Interpolation with Decidable Fixpoint Logics

Michael Benedikt
University of Oxford

Balder ten Cate
UC Santa Cruz and LogicBlox Inc

Michael Vanden Boom
University of Oxford

***Abstract*—A logic satisfies Craig interpolation if whenever one formula ϕ_1 in the logic entails another formula ϕ_2 in the logic, there is an intermediate formula — one entailed by ϕ_1 and entailing ϕ_2 — using only relations in the common signature of ϕ_1 and ϕ_2. Uniform interpolation strengthens this by requiring the interpolant to depend only on ϕ_1 and the common signature. A uniform interpolant can thus be thought of as a minimal upper approximation of a formula within a subsignature. For first-order logic, interpolation holds but uniform interpolation fails. Uniform interpolation is known to hold for several modal and description logics, but little is known about uniform interpolation for fragments of predicate logic over relations with arbitrary arity. Further, little is known about ordinary Craig interpolation for logics over relations of arbitrary arity that have a recursion mechanism, such as fixpoint logics.**

In this work we take a step towards filling these gaps, proving interpolation for a decidable fragment of least fixpoint logic called unary negation fixpoint logic. We prove this by showing that for any fixed k, uniform interpolation holds for the k-variable fragment of the logic. In order to show this we develop the technique of reducing questions about logics with tree-like models to questions about modal logics, following an approach by Grädel, Hirsch, and Otto. While this technique has been applied to expressivity and satisfiability questions before, we show how to extend it to reduce interpolation questions about such logics to interpolation for the μ-calculus.

I. Introduction

A desirable property of a logic is *interpolation*: if one formula ϕ_1 entails another ϕ_2, there is a formula in the common signature that is entailed by the first and entails the second. It implies the well-known *Beth Definability Property* as well as its extension, the *Projective Beth Definability Property* stating that implicit specifications can be converted to explicit ones, which is important in knowledge representation [1] and for rewriting queries in terms of views in databases [2], [3]. Interpolation has many other applications, both in simplifying definitions and in verification [4].

It is even more desirable to have *effective* interpolation: given ϕ_1 and ϕ_2, one can determine if entailment holds and compute an interpolant if it does. An even stronger goal is to prove effective *uniform interpolation* theorems. In uniform interpolation, the interpolating formula depends only on ϕ_1 and the signature of ϕ_2. Uniform interpolation can be thought of as stating that ϕ_1 has a minimal approximation from above in the signature of ϕ_2.

The first interpolation result was proven for first-order logic by Craig [5]; the undecidability of validity in first-order logic implies that this cannot be made effective in the sense above. Effective interpolation results were later shown for fragments of first-order logic, such as the guarded negation fragment [6], as well as a number of modal and description logics. On the negative side, it is known that first-order logic does not have uniform interpolation [7].

The situation is much less clear for logics that contain recursion. The standard way to capture recursion in a logic is via a fixpoint operator. One adds second-order variables $X, Y, \ldots$ which are allowed in atomic formulas. These variables are bound not by second-order quantifiers, but by $[\mathbf{lfp}\, X, \boldsymbol{x}.\phi(X, \boldsymbol{x}, \boldsymbol{y})]$ where X is a second-order variable of arity $|\boldsymbol{x}|$. In the setting of modal logic, the addition of fixpoints gives one the μ-calculus, which can express important reachability properties of a labelled transition system. The greater expressiveness of fixpoint logics makes them a natural candidate for the stronger uniform interpolation property. D'Agostino and Hollenberg [8] showed that in fact the μ-calculus has uniform interpolation. Uniform interpolation has also been shown for other modal and description logics [9], [10]. Little is known about interpolation, uniform or otherwise, effective or otherwise, for fixpoint logics over general relational structures, where relations can have arbitrary arity.

Effective interpolation results are relevant only for logics where entailment is decidable. This is not the case for LFP, the fixpoint extension of first-order logic. However, several decidable fragments of LFP are known, including *guarded fixpoint logic* (GFP), *guarded negation fixpoint logic* (GNFP), and *unary negation fixpoint logic* (UNFP). All of these achieve decidability by restricting the use of either quantification or negation.

Our main result is effective uniform interpolation for unary fixpoint queries of some fixed *width* k, denoted UNFP^k (the width of a UNFP formula is, roughly, a bound on the maximal number of free variables in any subformula). In particular, this result provides a decidable extension of the description logic $\mathcal{ALCO}$ having effective uniform interpolation with respect to relation symbols. This contrasts with an earlier conjecture in [10, Section 7] that no such extension could have uniform interpolation with respect to both relations and constants.

From effective uniform interpolation for UNFP^k, we can deduce effective Craig interpolation for the full unary negation fixpoint language UNFP. The fact that we have effective interpolation for UNFP (and hence, e.g., the ability to effectively convert implicit to explicit definitions) is significant in that UNFP is quite expressive, subsuming recursive query languages such as Monadic Datalog (with stratified negation).

Our approach for proving uniform interpolation will work via reducing logics that always admit "tree-like models" — such as the guarded and unary negation fragments — to modal logics. This technique was introduced by Grädel, Hirsch and

Otto [11]. In this approach we start with an input problem in some logic over unrestricted structures in some relational signature. Making use of the tree-like model property for the logic, we apply a "forward mapping" that translates to a corresponding problem in modal logics. Given a solution to the problem in the modal logic setting, we then apply a "backward mapping" to get a corresponding solution back in the setting of the original logic. [11] applied this technique to prove that GFP is precisely the "guarded-bisimulation" invariant fragment of a "guarded" second-order logic, in analogy to the fact that the μ-calculus is the bisimulation-invariant fragment of monadic second-order logic [12]. We will apply it to interpolation, using as our solution in the modal setting the interpolation result of D'Agostino and Hollenberg mentioned above.

We supplement our effective interpolation theorems with several negative results, showing the limitations of interpolation. In particular, we show that one cannot hope to extend the uniform interpolation result to full UNFP, and one can not extend even ordinary interpolation to the larger language of guarded negation fixpoint logic.

Due to space limitations, most proofs are deferred to the full version of this paper.

II. Preliminaries

A. Notation and conventions

We use $\boldsymbol{x}, \boldsymbol{y}, \ldots$ (respectively, $\boldsymbol{X}, \boldsymbol{Y}, \ldots$) to denote vectors of first-order (respectively, second-order) variables. For a formula ϕ, we write $\phi(\boldsymbol{x})$ to indicate that the free first-order variables in ϕ are among $\boldsymbol{x}$. If we want to emphasize that there are also free second-order variables $\boldsymbol{X}$, we write $\phi(\boldsymbol{x}, \boldsymbol{X})$. We often use α to denote atomic formulas, and for such formulas, if we write $\alpha(\boldsymbol{x})$ then we assume that the free variables in α are precisely $\boldsymbol{x}$.

A formula ϕ is assumed to be given in the standard tree representation of a formula, and the *size* of ϕ, denoted $|\phi|$, is the number of symbols in ϕ. We will sometimes represent ϕ using a node-labelled DAG (directed acyclic graph). The nodes represent formulas, and the edge relation connects a formula to its subformulas. The size of a DAG representation is the number of nodes and edges in the DAG.

B. Basics of unary negation and guarded logics

The *Unary Negation Fragment* of FO [13] (denoted UNF) is built up inductively according to the grammar:

$$\phi ::= R\boldsymbol{t} \mid \exists x.\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi(x)$$

where R is either a relation symbol or the equality relation, and $\boldsymbol{t}$ is a tuple over variables and constants. Notice that any use of negation must occur only on formulas with at most one free variable. The *Guarded Negation Fragment* GNF extends UNF by allowing $\alpha(\boldsymbol{x}) \wedge \neg\phi(\boldsymbol{x})$, where α is an atomic relation or equality that contains all of the free variables of the negated formula. Such an atomic relation is a *guard* of the formula. GNF is also related to the *Guarded Fragment* (GF), typically defined via the grammar:

$$\phi ::= R\boldsymbol{t} \mid \exists \boldsymbol{x}.(\alpha(\boldsymbol{xy}) \wedge \phi(\boldsymbol{xy})) \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi(\boldsymbol{x})$$

where R is either a relation symbol or the equality relation, α is an atomic relation or equality relation, and $\boldsymbol{t}$ is a tuple over variables and constants. Here it is the quantification that is guarded, rather than negation. As in GNF, we allow equality guards by default. Note that GNF subsumes GF sentences and UNF formulas.

The fixpoint extensions of these logics (denoted GNFP, UNFP, and GFP) extend the base logic by adding to the grammars the additional production rule

$$[\mathbf{lfp}\, X, \boldsymbol{x}.\, \mathrm{gdd}(\boldsymbol{x}) \wedge \phi(\boldsymbol{x}, X, \boldsymbol{Y})](\boldsymbol{t})$$

where (i) X only appears positively in ϕ, (ii) second-order variables like X cannot be used as guards and (iii) $\mathrm{gdd}(\boldsymbol{x})$ expresses that $\boldsymbol{x}$ is guarded by an atom from the original signature (it can be understood as an abbreviation for the disjunction of existentially quantified relational atoms involving all of the variables in $\boldsymbol{x}$).[1] In UNFP, there is an additional requirement that $\boldsymbol{x}$ has at most one variable, so only unary or 0-ary predicates can be defined. GNFP subsumes both GFP sentences and UNFP formulas. These logics are all contained in LFP, the fixpoint extension of FO.

The semantics for the existential quantifier and boolean connectives is standard. We briefly review the semantics of $[\mathbf{lfp}\, X, \boldsymbol{x}.\, \mathrm{gdd}(\boldsymbol{x}) \wedge \phi(\boldsymbol{x}, X, \boldsymbol{Y})](\boldsymbol{t})$. Since $\phi(\boldsymbol{x}, X, \boldsymbol{Y})$ is monotone in X, it induces an operator $U \mapsto \mathcal{O}_{\phi}^{\mathfrak{A},\boldsymbol{V}}(U) := \{\boldsymbol{a} : \mathfrak{A}, U, \boldsymbol{V} \models \mathrm{gdd}(\boldsymbol{a}) \wedge \phi(\boldsymbol{a}, X, \boldsymbol{Y})\}$ on every structure $\mathfrak{A}$ with valuation $\boldsymbol{V}$ for $\boldsymbol{Y}$, and this operator has a least fixpoint. Given some ordinal β, the *fixpoint approximant* $\phi^{\beta}(\mathfrak{A}, \boldsymbol{V})$ of ϕ on $\mathfrak{A}, \boldsymbol{V}$ is defined such that

$$\begin{aligned}\phi^{0}(\mathfrak{A}, \boldsymbol{V}) &:= \emptyset \\ \phi^{\beta+1}(\mathfrak{A}, \boldsymbol{V}) &:= \mathcal{O}_{\phi}^{\mathfrak{A},\boldsymbol{V}}(\phi^{\beta}(\mathfrak{A}, \boldsymbol{V})) \\ \phi^{\beta}(\mathfrak{A}, \boldsymbol{V}) &:= \bigcup_{\beta' < \beta} \phi^{\beta'}(\mathfrak{A}, \boldsymbol{V}) \quad \text{where } \beta \text{ is a limit ordinal.}\end{aligned}$$

We let $\phi^{\infty}(\mathfrak{A}, \boldsymbol{V}) := \bigcup_{\beta} \phi^{\beta}(\mathfrak{A}, \boldsymbol{V})$ denote the least fixpoint based on this operation. Thus, $[\mathbf{lfp}\, X, \boldsymbol{x}.\, \mathrm{gdd}(\boldsymbol{x}) \wedge \phi(\boldsymbol{x}, X, \boldsymbol{Y})]$ defines a new predicate named X of arity $|\boldsymbol{x}|$, and $\mathfrak{A}, \boldsymbol{V}, \boldsymbol{a} \models [\mathbf{lfp}\, X, \boldsymbol{x}.\, \mathrm{gdd}(\boldsymbol{x}) \wedge \phi(\boldsymbol{x}, X, \boldsymbol{Y})](\boldsymbol{x})$ iff $\boldsymbol{a} \in \phi^{\infty}(\mathfrak{A}, \boldsymbol{V})$. If $\boldsymbol{V}$ is empty or understood in context, we just write $\phi^{\infty}(\mathfrak{A})$.

We also write ϕ^{β} for the fixpoint approximations obtained by unfolding the fixpoint β times. That is, $\phi^{0} := \bot$, $\phi^{\beta+1}(\boldsymbol{x}) := \mathrm{gdd}(\boldsymbol{x}) \wedge \phi[\phi^{\beta}(\boldsymbol{y})/X(\boldsymbol{y})]$, and if β is a limit ordinal, $\phi^{\beta} := \bigvee_{\beta' < \beta} \phi^{\beta'}$. This formula defines the β-approximation of the fixpoint process based on $[\mathbf{lfp}\, X, \boldsymbol{x}.\, \mathrm{gdd}(\boldsymbol{x}) \wedge \phi(\boldsymbol{x}, X, \boldsymbol{Y})]$. In general these formulas live in an infinitary version of the logic that allows conjunctions and disjunctions over arbitrary sets of formulas. However, for the logics that we are considering, if β is finite, then ϕ^{β} remains in the same logic.

