *every occurrence of a fixpoint operator in ϕ is a positive occurrence (i.e., lies under an even number of negations).*

We first prove the claim for formulas containing only LFP operators (no GFP operators). Let ϕ be any UNFP formula containing only least fixpoint operators and M be a model. Let $\overline{X} = X_1, \ldots, X_n$ be the fixpoint variables occurring in ϕ (we assume each fixpoint operator binds a different variable). Initially, we assign each X_i to be the empty set. We then repeatedly consider each fixpoint variable X_i and evaluate its fixpoint definition β_i, viewed as a UNFO formula by using the current choice of sets $\overline{X}$ to interpret the free set variables of β_i as well as the fixpoint subformulas of β_i, and check if new elements are derived that do not already belong to the set X_i. If this is the case, we add the elements in question to X_i. We repeat this procedure until no new elements are derived. It is well known that the resulting sets we end up with are the least fixpoint solutions for the fixpoint subformulas of ϕ. Furthermore, the number of iterations is polynomial, since, in each iteration, at least one element of M gets added to on of the sets, and each iteration can be performed in $P^{NP[O(\log^2 n)]}$ by Theorem 5.1.

By dualization, we get the same result for formulas containing only greatest fixpoint operators. The result is then easily lifted to the full alternation free fragment by induction on the alternation rank of the fixpoint variable in question, where the alternation rank is defined as the (finite) maximal number of fixpoint alternations on an outgoing path from that variable in the dependency graph: we simply perform a bottom up evaluation based on the syntactic tree defining ϕ, coloring each node of M with the fixpoint formulas that it satisfies using the above algorithms.

For the lower bound, we reduce from LEX(SAT). Let $\phi(x_1, \ldots, x_n)$ be any Boolean formula that is the input of the LEX(SAT) problem. We may assume without loss of generality that ϕ is in negation normal form. We construct a structure M, whose domain is $\{t_1, \ldots, t_n, f_1, \ldots, f_n\}$, and with unary predicates T and $V_1, \ldots, V_n$, such that $T^M = \{t_1, \ldots, t_n\}$, and $V_i^M = \{t_i, f_i\}$. Intuitively, each element t_i represents the eventuality that the value of x_i in the lexicographically maximal satisfying assignment is 1, while f_i represents the eventuality that the value of x_i in the lexicographically maximal satisfying assignment is 0. Using a fixpoint formula, we will compute the set of all elements that are actually "true" in the lexicographically maximal satisfying assignment. It then suffices only to check whether t_n belongs to this set.

Let ϕ' be obtained from ϕ by replacing every occurrence of a Boolean variable x_i by $T(x_i)$. Let θ be the formula $\exists z(V_n(z) \wedge T(z) \wedge [\text{LFP}_{X,x}\psi](z))$ where ψ is the disjunction of all formulas of the following forms, for all $1 \le k \le n$:

$$V_k(x) \wedge T(x) \wedge \exists x_1, \ldots, x_n(\bigwedge_{i=1\ldots n} V_i(x_i) \wedge \phi' \wedge X(x_1) \wedge \cdots \wedge X(x_{k-1}) \wedge T(x_k))$$

and

$$V_k(x) \wedge F(x) \wedge \neg\exists x_1, \ldots, x_n(\bigwedge_{i=1\ldots n} V_i(x_i) \wedge \phi' \wedge X(x_1) \wedge \cdots \wedge X(x_{k-1}) \wedge T(x_k))$$

It is easy to see from the construction that θ is true in M if and only if $x_n = 1$ in the lexicographically maximal satisfying assignment for ϕ. □

We now turn to the general case.

Theorem 5.6. The UNFP model checking problem is in $\mathrm{NP^{NP}} \cap \mathrm{coNP^{NP}}$ and $\mathrm{P^{NP}}$-hard.

Proof. The lower bound is immediate from Theorem 5.5. We prove here the upper bound.

Since UNFP is closed under unary negation, it is enough to show that the model-checking problem is in $\mathrm{NP^{NP}}$. We show a slightly stronger result proving that, for every formula in at most one free variable, we can decide in $\mathrm{NP^{NP}}$ whether a given element of a given structure makes the formula true. The algorithm we describe below is inspired by an idea from [45] to reduce the problem to the case of formulas that only contain least fixpoint operators (and no greatest fixpoint operators) by guessing a set for each greatest fixpoint operator.

In the sequel we will only consider formulas ϕ with one free first-order variable and by "evaluating" this formula over M, intuitively, we mean computing, non-deterministically, a set of elements of M making the formula true.