[1]In GFP and UNFP, omitting $\mathrm{gdd}(\boldsymbol{x})$ does not change the expressivity of the logic; however, in GNFP this guardedness condition must be explicitly enforced.

We will sometimes utilize *simultaneous fixpoints*, or *vectorial fixpoints*, of the form $[\mathbf{lfp}\, X_i, \boldsymbol{x}_i.S](\boldsymbol{t})$ where

$$S = \begin{cases} X_1, \boldsymbol{x}_1 := \mathrm{gdd}(\boldsymbol{x}_1) \wedge \phi_1(\boldsymbol{x}_1, X_1, \dots, X_j, \boldsymbol{Y}) \\ \vdots \\ X_j, \boldsymbol{x}_j := \mathrm{gdd}(\boldsymbol{x}_j) \wedge \phi_j(\boldsymbol{x}_j, X_1, \dots, X_j, \boldsymbol{Y}) \end{cases}$$

is a system of formulas ϕ_i where $X_1, \dots, X_j$ occur positively, and satisfy the same requirements for the body of the fixpoint formulas as before. Such a system can be viewed as defining a monotone operation on vectors of relations, and $[\mathbf{lfp}\, X_i, \boldsymbol{x}_i.S](\boldsymbol{t})$ expresses that $\boldsymbol{t}$ is a tuple in the i-th component of the least fixpoint defined by this operation. Allowing simultaneous fixpoints does not change the expressivity of these logics since they can be eliminated in favor of traditional fixpoints using the Bekič principle [14], with a possible exponential blow-up in the size of the formula, and only a polynomial blow-up if a DAG-representation is used.

These logics are expressive: modal logic is contained in each of these logics (even without fixpoints), every *union of conjunctive queries* (UCQ) is expressible in UNF and GNF, and every GF sentence can be expressed in GNF [15]. Nevertheless, these logics are decidable and have nice model theoretic properties (see Theorem 2). In the first part of the paper we will focus primarily on UNFP. This includes Monadic Datalog and its extension with stratified negation. UNFP can also be viewed as an expressive generalization of many description logics: the concept language of many description logics is contained in UNFP. In particular, this holds for $\mathcal{ALCIO}_{reg}$, the extension of the basic description logic $\mathcal{ALC}$ with inverse roles, individuals, and the regular role operators [16]. UNFP also subsumes the hybrid μ-calculus, which contains some description logics [17].

We pause to give a simple example of UNFP expressing a reachability property.

Example 1. *Consider structures over a single unary relation P and a binary relation R. Then*

$$\neg\exists y. ((y = y) \wedge \neg[\mathbf{lfp}\, Y, y.Py \vee \exists z.(Ryz \wedge Yz)](y))$$

expresses that every element can R-reach an element where P holds. This sentence is in UNFP *and* GFP.

It is often helpful to consider the formulas in a normal form. For instance, UNFP formulas in *normal form* can be generated using the following grammar:

$$\phi ::= \bigvee_i \exists \boldsymbol{x}. \bigwedge_j \psi_{ij} \mid [\mathbf{lfp}\, X, x.\phi(x, X, \boldsymbol{Y})](t)$$
$$\psi ::= R\,\boldsymbol{t} \mid X\,t \mid \phi(x) \mid \neg\phi(x).$$

The idea is that each UNFP formula in normal form is built from fixpoint predicates and UCQ-shaped formulas, where each conjunct in a CQ-shaped subformula is an atom, a normalized subformula with at most one free variable, or the negation of a normalized subformula with at most one free variable. Every UNFP-formula ϕ can be converted into this form in a canonical way, with an exponential blow-up in size. The *width* of ϕ is the maximum number of free variables in any subformula of its equivalent normal form (obtained in this canonical way). We remark that this is different than the normal form used in [13], but the width of a UNFP-formula under the two definitions is identical. We denote by UNFP^k the set of UNFP-*formulas of width at most* k.

The following theorem summarizes the decidability and model theoretic results about UNFP and GNFP that we will make use of.

Theorem 2 ([13],[15])**.** *Satisfiability and finite satisfiability are* 2EXPTIME*-complete for* GNF *and* GNFP *(and hence for* UNF *and* UNFP*).*

GNF *(and hence* UNF*) has the finite-model property: if ϕ is satisfiable, then ϕ is satisfiable in a finite structure. This does not hold for* UNFP *or* GNFP*.*

GNFP *(and hence* UNFP*) has the tree-like model property: if ϕ is satisfiable, then ϕ is satisfiable over structures of bounded tree-width (in fact, tree-width* $\mathrm{width}(\phi) - 1$*).*

C. Interpolation

Given a logic $\mathcal{L}$ and a relational signature σ, possibly including constants, we write $\mathcal{L}[\sigma]$ to denote the logic over this signature σ. We write $\mathrm{rel}(\sigma)$ for the set of relations and $\mathrm{con}(\sigma)$ for the set of constants in σ.

For formula φ_{L} and φ_{R} over signatures σ_{L} and σ_{R}, we write $\varphi_{\mathrm{L}} \models \varphi_{\mathrm{R}}$ (φ_{L} *entails* φ_{R}) if every model of the antecedent φ_{L} is a model of the consequent φ_{R}, and we say this entailment is a *validity*.

An *interpolant* for such a validity is a formula θ for which $\varphi_{\mathrm{L}} \models \theta$ and $\theta \models \varphi_{\mathrm{R}}$, and θ mentions only relations in $\mathrm{rel}(\sigma_{\mathrm{L}}) \cap \mathrm{rel}(\sigma_{\mathrm{R}})$.

We say a logic $\mathcal{L}$ has *Craig interpolation* if any validity $\varphi_{\mathrm{L}} \models \varphi_{\mathrm{R}}$ for φ_{L} and φ_{R} in $\mathcal{L}$, has an interpolant θ in $\mathcal{L}$. Craig famously proved that FO has Craig interpolation [5]. In [6], the authors demonstrated that GNF even has *effective Craig interpolation*: the GNF interpolant can be constructed from the GNF validity.

We say $\mathcal{L}$ has *uniform interpolation* if given some φ_{L} over signature σ_{L} and some subsignature σ' of σ_{L}, there is a formula θ in $\mathcal{L}$ that is an interpolant for all validities $\varphi_{\mathrm{L}} \models \varphi_{\mathrm{R}}$ with $\mathrm{rel}(\varphi_{\mathrm{L}}) \cap \mathrm{rel}(\varphi_{\mathrm{R}}) \subseteq \mathrm{rel}(\sigma')$. That is, the interpolant depends only on the antecedent and the common signature, rather than on the particular consequent. It is clear that uniform interpolation implies Craig interpolation.

D. Main results

The main result of this work is the following:

Theorem 3 (Uniform interpolation for UNFP^k)**.** *Let φ_{L} be a sentence in* UNFP$^k[\sigma]$*. Then for all signatures σ' with $\mathrm{rel}(\sigma') \subseteq \mathrm{rel}(\sigma)$ and $\mathrm{con}(\sigma') = \mathrm{con}(\sigma)$, there is a sentence $\chi \in$* UNFP$^k[\sigma']$ *such that*

- $\varphi_{\mathrm{L}} \models \chi$*, and*
- $\chi \models \varphi_{\mathrm{R}}$ *for all sentences $\varphi_{\mathrm{R}} \in$* UNFP$^k[\sigma_{\mathrm{R}}]$ *where $\varphi_{\mathrm{L}} \models \varphi_{\mathrm{R}}$ and $\mathrm{rel}(\sigma) \cap \mathrm{rel}(\sigma_{\mathrm{R}}) \subseteq \mathrm{rel}(\sigma')$ and $\mathrm{con}(\sigma_{\mathrm{R}}) \subseteq \mathrm{con}(\sigma')$.*

Furthermore, a DAG-representation for χ of size at most doubly exponential in $|\varphi_{\mathrm{L}}|$ can be constructed in 2EXPTIME.

We emphasize that the subsignature σ' for the uniform interpolant can only restrict the relations, not the constants, in σ. We discuss this more in Section IV-B.

An immediate corollary of Theorem 3 is that UNFP has Craig interpolation.

Corollary 4 (Craig interpolation for UNFP)**.** *Let $\varphi_{\mathrm{L}} \in \mathrm{UNFP}[\sigma_{\mathrm{L}}]$ and $\varphi_{\mathrm{R}} \in \mathrm{UNFP}[\sigma_{\mathrm{R}}]$ be sentences such that $\varphi_{\mathrm{L}} \models \varphi_{\mathrm{R}}$. Then there is a sentence $\chi \in \mathrm{UNFP}^K[\sigma']$ such that $\varphi_{\mathrm{L}} \models \chi$ and $\chi \models \varphi_{\mathrm{R}}$, where $\mathrm{rel}(\sigma') := \mathrm{rel}(\sigma_{\mathrm{L}}) \cap \mathrm{rel}(\sigma_{\mathrm{R}})$, $\mathrm{con}(\sigma') := \mathrm{con}(\sigma_{\mathrm{L}}) \cup \mathrm{con}(\sigma_{\mathrm{R}})$, and $K := \max\{\mathrm{width}(\varphi_{\mathrm{L}}), \mathrm{width}(\varphi_{\mathrm{R}})\}$. Furthermore, a DAG-representation for χ of size at most doubly exponential in the size of φ_{L} and φ_{R} can be constructed in* 2EXPTIME.

E. Building blocks

In order to prove our results we will make use of GSO, a rich logic extending UNFP. We will also use two logics, MSO and L_μ, that will be interpreted over restricted signatures, so we review here some important prior results.

Guarded second-order logic. *Guarded second-order logic* (denoted GSO) over a signature σ is a fragment of second-order logic in which second-order quantification is interpreted only over guarded relations, i.e. over relations where every tuple in the relation is guarded by some predicate from σ. We refer the interested reader to [11] for more background and some equivalent definitions of this logic.

The logics UNFP, GNFP, and GFP considered in this paper can all be translated into GSO.

Proposition 5. *Given $\phi \in \mathrm{GNFP}[\sigma]$, we can construct an equivalent $\phi' \in \mathrm{GSO}[\sigma]$.*

Proof sketch: By structural induction on ϕ. The interesting case is for the least fixpoint. If $\phi(\boldsymbol{y}) = [\mathbf{lfp}\, X, \boldsymbol{x}.\, \mathrm{gdd}(\boldsymbol{x}) \wedge \psi(X, \boldsymbol{x})](\boldsymbol{y})$ then

$$\phi'(\boldsymbol{y}) := \forall X.[(\forall \boldsymbol{x}.((\mathrm{gdd}(\boldsymbol{x}) \wedge \psi'(X, \boldsymbol{x})) \rightarrow X\boldsymbol{x})) \rightarrow X\boldsymbol{y}]$$

where second-order quantifiers range over guarded relations. ■

Transition systems and their logics. A special kind of signature is a *transition system signature*, of the form $\tilde{\sigma}$ consisting of unary predicates (corresponding to a set of propositions $\mathit{props}(\tilde{\sigma})$) and binary predicates (corresponding to a set of actions $\mathit{actions}(\tilde{\sigma})$). A structure for such a signature is a *transition system.* Trees allowing both edge-labels and node-labels have a natural interpretation as transition systems.

We will be interested in two logics over transition systems signatures. One is *monadic second-order logic* (denoted MSO) — where second-order quantification is only over unary relations. MSO is contained in GSO, because unary relations are trivially guarded.

While MSO and GSO can be interpreted over arbitrary signatures, there are logics that have syntax specific to transition system signatures. One is the *modal μ-calculus* (denoted L_μ), an extension of modal logic with fixpoints. Given a transition system signature $\tilde{\sigma}$, formulas $\phi \in \mathrm{L}_\mu[\tilde{\sigma}]$ can be generated using the grammar

$$\phi ::= P \mid X \mid \phi \wedge \phi \mid \neg\phi \mid \langle\rho\rangle\phi \mid \mu X.\phi$$

where $P \in \mathit{props}(\tilde{\sigma})$, $\rho \in \mathit{actions}(\tilde{\sigma})$. The formulas $\mu X.\phi$ are required to use the variable X only positively in ϕ, and the semantics define a least-fixpoint operation based on ϕ. For instance, $\mu Y.(P \vee \langle E_R\rangle Y) \in \mathrm{L}_\mu$ holds at an element s in a transition system over a binary relation R and unary relation P iff s can R-reach an element where P holds. As usual, we refer to $\langle\rho\rangle\phi$ as a diamond modality, and $\mu X.\phi$ as a least fixpoint. Using negation, we also have the dual operators, the box modality $[\rho]\phi$ and the greatest fixpoint $\nu X.\phi$. We refer the reader to [18] for the formal semantics, and a survey of results. It is easy to see that L_μ can be translated into MSO.

Bisimulation games and logics. The logic L_μ applies to transition system signatures, and lies within MSO. Similarly the logics UNFP and GSO apply to arbitrary-arity signatures, with UNFP lying within GSO. It is easy to see that in both cases the containment is proper. In each case, what distinguishes the smaller logic from the larger is *invariance* under certain equivalences, called bisimulations.

For instance, the classical *bisimulation game* between transition systems $\mathfrak{A}$ and $\mathfrak{B}$ defines an equivalence relation over structures for a transition system signature $\tilde{\sigma}$. Positions in the game consist of pairs $(a, b) \in \mathrm{dom}(\mathfrak{A}) \times \mathrm{dom}(\mathfrak{B})$ that agree on all propositions in $\mathit{props}(\tilde{\sigma})$. From such a position (a, b), Spoiler first chooses which structure to move in, say $\mathfrak{A}$ (the choice of $\mathfrak{B}$ is symmetric). Spoiler then selects some a' with $(a, a') \in E_\rho^{\mathfrak{A}}$; if this is not possible, then the game terminates, and Duplicator wins. Duplicator must then choose some b' with $(b, b') \in E_\rho^{\mathfrak{B}}$ such that a' and b' agree on all propositions in $\mathit{props}(\tilde{\sigma})$. If this is not possible, the game terminates and Duplicator loses. Otherwise, the game proceeds from (a', b'). If the game never terminates then Duplicator wins. If Duplicator has a winning strategy in the bisimulation game starting from (a, b), we say $\mathfrak{A}, a$ and $\mathfrak{B}, b$ are *bisimilar*. We say two tree structures are bisimilar if they are bisimilar from their roots.

We say a formula ϕ is *$\tilde{\sigma}$-bisimulation-invariant* if it does not distinguish between $\tilde{\sigma}$-bisimilar transition systems. It is straightforward to check that $\mathrm{L}_\mu[\tilde{\sigma}]$-formulas are $\tilde{\sigma}$-bisimulation invariant. We will make use of a stronger result of Janin and Walukiewicz [12] that the μ-calculus is the bisimulation-invariant fragment of MSO (we state it here for trees because of how we use this later).

Theorem 6 ($\mathrm{L}_\mu \equiv$ bisimulation invariant MSO [12])**.** *Let $\tilde{\sigma}$ be a transition system signature. A class of trees is definable in $\mathrm{L}_\mu[\tilde{\sigma}]$ iff it is definable in $\mathrm{MSO}[\tilde{\sigma}]$ and closed under $\tilde{\sigma}$-bisimulation within the class of all $\tilde{\sigma}$-trees. Moreover, the translation between these logics is effective.*

There are variants of bisimulation and bisimulation games for the guarded logics mentioned earlier (see [19] and [15]). We describe here the UN^k*-bisimulation game* between σ-structures $\mathfrak{A}$ and $\mathfrak{B}$ corresponding to $\mathrm{UN}^k[\sigma]$*-bisimulation of width* k. Let $\mathfrak{A}$ and $\mathfrak{B}$ be σ-structures. We say f is a *partial unary-rigid homomorphism* if f is a partial homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$ or $\mathfrak{B}$ to $\mathfrak{A}$ (relative to σ), and the restriction of f to any single element in its domain is a partial isomorphism.

Positions in the $\mathrm{UN}^k[\sigma]$ bisimulation game between $\mathfrak{A}$ and $\mathfrak{B}$ are partial unary-rigid homomorphisms f with $|\mathrm{dom}(f)| \leq k$. In general, we call these *bag positions*, since they represent a bag of at most k elements from each structure. In the special case when $|\mathrm{dom}(f)| \leq 1$, f is also called an *interface position*. We say the *active structure* is the structure containing $\mathrm{dom}(f)$.

The initial position is an empty partial homomorphism (an interface position). Starting in an interface position f, one round of the game consists of the following:

- Spoiler selects k elements $\boldsymbol{d}$ in the active structure (these elements need not be distinct, but they must include any elements in $\mathrm{dom}(f)$);
- Duplicator chooses $\boldsymbol{d}'$ in the other structure such that $g : \boldsymbol{d} \mapsto \boldsymbol{d}'$ is a partial unary-rigid homomorphism consistent with f (i.e. $f(c) = g(c)$ for all $c \in \mathrm{dom}(f)$).

Duplicator immediately loses if this is not possible. Otherwise, the game proceeds from the bag position g. Starting in a bag position g, one round of the game consists of the following:

- Spoiler collapses to at most one element $d \in \mathrm{dom}(g)$, and chooses h to be $d \mapsto g(d)$ or $g(d) \mapsto d$ (in case Spoiler collapses to the empty set, h is an empty partial homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$ or vice versa).

The game proceeds from the interface position h.

If Duplicator has a winning strategy in this game, then we say that $\mathfrak{A}$ and $\mathfrak{B}$ are $\mathrm{UN}^k[\sigma]$-*bisimilar*. We say a sentence ϕ is $\mathrm{UN}^k[\sigma]$-*bisimulation invariant* if it does not distinguish between $\mathrm{UN}^k[\sigma]$-bisimilar structures. That is, if $\mathfrak{A}$ is $\mathrm{UN}^k[\sigma]$-bisimilar to $\mathfrak{B}$, then $\mathfrak{A} \models \phi$ iff $\mathfrak{B} \models \phi$. It is straightforward to show that $\mathrm{UNFP}^k[\sigma]$ sentences are $\mathrm{UN}^k[\sigma]$-bisimulation invariant.

Proposition 7. *Assume Duplicator has a winning strategy in the* $\mathrm{UN}^k[\sigma]$*-bisimulation game between* $\mathfrak{A}$ *and* $\mathfrak{B}$. *If* φ *is a* $\mathrm{UNFP}^k[\sigma]$ *sentence in normal form, then* $\mathfrak{A} \models \varphi$ *iff* $\mathfrak{B} \models \varphi$.

In the course of this work, we will prove a converse to this, an analog of the Janin-Walukiewicz theorem, showing that $\mathrm{UNFP}^k[\sigma]$ captures the $\mathrm{UN}^k[\sigma]$-bisimulation invariant subset of GSO (see Theorem 24).

Interpolation over transition systems. A motivation for this work was the main prior example of effective uniform interpolation for a fixpoint logic, D'Agostino and Hollenberg's result that L_μ has uniform interpolation [8]. Although it is not emphasized in their work, the proof they describe yields effective uniform interpolation.

Theorem 8 (Uniform interpolation for L_μ [8])**.** *Let* $\tilde{\sigma}$ *be a transition system signature, and let* $\tilde{\varphi}_{\mathrm{L}} \in \mathrm{L}_\mu[\tilde{\sigma}]$. *Then for all signatures* $\tilde{\sigma}'$ *such that* $\mathrm{rel}(\tilde{\sigma}') \subseteq \mathrm{rel}(\tilde{\sigma})$ *there is a formula* $\theta \in \mathrm{L}_\mu[\tilde{\sigma}']$ *such that*

- $\tilde{\varphi}_{\mathrm{L}} \models \theta$, *and*
- $\theta \models \tilde{\varphi}_{\mathrm{R}}$ *for all* $\tilde{\varphi}_{\mathrm{R}} \in \mathrm{L}_\mu[\tilde{\sigma}_{\mathrm{R}}]$ *where* $\tilde{\varphi}_{\mathrm{L}} \models \tilde{\varphi}_{\mathrm{R}}$ *and* $\mathrm{rel}(\tilde{\sigma}) \cap \mathrm{rel}(\tilde{\sigma}_{\mathrm{R}}) \subseteq \mathrm{rel}(\tilde{\sigma}')$.

Furthermore, θ *can be obtained effectively from* φ_{L} *and* $\tilde{\sigma}'$.

III. Uniform Interpolation for UNFP^k

A. Overview

The goal in this section is to prove Theorem 3, effective uniform interpolation for UNFP^k, but without the elementary bounds. As a by-product of this first "inefficient" version, we will get an analogue of the Janin-Walukiewicz theorem for UNFP^k (see Theorem 24). For now, we will only consider signatures without constants or equality. In the next section we will refine the construction to get the elementary bounds and we will describe how to adapt the arguments to handle features like constants and equality.

$$\begin{tikzcd}
\textbf{Relational structures} & \textbf{Coded structures} & \textbf{Coded structures} \\
\mathrm{UNFP}^k \arrow[r, "(1)"] & \mathrm{MSO} \arrow[r, "(2)"] & \text{well-struct. } \mathrm{L}_\mu \arrow[d, "(3)"] \\
\mathrm{UNFP}^k \text{ over subsignature} & & \text{well-struct. } \mathrm{L}_\mu \text{ over subsignature encoding} \arrow[ll, "(4)"]
\end{tikzcd}$$

Fig. 1. Proof structure for effective uniform interpolation

We first give the idea behind our proof. The proof structure is presented in Figure 1.

Because of the tree-like model property for UNFP, we can restrict to structures of bounded tree-width. In fact, if k is the width of the UNFP-formula, then by Theorem 2 we can restrict to structures of tree-width $k-1$.

We now want to convert a UNFP^k sentence into a logical formula defining the trees that encode its tree-like models. This requires two steps: we must first argue that the set of codes is a regular set of trees, and we must come up with a corresponding "regular" representation of it. We call the representation in MSO the *naïve forward mapping step* (1). Secondly we have a *simplification step* (2): we use the fact that the original sentence was in UNFP^k, and hence UN^k-bisimulation invariant, to simplify the regular representation into a L_μ-formula, of a particular "well-structured" form, defined later.

Next is the L_μ*-interpolation step* (3): we apply uniform interpolation for L_μ in the coded world to get a new L_μ formula representing the interpolant we want on these tree codes. Finally, there is the *backward mapping step* (4): we translate this L_μ-formula back into UNFP^k, which yields a uniform interpolant for the original UNFP^k sentence.

It turns out that in order to actually transform from L_μ to UNFP^k it helps to have stronger properties on the tree-like models and coded structures. We capture this idea in the notion of a "shrewd unravelling" and "shrewd tree" below.

B. Schema for coding structures by trees

It is well-known that structures of tree-width $k-1$ can be interpreted by labeled trees over an alphabet that depends only on the signature of the structure and k. But in this work we will need to choose a special kind of encoding — for example, we will need that the decoding can be done within UNFP^k.

Fix some signature σ and some $k \in \mathbb{N}$. We now define a signature $\tilde{\sigma}_k$ for structures that encode tree decompositions of σ-structures of tree-width at most $k-1$. Informally, these coded tree structures will mimic the structure of a UN-bisimulation game of width k, alternating between *interface* *nodes* representing positions with at most one element, and *bag nodes* representing positions with k elements. Edges between nodes will indicate the relationship between the names of elements in each node.

Formally, we define $\tilde{\sigma}_k$ as follows.

- There is a unary relation $I \in \tilde{\sigma}_k$, which will be used to indicate interface nodes.
- There are unary relations $D_n \in \tilde{\sigma}_k$ for $n \in \{0, 1, k\}$, which will be used to indicate the number of elements represented at each node.
- For every relation $R \in \sigma$ of arity n and every sequence $\boldsymbol{i} = i_1 \dots i_n$ of n indices taken from $\{1, \dots, k\}$, there is a unary relation $R_{\boldsymbol{i}} \in \tilde{\sigma}_k$, which will be used to indicate that the tuple of elements indexed by $\boldsymbol{i}$ at that node is in R. For instance, if $R \in \sigma$ is a binary relation and $k = 2$, then we would have $R_{11}, R_{12}, R_{21}, R_{22} \in \tilde{\sigma}_k$. For a sequence $\boldsymbol{i} = i_1 \dots i_n$, we say that $R_{\boldsymbol{i}}$ uses *indices* $i_1 \dots i_n$.
- For every partial 1-1 map ρ from $\{1, \dots, k\}$ to $\{1, \dots, k\}$ with $\mathrm{dom}(\rho) = \{1\}$ or $\mathrm{rng}(\rho) = \{1\}$, there is a binary relation $E_\rho \in \tilde{\sigma}_k$. These will be used to indicate the relationship between elements in neighboring nodes.

Let $Props_k := props(\tilde{\sigma}_k)$ (respectively, $Actions_k := actions(\tilde{\sigma}_k)$) denote the collection of unary relations (respectively, binary relations) in $\tilde{\sigma}_k$. Likewise, let $Props^1_k$ denote the set of unary relations from $\tilde{\sigma}_k$ that use only one index. We will sometimes think of $\tilde{\sigma}_k$-structures as transitions systems over propositions $Props_k$ and actions $Actions_k$. This means node labels come from $\mathcal{P}(Props_k)$ and edge labels from $Actions_k$.

Let $\mathcal{T}$ be a $\tilde{\sigma}_k$-tree. We say a node v is an *empty interface node* (respectively, *non-empty interface node*) if $v \in I$ and $v \in D_0$ (respectively, $v \in I$ and $v \in D_1$). We say v is a *bag node* if $v \notin I$ and $v \in D_k$. We say $\mathcal{T}$ is *consistent* if it satisfies certain natural conditions that ensure that these $\tilde{\sigma}_k$-structures correspond to a σ-structure: for instance, if $(u, v) \in E_\rho$, then for all $R \in \sigma$ of arity r and for all $\boldsymbol{i} \in \mathrm{dom}(\rho)^r$, $u \in R_{\boldsymbol{i}}$ iff $v \in R_{\rho(\boldsymbol{i})}$. We also enforce some simple properties about the structure of the coded trees, for instance, requiring that the root is an empty interface node, and nodes at even (respectively, odd) depths are interface nodes (respectively, bag nodes).