Recall from the beginning of the section that we consider UNFO-formula in which we allow both LFP and GFP-operators, and we require that all occurrences of fixpoint operators are positive. Equivalently, we view each UNFP formula as being defined inductively by the grammar:

$$\phi ::= \alpha(\overline{X}, \overline{\psi}, x)$$
$$\psi ::= [\mathrm{LFP}_{Y,y} \ \phi(Y, \overline{X}, y)](x) \mid [\mathrm{GFP}_{Y,y} \ \phi(Y, \overline{X}, y)](x)$$

where α is a UNFO formula with one free first-order variable, possibly several monadic second-order variables, and possibly using as atoms fixpoint subformulas $\psi(z)$ defined by mutual induction using the above grammar, such that $\overline{X}$ and $\overline{\psi}$ occur only positively (under an even number of negations) in α.

Given a formula of the form $\alpha(\overline{X}, \overline{\psi}, x)$ we denote by $\hat{\alpha}$ the UNFO formula constructed from α by replacing each nested fixpoint subformula $\psi(z)$ by $Y(z)$ where Y is the variable defined by ψ. Hence $\hat{\alpha}$ has x as unique first-order free variable and $\overline{X}$ and $\overline{Y}$ as monadic second-order free variables. The positivity condition stated above implies that $\hat{\alpha}$ is monotonic with respect to $\overline{X}$ and $\overline{Y}$.

Given a structure M, sets $\overline{U}$ of elements of M and a UNFO formula $\phi(\overline{X}, x)$ we denote by $eval(\phi, (M, \overline{U}))$ the subset of the domain of M computed by induction on the syntactic representation of ϕ using the following algorithm where, as a convenient notation, we also write $eval(\overline{\phi}, (M, \overline{U}))$ to denote the tuple $(eval(\phi_1, (M, \overline{U})), \ldots, eval(\phi_n, (M, \overline{U})))$, where $\overline{\phi} = \phi_1, \ldots, \phi_n$:

$eval(\phi, (M, \overline{U}))$
- Case ϕ is of the form $\alpha(\overline{X}, \overline{\psi}, x)$
  (1) Compute $\overline{V} := eval(\overline{\psi}, (M, \overline{U}))$ by induction
  (2) Evaluate $\hat{\alpha}$ on $(M, \overline{U}, \overline{V})$, using the algorithm of Theorem 5.1
- Case ϕ is of the form $[\mathrm{LFP}_{Y,y} \ \alpha(Y, \overline{X}, \overline{\psi}, y)](x)$

(1) set $S := \emptyset$
(2) Compute $\overline{V} := eval(\overline{\psi}, (M, \overline{U}, S))$ by induction
(3) Compute $S' := eval(\hat{\alpha}, (M, \overline{U}, \overline{V}, S))$ by induction
(4) If $S' = S$ return S otherwise go to Step 2 with $S := S \cup S'$
- Case ϕ is of the form $[\text{GFP}_{Y,y} \; \alpha(Y, \overline{X}, \overline{\psi}, y)](x)$
  (1) guess T
  (2) Compute $\overline{V} := eval(\overline{\psi}, (M, \overline{U}, T))$ by induction
  (3) Compute $T' := eval(\hat{\alpha}, (M, \overline{U}, \overline{V}, T))$ by induction
  (4) If $T' = T$ return T otherwise abort

It should be clear that the algorithm is in NP^{NP}: by monotonicity it makes only polynomially many inductive calls, and the base case corresponds to the evaluation of a UNFO formula and is therefore in $\text{P}^{\text{NP}[O(\log^2 n)]}$ by Theorem 5.1.

It remains to show that this result returns the correct answers. Recall the semantics of fixpoints as described in Section 2. It is easy to see that the algorithm below computes exactly the correct results according to this semantics.

$eval^*(\phi, (M, \overline{U}))$
- Case ϕ is of the form $\alpha(\overline{X}, \overline{\psi}, x)$
  (1) Compute $\overline{V} := eval(\overline{\psi}, (M, \overline{U}))$ by induction
  (2) Evaluate $\hat{\alpha}$ on $(M, \overline{U}, \overline{V})$
- Case ϕ is of the form $[\text{LFP}_{Y,y} \; \alpha(Y, \overline{X}, \overline{\psi}, y)](x)$
  (1) set $S := \emptyset$
  (2) Compute $\overline{V} := eval(\overline{\psi}, (M, \overline{U}, S))$ by induction
  (3) Compute $S' := eval(\hat{\alpha}, (M, \overline{U}, \overline{V}, S))$ by induction
  (4) If $S' = S$ return S otherwise go to Step 2 with $S := S \cup S'$
- Case ϕ is of the form $[\text{GFP}_{Y,y} \; \alpha(Y, \overline{X}, \overline{\psi}, y)](x)$
  (1) set $T = \text{dom}(M)$
  (2) Compute $\overline{V} := eval(\overline{\psi}, (M, \overline{U}, T))$ by induction
  (3) Compute $T' := eval(\hat{\alpha}, (M, \overline{U}, \overline{V}, T))$ by induction
  (4) If $T' = T$ return T otherwise go to Step 2 with $T := T \cap T'$