The tree decompositions of every σ-structure of tree-width $k - 1$ can be encoded in consistent $\tilde{\sigma}_k$-trees, and every consistent $\tilde{\sigma}_k$-tree corresponds to an actual σ-structure. Given a consistent $\tilde{\sigma}_k$-tree $\mathcal{T}$ with nodes $v, w \in \mathrm{dom}(\mathcal{T})$ and indices $i, j \in \{1, \dots, k\}$, we say (v, i) is equivalent to (w, j) if the i-th element in node v corresponds to the j-th element in node w, based on the edge label mappings between nodes in the tree. Let $[v, i]$ denote the equivalence class based on this equivalence relation. Using this, we can define the *decoding* of $\mathcal{T}$ to be the σ-structure $\mathfrak{D}(\mathcal{T})$ with universe $\{[v, i] : v \in \mathrm{dom}(\mathcal{T}) \text{ and } i \in \{1, \dots, k\}\}$ and $R^{\mathfrak{D}(\mathcal{T})}([v_1, i_1], \dots, [v_r, i_r])$ iff there is some node $w \in \mathrm{dom}(\mathcal{T})$ such that $w \in R_{j_1 \dots j_r}$ and $[w, j_m] = [v_m, i_m]$ for all $m \in \{1, \dots, r\}$. For v a node in a consistent tree $\mathcal{T}$ with $v \in D_n$, we write $\mathrm{elem}(v)$ for the vector of elements $a_1 \cdots a_n$ in $\mathfrak{D}(\mathcal{T})$ induced by v in $\mathcal{T}$.

The conditions for a consistent tree are definable in FO, and are $\tilde{\sigma}_k$-bisimulation invariant.

Lemma 9. *Within the class of $\tilde{\sigma}_k$-trees, the class of consistent $\tilde{\sigma}_k$-trees is $\tilde{\sigma}_k$-bisimulation invariant and definable in* $\mathrm{FO}[\tilde{\sigma}_k]$.

There is also a connection between bisimilarity of consistent trees and UN^k-bisimilarity of the corresponding structures.

Proposition 10. *If consistent $\tilde{\sigma}_k$-trees $\mathcal{T}$ and $\mathcal{T}'$ are $\tilde{\sigma}_k$-bisimilar, then $\mathfrak{D}(\mathcal{T})$ is $\mathrm{UN}^k[\sigma]$-bisimilar to $\mathfrak{D}(\mathcal{T}')$.*

C. Coding structures shrewdly

While the decoding of a consistent tree is well-defined, there are many ways to code a given structure, depending on the particular decomposition used. Below we will define particular codings based on unravellings.

The $\mathrm{UN}^k[\sigma]$*-unravelling* of a σ-structure $\mathfrak{A}$ is defined as follows. Consider the set Π of finite sequences of the form $X_0 Y_1 X_1 \dots Y_m$ or $X_0 Y_1 X_1 \dots Y_m X_m$, where each X_i is a tuple of elements of $\mathfrak{A}$ of size at most 1, each Y_i is a tuple of elements of $\mathfrak{A}$ of size k, and Y_{i+1} contains every element in X_i and X_{i+1}. We can assume $X_0 = \emptyset$. Each $\pi \in \Pi$ represents the projection to $\mathfrak{A}$ of a play in the $\mathrm{UN}^k[\sigma]$-bisimulation game between $\mathfrak{A}$ and some other structure, starting from the empty position.

We want to define a $\tilde{\sigma}_k$-tree $\mathcal{T} = \mathcal{T}^k(\mathfrak{A})$ based on this. Sequences $\pi \in \Pi$ can be arranged in a tree structure in the obvious way based on the prefix order. We interpret the predicates in $\tilde{\sigma}_k$ as discussed earlier. For all nodes π ending in a position $Y = a_1 \dots a_n$,

- $\pi \in I^{\mathcal{T}}$ iff π ends in an interface position, i.e. π is of the form $X_0 Y_1 X_1 \dots Y_m X_m$,
- $\pi \in D^{\mathcal{T}}_j$ iff $j = n$,
- $\pi \in R^{\mathcal{T}}_{\boldsymbol{i}}$ for $\boldsymbol{i} = i_1 \dots i_r$ iff $(a_{i_1}, \dots, a_{i_r}) \in R^{\mathfrak{A}}$,
- $(\pi, \pi') \in E^{\mathcal{T}}_\rho$ iff $\pi' = \pi Z$ and either (i) ρ is empty and there are no shared elements in Y and Z, or (ii) $\rho(i) = j$ for i (respectively, j) the index of the shared element in Y (respectively, Z).

The $\mathrm{UN}^k[\sigma]$*-unravelling* of $\mathfrak{A}$ is defined to be $\mathfrak{D}(\mathcal{T}^k(\mathfrak{A}))$, and its tree decomposition is encoded by $\mathcal{T}^k(\mathfrak{A})$.

Usually, $\mathcal{T}^k(\mathfrak{A})$ would be a suitable canonical coding of an infinite structure, allowing one to perform a forward and backward mapping as described in the overview. Indeed, an unravelling similar to this, but based on guarded bisimulation, is used in [11]. However, we were unable to do the backward mapping step when using this coding based on UN^k-unravelling, because this unravelling does not provide a close enough correspondence with the power — and limitations — of UNFP^k.

We overcome this difficulty by defining a "shrewd" unravelling. The idea is that a single tuple of elements in the original structure $\mathfrak{A}$ has many copies in the unravelling of $\mathfrak{A}$, as usual. However, we take this further, by including even more copies of certain parts of the structure, with variations to these copies that UNFP^k cannot distinguish. We capture the desired property of the coded structures in the following definition, and then define the corresponding unravelling that will yield coded trees like this. For $\tau, \tau' \subseteq \mathit{Props}_k$, we write $\tau' \subseteq_1 \tau$ if $\tau' \subseteq \tau$ and $\tau' \cap \mathit{Props}_k^1 = \tau \cap \mathit{Props}_k^1$.

Definition 11 (Shrewd tree)**.** *We say a consistent $\tilde{\sigma}_k$-tree $\mathcal{T}$ is* shrewd *if it satisfies the following property. For all interface nodes v, if w is a ρ-child of v and τ is the set of unary predicates from $\tilde{\sigma}_k$ that hold at w, then for any $\tau' \subseteq_1 \tau$, there is a ρ-child w' of v such that*

- *τ' describes exactly the collection of unary predicates that hold at w', and*
- *the subtrees rooted at w and w' are isomorphic (ignoring w and w').*

We now define the *shrewd* $\mathrm{UN}^k[\sigma]$*-unravelling*. Given a σ-structure $\mathfrak{A}$ and some sequence $\boldsymbol{a} = a_1 \dots a_n$ of elements from $\mathfrak{A}$ (of size at most k), we define $\tau(\boldsymbol{a}) \subseteq \mathit{Props}_k$ such that $R_{\boldsymbol{i}} \in \tau(\boldsymbol{a})$ for $\boldsymbol{i} = i_1 \dots i_r$ iff $(a_{i_1}, \dots, a_{i_r}) \in R^{\mathfrak{A}}$. Now consider the set Π of finite sequences of the form $X_0(Y_1, \tau_1)X_1 \dots (Y_m, \tau_m)$ or $X_0(Y_1, \tau_1)X_1 \dots (Y_m, \tau_m)X_m$, where each X_i is a tuple of elements of $\mathfrak{A}$ of size at most 1, each Y_i is a tuple of elements of $\mathfrak{A}$ of size k, Y_{i+1} contains every element in X_i and X_{i+1}, and $\tau_i \subseteq_1 \tau(Y_i)$.

We can define a $\tilde{\sigma}_k$-tree $\mathcal{T} = \mathcal{S}^k(\mathfrak{A})$ based on this. Sequences $\pi \in \Pi$ can be arranged in a tree structure in the obvious way based on the prefix order. We interpret the predicates in $\tilde{\sigma}_k$ as follows. For all nodes π ending in a position of the form Y or (Y, τ) for $Y = a_1 \dots a_n$,

- $\pi \in I^{\mathcal{T}}$ iff π ends in an interface position Y,
- $\pi \in D_j^{\mathcal{T}}$ iff $j = n$,
- $\pi \in R_{\boldsymbol{i}}^{\mathcal{T}}$ iff π ends in the interface position Y and $R_{\boldsymbol{i}} \in \tau(\boldsymbol{a})$, or π ends in the bag position (Y, τ) and $R_{\boldsymbol{i}} \in \tau$,
- $(\pi, \pi') \in E_\rho^{\mathcal{T}}$ iff π' is of the form πZ or $\pi(Z, \tau)$ and either (i) ρ is empty and there are no shared elements in Y and Z, or (ii) $\rho(i) = j$ for i (respectively, j) the index of the shared element in Y (respectively, Z).

The *shrewd* UN^k*-unravelling* of $\mathfrak{A}$ is $\mathfrak{D}(\mathcal{S}^k(\mathfrak{A}))$, and its key properties are stated in the following lemma.

Lemma 12. *For all σ-structures $\mathfrak{A}$, $\mathcal{S}^k(\mathfrak{A})$ is a shrewd consistent $\tilde{\sigma}_k$-tree, and $\mathfrak{D}(\mathcal{S}^k(\mathfrak{A}))$ is $\mathrm{UN}^k[\sigma]$-bisimilar to $\mathfrak{A}$.*

We remark that none of the logics considered in this paper can enforce that a given consistent $\tilde{\sigma}_k$-tree $\mathcal{T}$ is shrewd or that $\mathcal{T}$ is the shrewd-unravelling of some structure $\mathfrak{A}$ (i.e. that $\mathcal{T} = \mathcal{S}^k(\mathfrak{A})$), but this is not problematic for our purposes.

D. Forward and backward mapping between relational structures and their encodings as trees

We are now ready to state the naïve forward mapping step mentioned in the overview, moving between coded $\tilde{\sigma}_k$-structures and σ-structures. This mapping will be from $\mathrm{GSO}[\sigma]$ to $\mathrm{MSO}[\tilde{\sigma}_k]$.

Theorem 13 ($\mathrm{GSO}[\sigma]$ to $\mathrm{MSO}[\tilde{\sigma}_k]$)**.** *Let ψ be a sentence in $\mathrm{GSO}[\sigma]$. For all k, we can construct a sentence $\psi^{\rightarrow} \in \mathrm{MSO}[\tilde{\sigma}_k]$ such that for all consistent $\tilde{\sigma}_k$ trees $\mathcal{T}$, $\mathfrak{D}(\mathcal{T}) \models \psi$ iff $\mathcal{T} \models \psi^{\rightarrow}$.*

Note that the mapping "plays well" with bisimulations.

Corollary 14. *If $\psi \in \mathrm{GSO}[\sigma]$ is a sentence that is invariant under $\mathrm{UN}^k[\sigma]$-bisimulation, then $\psi^{\rightarrow}$ is $\tilde{\sigma}_k$-bisimulation invariant on consistent $\tilde{\sigma}_k$-trees.*

Proof: Assume $\mathcal{T}$ and $\mathcal{T}'$ are $\tilde{\sigma}_k$-bisimilar consistent $\tilde{\sigma}_k$-trees with roots ϵ and ϵ', respectively. Then using Theorem 13 and Proposition 10, we have $\mathcal{T}, \epsilon \models \psi^{\rightarrow}$ iff $\mathfrak{D}(\mathcal{T}) \models \psi$ iff $\mathfrak{D}(\mathcal{T}') \models \psi$ iff $\mathcal{T}', \epsilon' \models \psi^{\rightarrow}$. ■

Once we have moved from a UNFP^k formula over structures to an MSO formula on their encodings, the next step is the invariant simplification step, which gets us from bisimulation-invariant MSO on encodings to an L_μ formula on encodings. In fact, it is helpful at this stage to simplify to what we call "well-structured" L_μ. We say a L_μ-formula is *well-structured* if there is some set $Q = \{q_1, \dots, q_n\}$ with corresponding fixpoint variables X_{q_i} and fixpoints $\lambda_i \in \{\mu, \nu\}$ such that the formula is of the form

$$\lambda_n X_{q_n} \dots \lambda_1 X_{q_1} . \begin{pmatrix} \delta_{q_1} \\ \vdots \\ \delta_{q_n} \end{pmatrix} \quad \text{with}$$

$$\delta_q := \bigvee_{\tau} \left[\left(\bigwedge_{P \in \tau} P \wedge \bigwedge_{P \in \mathit{Props}_k \setminus \tau} \neg P \right) \wedge \delta_{q,\tau} \right]$$

$$\delta_{q,\tau} := \bigvee_{S} \left(\bigwedge_{(\rho,r) \in S} \langle \rho \rangle X_r \wedge \bigwedge_{\rho \in \mathit{Actions}_k} [\rho] \bigvee_{(\rho,r) \in S} X_r \right)$$

where the outer disjunction in δ_q ranges over some collection of $\tau \in \mathcal{P}(\mathit{Props}_k)$, and the outer disjunction in $\delta_{q,\tau}$ ranges over some collection of $S \in \mathcal{P}(\mathit{Actions}_k \times Q)$. We also require that there is some i such that the fixpoint predicates $X_{q_1}, \dots, X_{q_i}$ only contain bag nodes, and $X_{q_{i+1}}, \dots, X_{q_n}$ only contain interface nodes, by requiring that the corresponding τ in δ_q include or omit the proposition I as appropriate. The simplification step is captured in the following theorem.