Let's denote by $eval^*(\phi, (M, \overline{U}))$ the set computed by this second algorithm, i.e. the standard algorithm for fixpoint formulas. Notice that it differs with the previous algorithm only in the case of greatest fixpoints. The fact that $eval(\phi, (M, \overline{U})) = eval^*(\phi, (M, \overline{U}))$ is a consequence of the following claim.

Claim 5.1. For all $M, \overline{U}$ and ϕ we have $eval(\phi, (M, \overline{U})) \subseteq eval^*(\phi, (M, \overline{U}))$. Moreover, if $eval$ always guess the correct greatest fixpoint then we have quality.

Proof. This is a simple induction on ϕ. For Case (1) we obtain by induction that $\overline{V} \subseteq \overline{V}^*$ and the result follows immediately by the monotonicity assumption on $\hat{\alpha}$. Similarly, for Case (2), we obtain that each stage of the computation of the least fixpoint by $eval$ is included in the same stage by $eval^*$. The inclusion of the respective least fixpoints follows. For Case (3) we obtain again by induction that $\overline{V} \subseteq \overline{V}^*$ and therefore that $T \subseteq eval(\hat{\alpha}, (M, \overline{U}, \overline{V}^*, T))$ by the monotonicity assumption on $\hat{\alpha}$. If we denote by T^* the greatest fixpoint of $\hat{\alpha}$ on $(M, \overline{U}, \overline{V}^*)$, by Knaster-Tarski Theorem, this implies that $T \subseteq T^*$ as desired.

The second part of the claim is immediate.   □

This concludes the proof of Theorem 5.6.   □

6. Trees

In this section, we study the expressive power and computational complexity of UNFO and UNFP on trees. We consider three types of trees: *binary trees* (with two deterministic successor relations $child_1, child_2$, and any number of unary predicates), *unranked trees* (with a single child relation, and any number of unary predicates), and *XML trees* (that is, trees in which nodes can have any number of children and the children of each node are ordered, and where the signature consists of the horizontal and vertical successor and order relations, and any number of unary predicates). We will consider finite trees, but all proofs generalize to infinite trees.

On XML trees with all axes, it is known that *Core XPath* = FO^2 for unary queries while *Core XPath* = UCQ-*over*-FO^2-*unary-predicates* for binary queries [35]. It turns out that UNFO has the same expressive power as Core XPath, both for unary and for binary queries (cf. [16], where UNFO is called CRA(mon¬)) and therefore UNFO characterizes Core XPath in a more uniform way. In particular, since the XML tree languages definable in Core XPath are precisely the ones definable in FO^2 [35], this implies that UNFO defines the same XML tree languages as FO^2. The same holds already for $UNFO^2$.

In this section, we further analyze the expressive power and complexity of UNFO and UNFP on trees.

The following observation will be helpful. We say that two unranked trees are *root-to-root bisimilar* if the roots of the two trees are bisimilar.

Lemma 6.1. Two unranked trees are UN-bisimilar if and only if they are root-to-root bisimilar.

Proof. It is clear that every UN-bisimulation is a root-to-root bisimulation. Conversely suppose t, t' are root-to-root bisimilar trees, with roots r and r'. For any node a of t, we denote by $depth_t(a)$ the distance from r to a, and we denote by by t_a the subtree of t rooted at a. Similar notations apply to t'. Let Z consist of all pairs (a, b) of nodes from t and t', respectively, such that (i) $depth_t(a) = depth_{t'}(b) = k \geq 0$, and (ii) for each $i \leq k$, the subtree of t rooted at the i-th ancestor of a and the subtree of t' rooted at the i-th ancestor of b are root-to-root bisimilar.

We claim that Z is a UN-bisimulation. Let $(a, b) \in Z$. We show how to construct a homomorphism h from t to t' that maps a to b. The other direction is established in the same way.