Theorem 15. *Let $\psi \in \mathrm{MSO}[\tilde{\sigma}_k]$ be $\tilde{\sigma}_k$-bisimulation invariant over the class of $\tilde{\sigma}_k$-trees. Then we can effectively obtain a well-structured L_μ-formula ψ' such that for all $\tilde{\sigma}_k$-trees $\mathcal{T}$, $\mathcal{T} \models \psi$ iff $\mathcal{T} \models \psi'$.*

Proof: Apply the Janin-Walukiewicz theorem (Theorem 6), and then take advantage of the equivalence between L_μ and a form of automata called *μ-automata* [20]. The structure in the formula comes from the structure of the transition function of the automaton. ■

Uniform interpolation of well-structured L_μ-formulas can then be done, with the help of Theorem 8; the well-structured form again follows by observing that the interpolation result in [8] actually uses the μ-automata mentioned earlier.

Theorem 16. *Let $\psi \in \mathrm{L}_\mu[\tilde{\sigma}]$ be a well-structured L_μ-formula. Let $\tilde{\sigma}'$ be a subsignature of $\tilde{\sigma}$. We can construct a well-structured $\theta \in \mathrm{L}_\mu[\tilde{\sigma}']$ such that θ is a uniform interpolant for ψ and $\tilde{\sigma}'$.*

Finally, the backward step moves from L_μ-formulas on encodings, back to UNFP^k on relational structures. This is the step that requires the most work.

Theorem 17 ($\mathrm{L}_\mu[\tilde{\sigma}_k]$ to $\mathrm{UNFP}^k[\sigma]$)**.** *Let ψ be a well-structured formula in $\mathrm{L}_\mu[\tilde{\sigma}_k]$. We can construct a sentence $\psi^{\leftarrow} \in \mathrm{UNFP}^k[\sigma]$ such that for all σ-structures $\mathfrak{B}$, $\mathfrak{B} \models \psi^{\leftarrow}$ iff $\mathcal{S}^k(\mathfrak{B}) \models \psi$.*

Proof sketch: Fix $\psi \in \mathrm{L}_\mu[\tilde{\sigma}_k]$. We want to produce the $\mathrm{UNFP}^k[\sigma]$ sentence $\psi^{\leftarrow}$. Unfortunately, the well-structured L_μ-formula cannot be translated directly into UNFP, since we may need to use non-unary negation in the translation of some $\bigwedge_{P \in Props_k \setminus \tau} \neg P$ (for instance, if $R_{21} \in Props_k \setminus \tau$, then $\neg R_{21}$ would become the non-unary negation $\neg R x_2 x_1$). Hence we first transform into a "UNFP-safe" version that is easier to work with; this transformation takes advantage of the fact that we are working over shrewd consistent trees (see Claim 18). Once we are in this form, we then give the backward mapping (Claim 19).

The UNFP-safe L_μ formulas restrict indices and other problematic constructs that would lead to negation in the backward mapping. Let $\mathrm{indices}(\chi)$ denote the outermost indices in the formula, up to the next occurrences of a modality. We say an $\mathrm{L}_\mu[\tilde{\sigma}_k]$ formula is UNFP-*safe for interface nodes* (respectively, UNFP-*safe for bag nodes*) if

- there are no explicit ν-fixpoints or box-modalities (they have been rewritten using negation, μ-fixpoints, and diamond modalities);
- negation is only applied to subformulas χ where $|\mathrm{indices}(\chi)| \leq 1$;
- for every subformula χ in the scope of an even (respectively, odd) number of modalities, $\mathrm{indices}(\chi)$ is contained in $\{1\}$ (respectively, $\{1, \dots, k\}$);
- every fixpoint subformula or fixpoint variable is in the scope of an even (respectively, odd) number of modalities (i.e. fixpoints reference only interface nodes).

In general, we would not be able to convert an arbitrary L_μ-formula to an equivalent UNFP-safe form. However, because we are working over shrewd consistent trees, we can convert any well-structured L_μ formula to a UNFP-safe form. For instance, we can show that we can replace some instances of $\bigwedge_{P \in Props_k \setminus \tau} \neg P$ that are under a diamond modality with just $\bigwedge_{P \in Props^1_k \setminus \tau} \neg P$.

Claim 18. *Let $\psi' \in \mathrm{L}_\mu[\tilde{\sigma}_k]$ be a well-structured formula. Then we can effectively obtain a* UNFP-*safe formula φ such that for all shrewd consistent trees $\mathcal{T}$, $\mathcal{T} \models \psi'$ iff $\mathcal{T} \models \varphi$.*

Once we have this UNFP-safe L_μ formula, we can translate into UNFP^k. The proof proceeds by induction on the structure of the formula, so we must handle free variables. For each fixpoint variable X we introduce two fixpoint variables X_1 and X_0 to handle non-empty and empty interface nodes, respectively; we define $X^{\leftarrow} := (X_0, X_1)$. A set J of nodes in $\mathcal{S}^k(\mathfrak{B})$ is a UNFP-*safe valuation for a free variable* X if it (i) contains only interface nodes, and (ii) if it contains an empty interface node then it contains every empty interface node. We write $J^{\leftarrow}$ for its representation in $\mathfrak{B}$, i.e. $J^{\leftarrow} := (J_0, J_1)$ where J_1 is defined to be $\{\mathrm{elem}(v) : v \in J \text{ and } |\mathrm{elem}(v)| = 1\}$, and J_0 is $\top$ (respectively, $\bot$) if J contains all empty interface node (respectively, contains no empty interface nodes). The strengthened result for formulas with free variables is captured in the following claim.

Claim 19. *Let $\varphi \in \mathrm{L}_\mu[\tilde{\sigma}_k]$ be* UNFP-*safe for interface nodes (respectively, bag nodes) with free second-order variables $\boldsymbol{X}$. We can construct* $\mathrm{UNFP}[\sigma]$-*formulas $\varphi_0^{\leftarrow}(\boldsymbol{X}^{\leftarrow})$, $\varphi_1^{\leftarrow}(x_1, \boldsymbol{X}^{\leftarrow})$, and $\varphi_k^{\leftarrow}(x_1, \dots, x_k, \boldsymbol{X}^{\leftarrow})$ of width k such that for all σ-structures $\mathfrak{B}$, for all* UNFP-*safe valuations $\boldsymbol{J}$ of $\boldsymbol{X}$, and for all interface nodes (respectively, bag nodes) v in $\mathcal{S}^k(\mathfrak{B})$ with $|\mathrm{elem}(v)| = m$,*

$$\mathfrak{B}, \mathrm{elem}(v), \boldsymbol{J}^{\leftarrow} \models \varphi_m^{\leftarrow} \text{ iff } \mathcal{S}^k(\mathfrak{B}), v, \boldsymbol{J} \models \varphi.$$

Moreover, if $\mathrm{indices}(\varphi) = \{i_1, \dots, i_n\}$, then $\mathrm{free}(\varphi_m^{\leftarrow}) = \{x_{i_1}, \dots, x_{i_n}\}$, and any strict subformula in $\varphi_m^{\leftarrow}$ that begins with an existential quantifier and is not directly below another existential quantifier has at most one free variable.

We construct the formulas as follows.

- Assume $\varphi = I$. Then $\varphi_0^{\leftarrow} := \top$, $\varphi_1^{\leftarrow} := \top$, and $\varphi_k^{\leftarrow} := \bot$.
- Assume $\varphi = D_j$. Then $\varphi_m^{\leftarrow} := \begin{cases} \top & \text{if } m = j, \\ \bot & \text{otherwise.} \end{cases}$
- Assume $\varphi = R_{i_1 \dots i_n}$. Then $\varphi_0^{\leftarrow} := \bot$, $\varphi_1^{\leftarrow} := R x_{i_1} \dots x_{i_n}$, and $\varphi_k^{\leftarrow} := R x_{i_1} \dots x_{i_n}$.
- Assume $\varphi = X$. Then $\varphi_0^{\leftarrow} := X_0$, $\varphi_1^{\leftarrow} := X_1 x_1$, and $\varphi_k^{\leftarrow} := \bot$.
- The translation commutes with $\vee$, $\wedge$, and $\neg$. For the negation case, the definition of UNFP-safety and the inductive hypothesis ensures that the resulting formulas are in UNFP.
- Assume $\varphi = \langle \rho \rangle \chi$. Then

$$\varphi_0^{\leftarrow} := \exists y_1 \dots y_k . \left(\chi_k^{\leftarrow}(y_1, \dots, y_k)\right),$$
$$\varphi_1^{\leftarrow} := \exists y_1 \dots y_k . \left(x_1 = y_{\rho(1)} \wedge \chi_k^{\leftarrow}(y_1, \dots, y_k)\right),$$
$$\varphi_k^{\leftarrow} := \begin{cases} \chi_1^{\leftarrow}(x_i) & \text{if } \mathrm{dom}(\rho) = \{i\} \\ \chi_0^{\leftarrow} & \text{if } \mathrm{dom}(\rho) = \emptyset \end{cases}.$$

- Assume $\varphi = \mu Y . \chi(\boldsymbol{X}, Y)$. Then $\varphi_0^{\leftarrow}$ and $\varphi_1^{\leftarrow}$ are the first and second components of the vectorial fixpoint

$$\varphi^{\leftarrow} := \mathbf{lfp} \begin{pmatrix} Y_0 \\ Y_1, y_1 \end{pmatrix} . \begin{pmatrix} \chi_0^{\leftarrow}(\boldsymbol{X}^{\leftarrow}, Y_0, Y_1) \\ \chi_1^{\leftarrow}(y_1, \boldsymbol{X}^{\leftarrow}, Y_0, Y_1) \end{pmatrix}$$

and $\varphi_k^{\leftarrow} := \bot$.

It can be checked that the constructed formulas have width k.

This concludes the proof sketch of Theorem 17, the backward mapping going from L_μ to UNFP. ■

E. Putting it all together

We are now ready to give an initial algorithm for effective uniform interpolation for UNFP^k, without elementary bounds.

Fix $\varphi_{\mathrm{L}} \in \mathrm{UNFP}^k[\sigma]$, and fix some $\sigma' \subseteq \sigma$. Let $\tilde{\sigma}_k$ and $\tilde{\sigma}'_k$ be the signatures of tree representations for σ-structures and σ'-structures, respectively.

1) *Naïve forward mapping step*: Convert φ_{L} to GSO using Proposition 5, and then obtain $\varphi_{\mathrm{L}}^{\rightarrow} \in \mathrm{MSO}[\tilde{\sigma}_k]$ using Theorem 13. This is $\tilde{\sigma}_k$-bisimulation invariant over consistent $\tilde{\sigma}_k$-trees by Corollary 14.

2) *Simplification step:* Let $\gamma_{\sigma,k}$ be an FO$[\tilde{\sigma}_k]$-sentence expressing the consistency conditions for the trees over $\tilde{\sigma}_k$, using Lemma 9. Then $\gamma_{\sigma,k} \wedge \varphi_{\mathrm{L}}{}^{\rightarrow}$ is $\tilde{\sigma}_k$-bisimulation invariant over all $\tilde{\sigma}_k$-trees, so we can obtain well-structured $\tilde{\varphi}_{\mathrm{L}} \in \mathrm{L}_\mu[\tilde{\sigma}_k]$ by applying Theorem 15 to $\gamma_{\sigma,k} \wedge \varphi_{\mathrm{L}}{}^{\rightarrow}$.
3) L_μ *interpolation step*: Obtain well-structured $\theta \in \mathrm{L}_\mu[\tilde{\sigma}'_k]$ by applying Theorem 8 to $\tilde{\varphi}_{\mathrm{L}}$ and $\tilde{\sigma}'_k$.
4) *Backward mapping step*: Obtain $\theta^{\leftarrow} \in \mathrm{UNFP}^k[\sigma']$ using Theorem 17. Set $\chi := \theta^{\leftarrow}$.