The homomorphism h is constructed as follows: first, the root-to-root bisimulation between t_a and t'_b induces a homomorphism from t_a to t'_b. If $a = r$ and $b = r'$, we are done. Otherwise, let a' be the parent of a and let b' be the parent of b. Then the root-to-root bisimulation between $t_{a'}$ and $t'_{b'}$ induces a homomorphism from $t_{a'}$ to $t'_{b'}$, which, we may assume, extends the previously constructed homomorphism from t_a to t'_b. Repeating the same argument, after k many steps, we obtain a homomorphism from t to t' that maps a to b, and we are done. □

We start with UNFP. Recall that UNFP is included into MSO. Hence, over all kind of trees, UNFP sentences only define regular languages (that is, MSO-definable classes of trees). The converse is also true:

Theorem 6.1. *The following hold both over the class of finite trees and over the class of finite and infinite trees:*

(1) *On binary trees,* UNFP *defines the regular languages.*

(2) *On XML trees,* UNFP *defines the regular languages.*
(3) *On unranked trees,* UNFP *defines the root-to-root bisimulation invariant regular languages.*

The same hold for UNFP^2.

Proof. It is known that all the three statements hold for the μ-calculus [31] (as well as for monadic Datalog [23]). Hence, as the μ-calculus is a fragment of UNFP this shows the first two claims and one direction of the third claim. Moreover, as every formula of the μ-calculus is equivalent to a UNFP^2 sentence, UNFP collapses to UNFP^2 over trees. Clearly, UNFP can only define regular languages that are invariant under UN-bisimulation. Hence the other direction of the third claim follows from Lemma 6.1. $\qquad\square$

We now turn to UNFO. The k-neighborhood of a node of a tree is the subtree rooted at that node, up to depth k. A binary tree language is LT (*Locally Testable*) if membership into this language is determined by the presence or absence of isomorphism-types of k-neighborhoods for some k. Similarly, an unranked tree language is ILT (*Idempotent Locally Testable*) if membership is determined by the presence of absence of bisimulation-types of k-neighborhoods, for some k.

Theorem 6.2. *The following hold both over the class of finite trees and over the class of finite and infinite trees:*
(1) *On binary trees,* UNFO *defines the LT regular languages.*
(2) *On unranked trees,* UNFO *defines the ILT regular languages.*

The same hold for UNFO^2.

Proof. Since binary trees have bounded degree, there are only finitely many isomorphism types of k-neighborhoods for any given k. Moreover, each can be completely described by a UNFO formula. It follows that the LT regular languages can be defined in UNFO (in fact, in UNFO^2). Incidentally, note that the only negation used in this construction is Boolean negation (i.e., negation applied to sentences), except for expressing the fact that a node is the root or is a leaf.

For unranked trees, the ILT regular languages are precisely the ones that can be defined by a global ML formula as defined in Section 2.2 [40]. It is clear that this language is contained in UNFO, and therefore all ILT regular language are definable by a UNFO-sentence (in fact, a UNFO^2-sentence).

For the other direction, let ϕ be a sentence of UNFO. Without loss of generality we can assume that ϕ is in UN-normal form and satisfies the simplifying assumptions described in Step 1 of Section 4.2.

We can further assume without loss of generality that the conjuncts $\tau(\bar{z})$ occurring in the formulas:

$$\exists \bar{z}\big(\tau(\bar{z}) \wedge z_i = y \wedge \bigwedge_{j\in\{1\dots n\}\backslash\{i\}} \phi_j(z_j)\big) \qquad \text{or} \qquad \exists \bar{z}\big(\tau(\bar{z}) \wedge \bigwedge_{j\in\{1\dots n\}} \phi_j(z_j)\big)$$

are connected when seen as structures. If this were not the case, let I be the set of indices j such that z_j is in the component of y, $\overline{z_I}$ and τ_I the corresponding fragments of $\bar{z}$ and τ. Let J, $\overline{z_J}$ and τ_J be parts containing the remaining indices. Then

$$\exists \bar{z}\big(\tau(\bar{z}) \wedge z_i = y \wedge \bigwedge_{j\in\{1\dots n\}\backslash\{i\}} \phi_j(z_j)\big)$$

is equivalent to

$$\exists \overline{z_I}\big(\tau_I(\overline{z_I}) \wedge z_i = y \wedge \bigwedge_{j \in I} \phi_j(z_j)\big) \ \wedge \ \exists \overline{z_J}\big(\tau_J(\overline{z_J}) \wedge \bigwedge_{j \in J} \phi_j(z_j)\big)$$

Notice that the right-hand side of the resulting formula is a sentence. Therefore ϕ is equivalent to the disjunction of ϕ_1 and ϕ_2 where ϕ_1 is the conjunction of $\exists \overline{z_J}\big(\tau_J(\overline{z_J}) \wedge \bigwedge_{j \in J} \phi_j(z_j)\big)$ with ϕ where this right-hand side part was replaced with *true*, and ϕ_2 is the conjunction of $\neg \exists \overline{z_J}\big(\tau_J(\overline{z_J}) \wedge \bigwedge_{j \in J} \phi_j(z_j)\big)$ and ϕ where this right-hand side part was replaced by *false*.