First we prove that $\varphi_{\mathrm{L}} \models \chi$. Let $\mathfrak{B}$ be a σ-structure and assume $\mathfrak{B} \models \varphi_{\mathrm{L}}$. Then $\mathfrak{D}(\mathcal{S}^k(\mathfrak{B})) \models \varphi_{\mathrm{L}}$ (since φ_{L} is $\mathrm{UN}^k[\sigma]$-bisimulation invariant), so $\mathcal{S}^k(\mathfrak{B}) \models \varphi_{\mathrm{L}}{}^{\rightarrow}$ by Theorem 13. Since $\mathcal{S}^k(\mathfrak{B})$ is a consistent $\tilde{\sigma}_k$-tree, $\mathcal{S}^k(\mathfrak{B}) \models \gamma_{\sigma,k} \wedge \varphi_{\mathrm{L}}{}^{\rightarrow}$, so $\mathcal{S}^k(\mathfrak{B}) \models \tilde{\varphi}_{\mathrm{L}}$. By Theorem 8, this means that $\mathcal{S}^k(\mathfrak{B}) \models \theta$ (now viewed as a $\tilde{\sigma}'_k$ structure). Finally, by Theorem 17, this means that $\mathfrak{B} \models \theta^{\leftarrow}$.

Next, assume that $\varphi_{\mathrm{L}} \models \varphi_{\mathrm{R}}$ for some $\varphi_{\mathrm{R}} \in \mathrm{UNFP}^k[\sigma_{\mathrm{R}}]$ where $\sigma \cap \sigma_{\mathrm{R}} \subseteq \sigma'$. Let $\sigma'' = \sigma \cup \sigma_{\mathrm{R}}$. Let $\tilde{\varphi}_{\mathrm{L}} \in \mathrm{L}_\mu[\tilde{\sigma}_k]$ as above. Let $\tilde{\varphi}_{\mathrm{R}} \in \mathrm{L}_\mu[\tilde{\sigma}_{\mathrm{R},k}]$ be the formula obtained by applying Theorem 6 to $\gamma_{\sigma_{\mathrm{R}},k} \rightarrow \varphi_{\mathrm{R}}{}^{\rightarrow}$, where $\varphi_{\mathrm{R}}{}^{\rightarrow}$ is obtained using Theorem 13. We aim to prove that $\chi \models \varphi_{\mathrm{R}}$.

We first prove that $\tilde{\varphi}_{\mathrm{L}} \models \tilde{\varphi}_{\mathrm{R}}$ over all $\tilde{\sigma}''_k$-trees. Assume $\mathcal{T} \models \tilde{\varphi}_{\mathrm{L}}$, where $\mathcal{T}$ is a $\tilde{\sigma}''_k$-tree. Then $\mathcal{T} \models \gamma_{\sigma,k} \wedge \varphi_{\mathrm{L}}{}^{\rightarrow}$ so we know that $\mathcal{T}$ is a consistent $\tilde{\sigma}_k$-tree. If $\mathcal{T}$ is not $\tilde{\sigma}_{\mathrm{R},k}$-consistent, then $\mathcal{T} \models \gamma_{\sigma_{\mathrm{R}},k} \rightarrow \varphi_{\mathrm{R}}{}^{\rightarrow}$, so $\mathcal{T} \models \tilde{\varphi}_{\mathrm{R}}$ and we are done. Otherwise, $\mathcal{T}$ is both $\tilde{\sigma}_k$-consistent and $\tilde{\sigma}_{\mathrm{R},k}$-consistent, which is enough to conclude that it is a consistent $\tilde{\sigma}''_k$-tree. Hence, by Theorem 13, $\mathfrak{D}(\mathcal{T}) \models \varphi_{\mathrm{L}}$. By our initial assumption that $\varphi_{\mathrm{L}} \models \varphi_{\mathrm{R}}$, this means $\mathfrak{D}(\mathcal{T}) \models \varphi_{\mathrm{R}}$. By Theorem 13, this means that $\mathcal{T} \models \varphi_{\mathrm{R}}{}^{\rightarrow}$, so by weakening, $\mathcal{T} \models \gamma_{\sigma_{\mathrm{R}},k} \rightarrow \varphi_{\mathrm{R}}{}^{\rightarrow}$. This means that $\mathcal{T} \models \tilde{\varphi}_{\mathrm{R}}$.

Now we claim that $\tilde{\varphi}_{\mathrm{L}} \models \tilde{\varphi}_{\mathrm{R}}$ over all $\tilde{\sigma}''_k$-structures. Assume not. Then there is some $\tilde{\sigma}''_k$-structure $\mathfrak{A}$ such that $\mathfrak{A} \models \tilde{\varphi}_{\mathrm{L}} \wedge \neg\tilde{\varphi}_{\mathrm{R}}$. But by the tree-model property of L_μ [18], this means there is some $\tilde{\sigma}''_k$-tree $\mathcal{T}$ such that $\mathcal{T} \models \tilde{\varphi}_{\mathrm{L}} \wedge \neg\tilde{\varphi}_{\mathrm{R}}$, which contradicts the previous paragraph. This means that $\tilde{\varphi}_{\mathrm{L}} \models \tilde{\varphi}_{\mathrm{R}}$ over all $\tilde{\sigma}''_k$-structures.

Since $\tilde{\varphi}_{\mathrm{L}} \models \tilde{\varphi}_{\mathrm{R}}$ we can apply Theorem 8 to see that $\theta \models \tilde{\varphi}_{\mathrm{R}}$ (over all $\tilde{\sigma}_{\mathrm{R},k}$-structures).

Now we are ready to show that $\chi \models \varphi_{\mathrm{R}}$. If $\mathfrak{B} \models \theta^{\leftarrow}$ for a σ_{R}-structure $\mathfrak{B}$, then $\mathcal{S}^k(\mathfrak{B}) \models \theta$ by Theorem 17. By the previous paragraph, this implies that $\mathcal{S}^k(\mathfrak{B}) \models \tilde{\varphi}_{\mathrm{R}}$ and hence $\mathcal{S}^k(\mathfrak{B}) \models \gamma_{\sigma_{\mathrm{R}},k} \rightarrow \varphi_{\mathrm{R}}{}^{\rightarrow}$. But this means that $\mathfrak{D}(\mathcal{S}^k(\mathfrak{B})) \models \varphi_{\mathrm{R}}$ by Theorem 17. Since $\mathfrak{B}$ and $\mathfrak{D}(\mathcal{S}^k(\mathfrak{B}))$ are $\mathrm{UN}^k[\sigma_{\mathrm{R}}]$-bisimilar, this means that $\mathfrak{B} \models \varphi_{\mathrm{R}}$ as desired.

This completes the proof of uniform interpolation for UNFP^k, without the elementary bounds. We will come back to this technique using the naïve forward mapping in order to prove Theorem 24.

IV. Refining and Extending the Construction

A. Obtaining elementary bounds

The prior approach using the naïve forward mapping yields a non-elementary upper bound on the size of the uniform interpolant and the time complexity of constructing it. This is due to the fact that we pass through MSO, and converting from MSO to L_μ can result in a non-elementary blow-up [21].

$$\begin{tikzcd}
\textbf{Relational structures} & \textbf{Coded structures} & \textbf{Coded structures} \\
\mathrm{UNFP}^k \arrow[r, "(1)"] & \text{2-way } \mu\text{-aut.} \arrow[r, "(2)"] & \text{well-struct. } \mathrm{L}_\mu \arrow[d, "(3)"] \\
\mathrm{UNFP}^k & & \text{well-struct. } \mathrm{L}_\mu \arrow[ll, "(4)"'] \\
\text{over subsignature} & & \text{over subsignature encoding}
\end{tikzcd}$$

Fig. 2. Proof structure for elementary uniform interpolation

We improve the complexity of this uniform interpolation algorithm in this section by avoiding GSO and MSO, and using automata on trees instead (see Figure 2).

The automata that we use are designed to work on trees with arbitrary branching, so they cannot refer to specific children of a node. This is different than traditional tree automata on binary trees that can refer to the left child and right child. To start, we also allow 2-way movement.

Formally, a *2-way alternating μ-automaton* $\mathcal{A}$ is a tuple $\langle \Sigma_p, \Sigma_a, Q_E, Q_A, q_0, \delta, \Omega \rangle$ where Σ_p and Σ_a are finite alphabets of propositions and actions, $Q := Q_E \cup Q_A$ is a finite set of states partitioned into states Q_E controlled by Eve and states Q_A controlled by Adam, and $q_0 \in Q$ is the initial state. The transition function has the form $\delta : Q \times \Sigma_p \rightarrow \mathcal{P}(\mathrm{Dir} \times \Sigma_a \times Q)$ where $\mathrm{Dir} = \{\uparrow, 0, \downarrow\}$ is the set of possible directions (up $\uparrow$, stay 0, down $\downarrow$). The acceptance condition is a parity condition specified by $\Omega : Q \rightarrow \mathrm{Pri}$, which maps each state to a priority in a finite set of priorities Pri.

We view $\mathcal{A}$ running on a tree $\mathcal{T}$ starting at node $v_0 \in \mathrm{dom}(\mathcal{T})$ as a game $\mathcal{G}(\mathcal{A}, \mathcal{T}, v_0)$. The arena is $Q \times \mathrm{dom}(\mathcal{T})$, and the initial position is (q_0, v_0). From a position (q, v) with $q \in Q_E$ (respectively, $q \in Q_A$), Eve (respectively Adam) selects $(d, a, r) \in \delta(q, \mathcal{T}(v))$. If $d = 0$, then the game continues from position (r, v). Otherwise, Eve (respectively, Adam) selects an a-neighbor w of v in direction d and the game continues from position (r, w).

A *play* in $\mathcal{G}(\mathcal{A}, \mathcal{T}, v_0)$ is a sequence $(q_0, v_0), (q_1, v_1), (q_2, v_2), \ldots$ of moves in the game. Such a *play is winning* for Eve if the *parity condition* is satisfied: the maximum priority that occurs infinitely often in $\Omega(q_0), \Omega(q_1), \ldots$ is even.

A *strategy* for one of the players is a function that returns the next choice for that player given the history of the play. Choosing a strategy for both players fixes a play in $\mathcal{G}(\mathcal{A}, \mathcal{T}, v_0)$. A play π is *compatible* with a strategy ζ if there is a strategy ζ' for the other player such that ζ and ζ' yield π. A *strategy is winning* for Eve if every play compatible with it is winning.

We write $L(\mathcal{A})$ for the set of trees $\mathcal{T}$ such that Eve has a winning strategy in $\mathcal{G}(\mathcal{A}, \mathcal{T}, \epsilon)$, where ϵ is the root of $\mathcal{T}$.

The idea is to redo the forward mapping, but now go directly from UNFP^k to a 2-way alternating μ-automaton. For brevity in the following theorem (and the rest of this section), we give only bounds on the output size, not the running time of the algorithm. However the proofs will show that the worst-case running time is bounded by a polynomial in the output size.

Theorem 20 (UNFP$^k[\sigma]$ to 2-way alternating μ-automata)**.** *Let ψ be a sentence in* UNFP$^k[\sigma]$*. We can construct a 2-way alternating μ-automaton $\mathcal{A}_\psi$ such that for all consistent $\tilde{\sigma}_k$ trees $\mathcal{T}$, $\mathfrak{D}(\mathcal{T}) \models \psi$ iff $\mathcal{T} \models L(\mathcal{A}_\psi)$ and the size of $\mathcal{A}_\psi$ is doubly exponential in $|\psi|$, but the number of states and priorities of $\mathcal{A}_\psi$ is at most singly exponential in $|\psi|$.*

Proof sketch: We need to test whether ψ holds in the decoding of the input tree $\mathcal{T}$. We construct the automaton by induction on the structure of ψ. The idea behind the construction is to allow Eve to guess an annotation of $\mathcal{T}$ with information about which unary subformulas of ψ hold, and then run an automaton that checks ψ with the help of these annotations. In order to prevent Eve from cheating with her guesses about the subformulas, Adam is also allowed to launch automata that check Eve's claims about the subformulas.

Likewise, testing $\mathfrak{D}(\mathcal{T}) \models [\mathbf{lfp}\, Y, y.\chi(y, Y)](a)$ can be viewed as a game between Adam and Eve which starts with $y = a$ and proceeds as follows:

- Eve chooses some valuation for Y such that $\mathfrak{D}(\mathcal{T}) \models \chi(y, Y)$ (she loses if this is not possible), then
- Adam chooses some new $y \in Y$ (he loses if this is not possible), and then the game proceeds to the next turn.

If the game never terminates then Adam is declared the winner. We can implement this game as an automaton running on $\mathcal{T}$, where Eve guesses an annotation of $\mathcal{T}$ with the valuation of Y in the current round, and then simulates the inductively-defined automaton checking $\chi(y, Y)$. Adam can challenge any b in the set Y chosen by Eve by launching another copy of the automaton checking χ starting from the node carrying b. Correctness is enforced by using the parity condition: an odd priority is used if Adam challenges a **lfp** fixpoint. ∎

The new simplification step then converts this 2-way alternating μ-automaton to a well-structured L$_\mu$-formula.