In summary, we can assume that ϕ is a Boolean combination of sentences in UN-normal form, satisfying the simplifying assumption, starting with an existential quantifier, and such that all its neighborhood types are connected. Hence it is enough to consider a single such sentence.

Let x be the first existentially quantified variable of ϕ, i.e. ϕ is $\exists x \psi(x)$. By our assumptions on ϕ, all quantified variables of ψ can be taken to range over the neighborhood of x up to distance $|\phi|$. Hence, whether $\psi(x)$ holds or not at a node a of a tree T only depends on the neighborhood of a up to distance $|\phi|$.

In the binary tree case, there are only finitely many such neighborhoods and each of them is implied by the existence of the isomorphism-type of a k-neighborhood for $k = 2|\phi|$. Hence ϕ describes a language in LT.

In the unranked tree case, there are infinitely many such neighborhoods. But, as UNFO is invariant under UN-bisimulation, it is enough to consider those neighborhoods up to UN-bisimulation. Each of the UN-bisimulation classes of these neighborhoods is implied by the existence of the UN-bisimulation-type of a k-neighborhood for $k = 2|\phi|$. By Lemma 6.1 (and the characterization of ILT on unranked trees in terms of global ML), this implies that ϕ describes a language in ILT.   □

We conclude this section by investigating the complexity of satisfiability.

Theorem 6.3. *The satisfiability problem for* UNFO *and for* UNFP *is 2ExpTime-complete on binary trees, on ranked trees, and on XML trees.*

Proof. For the lower bound, recall that the proof of Proposition 4.2 was based on an encoding of Turing machine runs as finite trees. Hence it applies here too.

For the upper bound, we will consider the case of UNFP on XML trees, as all other cases can be seen as a special case of this one. We will describe an exponential-time translation to μRegular XPath, the extension of Core XPath [24] with the Kleene star and with the least fixpoint operator, for which satisfiability on XML trees can decided in ExpTime [15]. We briefly recall the syntax of μRegular XPath (cf. [15] for more details). The language has two sort of expressions, *node expressions* ϕ and *path expressions* α which are defined by mutual recursion:

$$\alpha \ ::= \uparrow \ | \downarrow \ | \leftarrow \ | \rightarrow \ | \ . \ | \ \alpha[\phi] \ | \ \alpha/\beta \ | \ \alpha \cup \beta \ | \ \alpha^*$$

$$\phi \ ::= \ P_i \ | \ \top \ | \ \neg\phi \ | \ \phi \wedge \phi \ | \ \langle \alpha \rangle \ | \ X \ | \ \mu X \phi$$

where, in node expressions of the form $\mu X \phi$, the variable X is required to occur only positively (i.e., under an even number of negations) in ϕ.

Let ϕ be any UNFP-sentence, or, more generally, a UNFP-formula with at most one free variable. We may assume without loss of generality that ϕ is in UN-normal form. In other words, ϕ is built up from atomic formulas using (i) negation, (ii) fixpoint operators, and (iii) existential positive formulas. The translation to μRegular XPath is now by induction. The base case of the induction, as well as the induction step for negation and for the fixpoint operators, is immediate. Now, suppose that ϕ is given by an existential positive formula built from atomic relations and subformulas in one free variable for which we already know that there exists a translation. Here, we can apply the known result from [25, 7] that, on XML trees, every positive existential first-order formula in one free variable (over the given signature) can be translated to a Regular XPath expression (in fact, to a Core XPath expression) in exponential time. Note that when we apply this translation to a UNFP formula, we may treat subformulas in one free variable as unary predicates).   □

7. Discussion

7.1. Logics that are contained in UNFO and UNFP.

We have seen that unary negation logics generalize UCQ, ML, monadic Datalog and μ-calculus. We list here other related formalisms.

7.1.1. *Unary conjunctive view logic.* First-order unary-conjunctive-view logic (UCV) was introduced in [3] as a fragment of FO. A UCV query is an equality-free first-order formula over a signature consisting of unary predicates only, but where each of these unary predicates is in fact a view defined by a unary conjunctive query. It is easy to see that every UCV query can be expressed by a UNFO-formula. Indeed, a quantifier elimination argument (cf. [29]) shows that, over signatures consisting only of monadic relations, every equality-free first-order formula is equivalent to a UNFO formula). UNFO can be viewed as a generalization of UCV where views may be defined in terms of each other (but without cyclic dependencies between view predicates).