Theorem 21 (2-way μ-automata to well-structured L$_\mu[\tilde{\sigma}_k]$)**.** *Let $\mathcal{A}$ be a 2-way alternating μ-automaton on $\tilde{\sigma}_k$ trees. We can construct a well-structured* L$_\mu[\tilde{\sigma}_k]$*-formula ψ such that for all consistent $\tilde{\sigma}_k$ trees $\mathcal{T}$, $\mathfrak{D}(\mathcal{T}) \models \psi$ iff $\mathcal{T} \in L(\mathcal{A})$. The size of ψ is at most $|\mathcal{A}|^{f(m)}$ where m is the number of states and priorities of $\mathcal{A}$ and f is a polynomial function independent of $\mathcal{A}$.*

This is accomplished by converting to a μ-automaton [20] — which can be seen as the 1-way nondeterministic counterpart to the 2-way alternating automata described earlier — using a variation of a construction in [22].

Uniform interpolation of well-structured L$_\mu$-formulas can be done without a blow-up in the size of the formula.

Theorem 22. *Let $\psi \in$ L$_\mu[\tilde{\sigma}]$ be a well-structured* L$_\mu$*-formula. Let $\tilde{\sigma}'$ be a subsignature of $\tilde{\sigma}$. We can construct a well-structured $\theta \in$ L$_\mu[\tilde{\sigma}']$ such that θ is a uniform interpolant for ψ and $\tilde{\sigma}'$ and is of size at most $|\psi|$.*

As with the previous step, this is also accomplished using μ-automata. Finding the uniform interpolant corresponds to taking a projection of the automaton language based on the (encoding) of the desired subsignature. Indeed, this is how uniform interpolation for μ-calculus is proven (see [8, Theorem 3.3 and Corollary 3.4]). This does not increase the size of the automaton, or the corresponding well-structured L$_\mu$-formula.

Finally, the backward step is now from well-structured L$_\mu$ to UNFP^k.

Theorem 23 (well-structured L$_\mu[\tilde{\sigma}_k]$ to UNFP$^k[\sigma]$)**.** *Let $\psi \in$ UNFP$^k[\tilde{\sigma}_k]$ be well-structured. We can construct a DAG-representation of a sentence $\psi^{\leftarrow} \in$ UNFP$^k[\sigma]$ such that for all σ-structures $\mathfrak{B}$, $\mathfrak{B} \models \psi^{\leftarrow}$ iff $\mathcal{S}^k(\mathfrak{B}) \models \psi$, and the size of the DAG-representation of $\psi^{\leftarrow}$ is polynomial in the size of ψ.*

The proof of the backward mapping step is actually the same as before, noting that the conversion starting from a well-structured L$_\mu$-formula can be done in polynomial time.

We now summarize this improved uniform interpolation algorithm (see Figure 2). Let $\varphi \in$ UNFP$^k[\sigma]$, and let $\sigma' \subseteq \sigma$ be the target subsignature for the uniform interpolant.

1) *Forward mapping step*: Apply Theorem 20 to obtain a 2-way alternating μ-automaton $\mathcal{A}$ for φ over $\tilde{\sigma}_k$-trees, of size doubly exponential in $|\varphi|$, but with the number of states and priorities only singly exponential in $|\varphi|$.
2) *Simplification step*: Apply Theorem 21 to obtain a well-structured L$_\mu[\tilde{\sigma}_k]$-formula ψ equivalent to $\mathcal{A}$. Note that ψ is of size doubly exponential in $|\varphi|$, since the exponential blow-up in this step is only relative to the number of states and priorities in $\mathcal{A}$.
3) *Interpolation step*: Apply Theorem 22 to ψ and $\tilde{\sigma}'_k$ to obtain a well-structured L$_\mu[\tilde{\sigma}'_k]$-formula θ. This is at most the size of ψ, so it is still doubly exponential in $|\varphi|$.
4) *Backward mapping step*: Obtain a DAG-representation of the UNFP$^k[\sigma']$ formula $\theta^{\leftarrow}$ equivalent to θ using Theorem 23. The DAG-representation is at most doubly exponential in $|\varphi|$. Set $\chi := \theta^{\leftarrow}$.

Overall, this means that the uniform interpolant χ has a DAG-representation that is at most doubly exponential in the size of φ, as stated in Theorem 3. As mentioned earlier, because the time complexity is bounded by the size of the output at each stage, this algorithm also yields the 2EXPTIME bound on uniform interpolation for UNFP^k.

B. Extension for unary formulas, constants, and equality

The uniform interpolation results can be easily extended to handle formulas with at most one free variable, instead of sentences. This requires only a slight change to the definition of consistent tree to allow the root to be a non-empty interface node, representing the free variable.

In order to handle signatures σ with constants, we must modify the encodings of the tree decompositions so the constants from σ are represented at every node in the tree encodings (so in particular, interface nodes contain every constant, plus at most one additional element). This comes from the fact that in order to have a connection between the UN$^k[\sigma]$-bisimulation game and satisfaction of UNFP$^k[\sigma]$

formulas with constants (as required for Proposition 7), we need information about constants at every position in the game.

Note that these constants do not adversely impact the forward and backward mapping. In particular, the constants do not lead to any difficulties in the backward mapping from L_μ to UNFP because constants can appear freely in UNFP-formulas. Indeed, as long as some formula ψ has at most one free variable, we can negate ψ regardless of the number of constants it mentions.

It is important to note that in our uniform interpolation result, the uniform interpolant may make use of any constants from the original signature. That is, although the uniform interpolant can restrict the relations that are used, it cannot restrict the constants that are used. This is unavoidable, at least for interpolation over formulas. We explain this more in the full version of this paper.

For equality, we must first convert formulas with equality to an equality normal form with very restricted uses of equality. This allows us to use the algorithms presented earlier, treating equality like any other relation. We provide more details on this equality extension in the full version of this paper.

C. *Characterization of* UNFP^k *as* UN^k*-bisimulation invariant fragment of* GSO

As a by-product of our translations, we can conclude that UNFP^k is the UN^k-bisimulation invariant fragment of GSO. This is in analogy to the fact that L_μ is the bisimulation invariant fragment of MSO (see Theorem 6), and that GFP is the "guarded bisimulation" invariant fragment of GSO [11].

Theorem 24 (UNFP^k $\equiv$ UN^k-bisimulation invariant GSO)**.** *Every sentence ψ in* GSO$[\sigma]$ *that is invariant under* UN$^k[\sigma]$*-bisimulation is effectively equivalent to a sentence φ in* UNFP$^k[\sigma]$.

Proof: If we start from a UNFP^k-sentence, then we can translate into UNFP^k-bismilar GSO (even MSO), using Propositions 5 and 7. For the other direction, fix some UN$^k[\sigma]$-bisimulation invariant sentence ψ in GSO$[\sigma]$.

1) *Naïve forward mapping step*: Obtain $\psi^{\rightarrow}$ in MSO$[\tilde{\sigma}_k]$ from ψ using Theorem 13. By Corollary 14, $\psi^{\rightarrow}$ is $\tilde{\sigma}_k$-bisimulation invariant on consistent $\tilde{\sigma}_k$-trees.
2) *Simplification step*: Let γ be an FO$[\tilde{\sigma}_k]$ sentence expressing the conditions for consistent trees, obtained using Lemma 9. Then $(\gamma \rightarrow \psi^{\rightarrow})$ in MSO$[\tilde{\sigma}_k]$ is $\tilde{\sigma}_k$-bisimulation invariant within the class of all $\tilde{\sigma}_k$-trees, so we can obtain φ in $\mathrm{L}_\mu[\tilde{\sigma}_k]$ from $(\gamma \rightarrow \psi^{\rightarrow})$ using Theorem 6.
3) *Backward mapping step*: Obtain $\varphi^{\leftarrow}$ in UNFP$^k[\sigma]$ from φ using Theorem 17.

We must show that $\varphi^{\leftarrow}$ is the UNFP$^k[\sigma]$-sentence equivalent to ψ. Consider some σ-structure $\mathfrak{B}$. Assume that $\mathfrak{B} \models \psi$. Then $\mathfrak{D}(\mathcal{S}^k(\mathfrak{B})) \models \psi$ by Lemma 12 and UN$^k[\sigma]$-bisimulation invariance of ψ. Using Theorem 13, this means $\mathcal{S}^k(\mathfrak{B}) \models \psi^{\rightarrow}$. Moreover, $\mathcal{S}^k(\mathfrak{B}) \models \gamma \rightarrow \psi^{\rightarrow}$ since $\mathcal{S}^k(\mathfrak{B})$ is a consistent $\tilde{\sigma}_k$-tree. so $\mathcal{S}^k(\mathfrak{B}) \models \varphi$. Finally, by Theorem 17, $\mathfrak{B} \models \varphi^{\leftarrow}$ as desired. Similar reasoning shows that $\mathfrak{B} \models \varphi^{\leftarrow}$ implies $\mathfrak{B} \models \psi$. ■

V. Failure of interpolation

In this section we will see that some natural extensions and variants of our main interpolation theorems fail.

A. *Failure of uniform interpolation*

Although we have shown that UNFP has Craig interpolation, it fails to have uniform interpolation.

Proposition 25. *Uniform interpolation fails for* UNFP. *In particular, there is a* UNF *antecedent with no uniform interpolant in* LFP*, even when the consequents are restricted to sentences in* UNF. *The variant of uniform interpolation where entailment is considered only over finite structures also fails for* UNFP.

Proof: There is a UNF sentence φ that expresses that unary relations R,G,B form a 3-coloring of a graph with edge relation E. Consider a uniform interpolant θ (in any logic) for the UNF sentence φ with respect to its UNF-consequences in the signature $\{E\}$. We claim that there cannot be an LFP formula equivalent to θ.

For all finite graphs G that are not 3-colorable, let ψ_G be the UNF sentence corresponding to the canonical conjunctive query of G over relation E — that is, if G consists of edges E mentioning vertices $v_1 \dots v_n$, ψ_G is $\exists v_1 \dots v_n \bigwedge_{e \in G, e=(v_i,v_j)} E(v_i,v_j)$. Then φ must entail $\neg\psi_G$, since the 3-coloring of a graph G' satisfying ψ_G is easily seen to induce a 3-coloring on G.

Now consider a finite graph G. If G is 3-colorable, then $G \models \varphi$, and hence $G \models \theta$. On the other hand, if G is not 3-colorable, then $G \models \psi_G$, so $G \models \neg\theta$ because φ entails $\neg\psi_G$ and thus, by the assumption on θ, θ entails $\neg\psi_G$. Therefore, θ holds in G iff G is 3-colorable.

Dawar [23] showed that 3-colorability is not expressible in the infinitary logic $\mathcal{L}^{\omega}_{\infty\omega}$ over finite structures. Since LFP can be translated into $\mathcal{L}^{\omega}_{\infty\omega}$ over finite structures, this implies that θ cannot be in LFP.

The above argument only makes use of the properties of θ over finite structures, and thus demonstrates the failure of the variant of uniform interpolation in the finite. ■

Recall that we have trivial uniform interpolants in existential second-order logic, i.e. in NP. The previous arguments shows that interpolants for UNFP express NP-hard problems, and thus cannot be in any PTime language if PTime is not equal to NP. We remark that one could still hope to find uniform interpolants for UNFP by allowing the interpolants to live in a larger fragment that is still "tame", but we leave this as an open question.

Uniform interpolation also fails for GSO.

Proposition 26. *Uniform interpolation fails for* GSO. *In particular, there is a* GF *antecedent with no uniform interpolant in* GSO*, even when the consequents are restricted to sentences of* GF *(or* UNF*) of width 2.*

Proof sketch: Consider $\sigma = \{P, Q, R_1, R_2, S\}$, and let $\varphi \in \mathrm{GSO}[\sigma]$ be

$$\begin{aligned}&\forall z.[Qz \rightarrow \exists xy.(Sxy \wedge R_1zx \wedge R_2zy)] \wedge \\ &\forall xy.[Sxy \rightarrow \exists x'y'.(Sx'y' \wedge R_1xx' \wedge R_2yy' \wedge \\ &\qquad\qquad ((Px' \wedge Py') \vee (\neg Px' \wedge \neg Py')))]\end{aligned}$$

which expresses that there is an infinite "ladder" starting at every Q-node (where S connects pairs of elements on the same rung, and R_i connects corresponding elements on different rungs) and the pair of elements on each rung agree on P. By adding an additional dummy guard G (of arity 4), we can write a GF sentence that implies the same property.

Then for each n, we can define over $\sigma' = \{P, Q, R_1, R_2\}$ a formula $\psi_n \in \mathrm{GSO}[\sigma']$,

$$[\exists x.(Qx \wedge \forall x_1 \dots x_n((\bigwedge_i R_1 x_i x_{i+1} \wedge x_1 = x) \rightarrow Px_n))] \rightarrow$$
$$[\exists y(Qy \wedge \exists y_1 \dots y_n(\bigwedge_i R_2 y_i y_{i+1} \wedge y_1 = y \wedge Py_n))]$$

which expresses that if there is some Q-position x such that every R_1-path of length n from x ends in a position satisfying P, then there is an R_2-path of length n from some Q-position y that ends in a position satisfying P. Note that for all n, $\varphi \models \psi_n$. Moreover, ψ_n can also be expressed in either GF or UNF of width 2.