7.1.2. *The temporal logic* $\mathrm{CTL}^*(\mathsf{X})$. $\mathrm{CTL}^*(\mathsf{X})$ is the fragment of the temporal logic CTL^* in which only the modal operator X ("next") is allowed. More precisely, the syntax of $\mathrm{CTL}^*(\mathsf{X})$ can be defined as follows:

$$\begin{array}{llll} \text{State formulas:} & \phi & ::= & p \mid \mathsf{E}\alpha \mid \mathsf{A}\alpha \mid \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \\ \text{Path formulas:} & \alpha & ::= & \phi \mid \mathsf{X}\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \neg\alpha \end{array}$$

The semantics is the usual one for CTL^*, and we refer the interested reader to [19]. One can show that there is a polynomial truth-preserving translation from $\mathrm{CTL}^*(\mathsf{X})$ formulas to UNFO formulas. The model checking problem for $\mathrm{CTL}^*(\mathsf{X})$ is known to be complete for the complexity class $\mathrm{P}^{\mathrm{NP}[O(\log^2 n)]}$ [42]. This can be used to provide an alternative proof of the $\mathrm{P}^{\mathrm{NP}[O(\log^2 n)]}$-hardness of the model checking problem of UNFO, cf. Theorem 5.1.

7.1.3. *Description logics.* The basic description logic $\mathcal{ALC}$ (which is a notational variant of the basic multi-modal logic **K**) can be viewed as a fragment of UNFO. The same holds for a number of extensions of $\mathcal{ALC}$. Moreover, the *query answering problem* for these description logics reduces to the entailment problem for UNFO. Recall that the query answering problem is the following problem (cf. [2] for basic terminology):

Given a TBox (i.e., set of concept inclusions) T, an ABox (set of atomic formulas speaking about individuals $c_1 \ldots c_n$) A, a conjunctive query $q(x_1, \ldots, x_k)$, and a tuple of individuals $(c_{i_1}, \ldots, c_{i_k})$, is $(c_{i_1}, \ldots, c_{i_k})$ an answer to q in every model of $T \cup A$?

It is easy to see that this is equivalent to the validity of the UNFO-entailment

$$\phi_T \wedge \bigwedge A[c_1/x_1, \ldots, c_n/x_n] \;\models\; q(x_{i_1}, \ldots, x_{i_k})$$

where ϕ_T is the UNFO-translation of T. This (together with Remark 3.9) gives a new proof of the known result that query answering for $\mathcal{ALC}$ is decidable and that it has the finite model property (cf. [34]). Note that the same argument works not only for $\mathcal{ALC}$ but for any description logic whose TBoxes can be expressed in UNFP. Moreover, the argument works not only for conjunctive queries, but for any class of queries expressible in UNFP.

7.2. **Comparison with guarded logics.** We have already seen in Example 2.1 that UNFO and GFO are incomparable in terms of expressive power: It is easy to show that the GFO formula $\forall xy(R(x, y) \rightarrow S(x, y))$ is not invariant under UN-bisimulations and therefore not expressible in UNFO. On the other hand a simple argument shows that the UNFO formula $\exists yzu(R(x, y) \wedge R(y, z) \wedge R(z, u) \wedge R(u, x))$ is not invariant under guarded-bisimulations and therefore not expressible in GFO.

The decidability and expressibility results obtained in this paper for UNFO and UNFP have many similarities with those of their modal logic counterparts. Actually several proofs are reductions to the modal counterparts. This is in contrast with guarded logics that often require new and difficult arguments. A typical example is the finite model property of GFO whose proof is based on the Herwig Extension Theorem (cf. [26]). In contrast, we prove the finite model property for UNFO by reduction to the analogous result for ML (which has a very simple proof using filtration, cf. [10]).

It is possible to reconcile the unary negation approach and the guarded approach into one logical formalism called *guarded negation logic*, where it is possible to negate a formula if all its free variables are guarded: $R(\overline{x}) \wedge \neg\phi(\overline{x})$. The first-order and fixpoint formalisms obtained this way generalize both the unary negation and guarded approaches and enjoy all the nice properties of UNFO and UNFP [5].

7.3. **Undecidable extensions.** Our results show that UNFO and UNFP are well behaved logics. One may ask if there are extensions that are still well behaved. Inequalities are a minimal form of negation not supported by UNFO. Unfortunately, extending UNFO with inequalities leads to undecidability. Let us denote by $\mathrm{UNFO}^{\neq}$ the extension of UNFO with inequalities, and by $\mathrm{UNFO}^{\neg}$ the extension of UNFO with negative relational atomic formulas. Recall that a fragment of first-order logic is called a *conservative reduction class* if there is a computable map from arbitrary first-order formulas to formulas in the fragment, which preserves (un)satisfiability as well as finite (un)satisfiability.

Theorem 7.1. UNFO$^{\neq}$ and UNFO$^{\neg}$ are conservative reduction classes, and hence undecidable for satisfiability on arbitrary structures and on finite structures.

Proof. It is known [12] that FO^1 with two unary functions and equality is a conservative reduction class. We can translate this logic into UNFO$^{\neq}$ with two binary relations (we can use inequalities to express that the two relations are graphs of functions, as in $\neg\exists xyz(Rxy \wedge Rxz \wedge y \neq z)$, and atomic formulas such as $f(x) = g(x)$ are expressed by $\exists yz(F(x,y) \wedge G(x,z) \wedge y = z)$). This shows that UNFO$^{\neq}$ is a conservative reduction class.

For UNFO$^{\neg}$, we use a similar argument. Let E, F, G be binary relations. Using negative atomic formulas, it is possible to express that E is an equivalence relation, as in $\neg\exists xyz(Exy \wedge Eyz \wedge \neg Exz) \wedge \neg\exists xy(Exy \wedge \neg Eyx) \wedge \neg\exists x \neg Exx$, and that F, G are graphs of functions defined on equivalence classes of E, as in $\neg\exists x \neg\exists y(Fxy) \wedge \neg\exists xx'yy'(Exx' \wedge Fxy \wedge Fx'y' \wedge \neg Eyy')$, and similarly for G. We then use the same reduction as in the case of UNFO$^{\neq}$, except that E takes the role of the equality relation. $\square$

Also, in the fixed point case, one can wonder whether the restriction to *monadic* least fixed-points was necessary. Indeed, this question naturally arises since it is known that the *guarded fragment* of first-order logic is decidable even when extended with (guarded) fixed point operators of arbitrary arity. However, adding non-monadic fixpoint operators to our setting make the logic undecidable.

Theorem 7.2. The extension of UNFP with non-monadic fixed point operators is undecidable for satisfiability on arbitrary structures and on finite structures.

Proof. It was shown in [43] that the containment problem for Datalog is undecidable on finite structures and on arbitrary structures (the result is only stated in [43] for finite structures, but the proof applies to any class of structures that contains all encodings of finite strings, under some encoding).

We reduce this problem to the problem at hand. A containment between two Datalog queries Π_1, Π_2 holds precisely if $\exists \overline{x}(\phi_{\Pi_1}(\overline{x}) \wedge \bigwedge_i P_i(x_i)) \wedge \neg\exists \overline{x}(\phi_{\Pi_2}(\overline{x}) \wedge \bigwedge_i P_i(x_i))$ is unsatisfiable, where ϕ_{Π_i} is the Datalog query Π_i written as a formula of LFP. Note that ϕ_{Π_i} does not contain negation, and therefore belongs to the extension of UNFP with non-monadic fixed point operators. Incidentally, note that the overall reduction uses only Boolean negation (i.e., negation applied to sentences). $\square$

References

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

[2] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, 2003.

[3] J. Bailey, G. Dong, and A.W. To. Logical queries over views: Decidability and expressiveness. *Transactions on Computational Logic*, 11(2):8, 2010.

[4] V. Bárány and M. Bojańczyk. Finite satisfiability for guarded fixpoint logic. *Information Processing Letters*, 112(10):371–375, 2012.

[5] V. Bárány, B. ten Cate, and L. Segoufin. Guarded negation. In *Intl. Coll. on Automata, Languages and Programming (ICALP)*, pages 356–367, 2011.

[6] M. Benedikt, P. Bourhis, and P. Senellart. Monadic Datalog Containment. In *Intl. Coll. on Automata, Languages, and Programming (ICALP)*, pages 79–91, 2012.

[7] M. Benedikt, W. Fan, and G.M. Kuper. Structural properties of XPath fragments. *Theoretical Computer Science*, 336(1):3–31, 2005.

[8] J. van Benthem. *Modal Logic and Classical Logic*. Bibliopolis, 1983.

[9] D. Berwanger and E. Grädel. Games and model checking for guarded logics. In *Intl. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, pages 70–84, 2001.

[10] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2002.

[11] M. Bojańczyk. Two-way alternating automata and finite models. In *Intl. Coll. on Automata, Languages and Programming (ICALP)*, pages 833–844. Springer, 2002.

[12] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Springer, Berlin, 1997.

[13] S.R. Buss and H. Louise. On truth-table reducibility to SAT. *Information and Computation*, 91(1):86–102, 1991.

[14] J. Castro and C. Seara. Complexity classes between θ_k^p and δ_k^p. *Informatique Théorique et Applications*, 30(2):101–121, 1996.

[15] B. ten Cate. The expressivity of XPath with transitive closure. In *Symp. on Principles of Database Systems (PODS)*, pages 328–337, 2006.

[16] B. ten Cate and M. Marx. Navigational XPath: calculus and algebra. *SIGMOD Record*, 36(2):19–26, 2007.

[17] B. ten Cate and L. Segoufin. Unary negation. In *Intl. Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 344–355, 2011.

[18] C.C. Chang and H.J. Keisler. *Model Theory*. Studies in Logic and the Foundations of Mathematics. Elsevier Science, 1990.

[19] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model checking*. MIT Press, 1999.

[20] S. Cosmadakis, H. Gaifman, P. Kanellakis, and M.Y. Vardi. Decidable optimization problems for database logic programs. In *Symp. on Theory of Computing (STOC)*, pages 477–490, 1988.

[21] M. Fisher and R. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18:194–211, 1979.

[22] G. Gottlob. NP trees and Carnap's modal logic. *Journal of the ACM*, 42(2):421–457, 1995.

[23] G. Gottlob and C. Koch. Monadic datalog and the expressive power of languages for web information extraction. *Journal of the ACM*, 51(1):74–113, 2004.

[24] G. Gottlob, C. Koch, and R. Pichler. Efficient algorithms for processing XPath queries. *ACM Transaction on Database Systems*, 30(2):444–491, 2005.

[25] G. Gottlob, C. Koch, and K. Schulz. Conjunctive queries over trees. In *Symp. Principles of Database Systems (PODS)*, pages 189–200, 2004.

[26] E. Grädel. On the restraining power of guards. *J. on Symbolic Logic*, 64(4):1719–1742, 1999.

[27] E. Grädel. Why are modal logics so robustly decidable? In *Current Trends in Theoretical Computer Science*, pages 393–408. World Scientific, 2001.

[28] E. Grädel and I. Walukiewicz. Guarded fixed point logic. In *Symp. on Logic in Computer Science (LICS)*, pages 45–54, 1999.

[29] W. Hodges. *Model Theory*. Cambridge University Press, 1993.

[30] E. Hoogland and M. Marx. Interpolation and definability in guarded fragments. *Studia Logica*, 70(3):373–409, 2002.

[31] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional μ-calculus w.r.t. monadic second-order logic. In *Intl. Conf. on Concurrency Theory (CONCUR)*, 1996.

[32] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333–354, 1983.

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

[34] C. Lutz. The complexity of conjunctive query answering in expressive description logics. In *Intl. Joint Conf. on Automated Reasoning (IJCAR)*, pages 179–193, 2008.

[35] M. Marx and M. de Rijke. Semantic characterizations of navigational XPath. *SIGMOD Record*, 34(2):41–46, 2005.

[36] M. Marx and Y. Venema. *Multidimensional Modal Logic*. Kluwer, 1997.

[37] D. Niwinski. On fixed-point clones. In *Intl. Coll. on Automata, Languages and Programming (ICALP)*, pages 464–473, 1986.

[38] M. Otto. Modal and guarded characterisation theorems over finite transition systems. *Annals of Pure and Applied Logic*, 130(1–3):173 – 205, 2004.

[39] M. Otto. Highly acyclic groups, hypergraph covers, and the guarded fragment. *J. ACM*, 59(1):5, 2012.

[40] T. Place and L. Segoufin. A decidable characterization of locally testable tree languages. In *Intl. Coll. on Automata, Languages and Programming (ICALP)*, pages 285–296, 2009.

[41] E. Rosen. Modal logic over finite structures. *Journal of Logic, Language, and Computation*, 6(4):427–439, 1997.

[42] P. Schnoebelen. Oracle circuits for branching-time model checking. In *Intl. Coll. on Automata, Languages and Programming (ICALP)*, pages 187–187, 2003.

[43] O. Shmueli. Equivalence of datalog queries is undecidable. *Journal of Logic Programming*, 15(3):231–241, 1993.

[44] R.S. Streett. Propositional dynamic logic of looping and converse. *Information and Control*, 54:121–141, 1982.

[45] M.Y. Vardi. On the complexity of bounded-variable queries. In *Symp. on Principles of Database Systems (PODS)*, pages 266–276, 1995.

[46] M.Y. Vardi. Why is modal logic so robustly decidable? In *Descriptive Complexity and Finite Models*, pages 149–184. American Mathematical Society, 1996.

[47] M.Y. Vardi. Reasoning about the past with two-way automata. In *Intl. Coll. on Automata, Languages and Programming (ICALP)*, pages 628–641, 1998.

[48] K.W. Wagner. More Complicated Questions about Maxima and Minima, and some Closures of NP. *Theoretical Computer Science*, 51(1-2):53 – 80, 1987.

[49] K.W. Wagner and G. Wechsung. *Computational Complexity*. Verlag der Wissenschaften, Berlin, 1986.