Assume for the sake of contradiction that there is some uniform interpolant θ in $\mathrm{GSO}[\sigma']$. Since GSO coincides with MSO over trees ([24], as cited in [11]), this means we can construct a nondeterministic parity tree automaton $\mathcal{A}$ that recognizes precisely the language of tree structures satisfying θ. A pumping argument can then be used to construct a tree structure accepted by this automaton, and hence satisfying θ, that fails to satisfy some ψ_n. This contradicts $\theta \models \psi_n$. ■

It has been known for some time that GF fails to have even ordinary Craig interpolation [25], and hence fails to have uniform interpolation. The previous proposition shows that we cannot get uniform interpolants for GF even when we allow the uniform interpolants to come from GSO.

B. Failure of Craig interpolation for GNFP

It is natural to try to extend our results to the logic GNFP. Unfortunately, Craig interpolation fails for GNFP.

Proposition 27. *Craig interpolation fails for* GNFP. *In particular, there is an entailment of* GFP *sentences with no* GNFP *interpolant, even over finite structures.*

Proof sketch: Let φ' be

$$[\mathbf{lfp}\, X, xy. Gxyy \wedge (Rxy \vee \exists y'.(Gxy'y \wedge Rxy' \wedge Xy'y)](xx)$$

and let φ be $\forall x.(Qx \rightarrow \varphi')$. This sentence $\varphi \in \mathrm{GFP}[\sigma]$ over the signature $\sigma = \{G, Q, R\}$ implies that there is an R-loop from every Q-labelled element. Define the $\mathrm{GFP}[\sigma']$ sentence ψ over signature $\sigma' = \{P, Q, R\}$ to be

$$\forall x.((Qx \wedge Px) \rightarrow [\mathbf{lfp}\, X, x. \exists y.(Rxy \wedge (Py \vee Xy))](x))$$

which expresses that for all Q and P labelled elements x, there is an R-path from x leading to some node y with Py.

Now $\varphi \models \psi$, but it can be shown that any GNFP sentence θ over the common signature $\{Q, R\}$ cannot distinguish between a structure consisting of a single loop satisfying φ and ψ, and a structure built from "lassos" of sufficient size (depending only on the width of θ) where φ and ψ fail to hold. Hence, there can be no GNFP interpolant. ■

VI. Conclusion

In this work we have extended the "bootstrapping from modal logic" technique of [11] to show that the logic UNFP has interpolation. We also have shown that the richer logics, such as guarded negation fixpoint, do not have interpolation. It is still possible that there is a decidable logic containing GNFP which has interpolation. We also leave open the question of whether interpolation holds over finite structures for UNFP.

Acknowledgment

Benedikt and Vanden Boom are supported by EPSRC grants EP/H017690/1 and EP/L012138/1. ten Cate is supported by NSF grant IIS-1217869.

References

[1] B. ten Cate, E. Franconi, and I. Seylan, "Beth definability in expressive description logics," in *IJCAI*, 2011.
[2] A. Nash, L. Segoufin, and V. Vianu, "Views and queries: Determinacy and rewriting," *ACM TODS*, vol. 35, no. 3, 2010.
[3] M. Marx, "Queries determined by views: pack your views," in *PODS*, 2007.
[4] K. L. McMillan, "Applications of Craig Interpolation to model checking," in *CSL*, 2004.
[5] W. Craig, "Linear reasoning. A new form of the Herbrand-Gentzen theorem," *The Journal of Symbolic Logic*, vol. 22, no. 03, pp. 250–268, 1957.
[6] M. Benedikt, B. ten Cate, and M. Vanden Boom, "Effective interpolation and preservation in guarded logics," in *CSL-LICS*, 2014.
[7] L. Henkin, "An extension of the Craig-Lyndon interpolation theorem," *The Journal of Symbolic Logic*, vol. 28, no. 3, pp. 201–216, 1963.
[8] G. D'Agostino and M. Hollenberg, "Logical Questions Concerning The mu-Calculus: Interpolation, Lyndon and Los-Tarski," *The Journal of Symbolic Logic*, vol. 65, no. 1, pp. 310–332, 2000.
[9] C. Lutz, I. Seylan, and F. Wolter, "An automata-theoretic approach to uniform interpolation and approximation in the description logic EL," in *KR*, 2012.
[10] P. Koopmann and R. Schmidt, "Count and forget: Uniform interpolation of SHQ-ontologies," in *IJCAR*, 2014.
[11] E. Grädel, C. Hirsch, and M. Otto, "Back and forth between guarded and modal logics," *ACM TOCL*, vol. 3, no. 3, pp. 418–463, 2002.
[12] D. Janin and I. Walukiewicz, "On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic," in *CONCUR*, 1996.
[13] B. ten Cate and L. Segoufin, "Unary negation," in *STACS*, 2011.
[14] A. Arnold and D. Niwiński, *Rudiments of mu-calculus*. North Holland, 2001.
[15] V. Bárány, B. ten Cate, and L. Segoufin, "Guarded negation," in *ICALP*, 2011.
[16] G. D. Giacomo, "Decidability of class-based knowledge representation formalisms," Ph.D. dissertation, Universitá di Roma "La Sapienza", 1995.
[17] U. Sattler and M. Y. Vardi, "The hybrid μ-calculus," in *IJCAR*, 2001.
[18] J. Bradfield and C. Stirling, "Modal mu-calculi," in *Handbook of Modal Logic*. Elsevier, 2007, pp. 721–756.
[19] H. Andréka, I. Németi, and J. van Benthem, "Modal languages and bounded fragments of predicate logic," *Journal of Philosophical Logic*, vol. 27, no. 3, pp. 217–274, 1998.
[20] D. Janin and I. Walukiewicz, "Automata for the modal mu-calculus and related results," in *MFCS*, 1995.
[21] L. J. Stockmeyer, "The complexity of decision problems in automata theory and logic," Ph.D. dissertation, Department of Electrical Engineering, MIT, 1974.
[22] M. Y. Vardi, "Reasoning about the past with two-way automata," in *ICALP*, 1998.
[23] A. Dawar, "A restricted second order logic for finite structures," *Inf. Comput.*, vol. 143, no. 2, pp. 154–174, 1998.
[24] B. Courcelle, "The expression of graph properties and graph transformations in monadic second-order logic," in *Handbook of Graph Grammars and Computing by Graph Transformations*, vol. 1, 1997, pp. 313–400.
[25] E. Hoogland, M. Marx, and M. Otto, "Beth definability for the guarded fragment," in *LPAR*, 1999.

[26] W. Thomas, “Languages, Automata, and Logic,” in *Handbook of Formal Languages*, G. Rozenberg and A. Salomaa, Eds., 1997.
[27] E. Grädel and I. Walukiewicz, “Guarded fixed point logic,” in *LICS*, 1999.
[28] I. Walukiewicz, “Automata and logic,” 2001, available at http://www.labri.fr/perso/igw/Papers/igw-eefss01.pdf.
[29] E. A. Emerson and C. S. Jutla, “The complexity of tree automata and logics of programs (extended abstract),” in *FOCS*, 1988.

APPENDIX A
UN^k-BISIMULATION GAME

A. *Proof of Proposition 7 (Connection between* UNFP^k *and* UN^k*-bisimulation game)*

Recall the statement:

> Assume Duplicator has a winning strategy in the UN$^k[\sigma]$-bisimulation game between $\mathfrak{A}$ and $\mathfrak{B}$. If $\varphi(x)$ is a UNFP$^k[\sigma]$ sentence in normal form, then $\mathfrak{A} \models \varphi$ iff $\mathfrak{B} \models \varphi$.

We first prove the following result about *infinitary* UNF^k, the extension of UNF with infinitary disjunctions and conjunctions.

Proposition 28. *Assume Duplicator has a winning strategy in the* UN$^k[\sigma]$*-bisimulation game between* $\mathfrak{A}$ *and* $\mathfrak{B}$ *starting from an interface position* $a \mapsto b$*. If* $\varphi(x)$ *is an infinitary* UNF$^k[\sigma]$ *formula in normal form with at most one free variable, then* $\mathfrak{A} \models \varphi(a)$ *iff* $\mathfrak{B} \models \varphi(b)$.

Proof: We proceed by induction on the structure of φ. The base case for an atomic formula, or boolean combination of atomic formulas, is trivial.

Consider φ of the form $\exists \boldsymbol{y}.\psi(x, \boldsymbol{y})$, where ψ is a boolean combination of formulas χ that are atomic or have at most one free variable. Let $\boldsymbol{c}$ be the tuple of elements from $\mathfrak{A}$ such that $\mathfrak{A} \models \psi(a, \boldsymbol{c})$, and consider Duplicator's strategy in the game when Spoiler selects $\boldsymbol{c}$ in $\mathfrak{A}$. Her strategy specifies a corresponding tuple $\boldsymbol{d}$ in $\mathfrak{B}$ such that $f : a\boldsymbol{c} \mapsto b\boldsymbol{d}$ is a partial unary-rigid homomorphism. Now consider some χ in ψ that holds in $\mathfrak{A}$. If χ is atomic, then the inductive hypothesis ensures that χ also holds in $\mathfrak{B}$. Otherwise, χ must have at most one free variable, say c'. Duplicator has a winning strategy in the game when Spoiler collapses to c', so if χ begins with an existential quantifier, the inductive hypothesis implies that χ also holds in $\mathfrak{B}$ for element $f(c')$. Likewise, Duplicator has a winning strategy in the game when Spoiler collapses to c' and switches structures, so if χ begins with a negation, the inductive hypothesis implies that if χ does not hold in $\mathfrak{B}$ for element $f(c')$, then it does not hold in $\mathfrak{A}$ for c, so it must be the case that χ holds in $\mathfrak{B}$ as desired. ■

Proposition 28 implies $\mathfrak{A}$ and $\mathfrak{B}$ cannot distinguish between infinitary UNF$^k[\sigma]$-sentences. Moreover, for all UNFP$^k[\sigma]$-formulas φ and for all cardinals κ, there is an infinitary UNF$^k[\sigma]$-formula that is equivalent to φ on structures of cardinality at most κ: each least fixpoint subformula is replaced by an infinitary disjunction of the possible fixpoint approximants, up to the successor cardinal of $\kappa = \max\{\mathfrak{A}, \mathfrak{B}\}$, which is an upper bound on the closure ordinal where the fixpoint is reached. Therefore, Proposition 7 must hold.

APPENDIX B
μ-AUTOMATA

A. *Automaton models*

We will make use of automata on trees in Appendix C and D. We assume familiarity with standard automata theory over infinite structures (see, e.g., [26]). However, the automata that we use are designed to work on trees with arbitrary branching, so they cannot refer to specific children of a node. This is different than traditional tree automata on binary trees that can refer to the left child and right child.

We define two automaton models, and state some properties that will be needed later.

Fix finite alphabets Σ_p and Σ_a for the node labels and edge labels, respectively.

A *2-way alternating μ-automaton* $\mathcal{A}$ is a tuple $\langle \Sigma_p, \Sigma_a, Q_E, Q_A, q_0, \delta, \Omega \rangle$ where $Q := Q_E \cup Q_A$ is a finite set of states partitioned into states Q_E controlled by Eve and states Q_A controlled by Adam, and $q_0 \in Q$ is the initial state. The transition function has the form

$$\delta : Q \times \Sigma_p \to \mathcal{P}(\mathrm{Dir} \times \Sigma_a \times Q)$$

where $\mathrm{Dir} = \{\uparrow, 0, \downarrow\}$ is the set of possible directions (up $\uparrow$, stay 0, down $\downarrow$). The acceptance condition is a parity condition specified by $\Omega : Q \to \mathrm{Pri}$, which maps each state to a priority in a finite set of priorities Pri.

We view $\mathcal{A}$ running on a tree $\mathcal{T}$ starting at node $v_0 \in \mathrm{dom}(\mathcal{T})$ as a game $\mathcal{G}(\mathcal{A}, \mathcal{T}, v_0)$. The arena is $Q \times \mathrm{dom}(\mathcal{T})$, and the initial position is (q_0, v_0). From a position (q, v) with $q \in Q_E$ (respectively, $q \in Q_A$), Eve (respectively Adam) selects $(d, a, r) \in \delta(q, \mathcal{T}(v))$, and an a-neighbor w of v in direction d (note if $d = 0$, then v is considered the only option). The game continues from position (r, w).

A *play* in $\mathcal{G}(\mathcal{A}, \mathcal{T}, v_0)$ is a sequence $(q_0, v_0), (q_1, v_1), (q_2, v_2), \ldots$ of moves in the game. Such a *play is winning* for Eve if the *parity condition* is satisfied: the maximum priority that occurs infinitely often in $\Omega(q_0), \Omega(q_1), \ldots$ is even.

A *strategy* for one of the players is a function that returns the next choice for that player given the history of the play. If the function depends only on the current position (rather than the full history), then it is *positional*. Choosing a strategy for both players fixes a play in $\mathcal{G}(\mathcal{A}, \mathcal{T}, v_0)$. A play π is *compatible* with a strategy ζ if there is a strategy for the other play such that ζ and ζ' yield π. A *strategy is winning* for Eve if every play compatible with it is winning.

We write $L_{v_0}(\mathcal{A})$ for the set of trees $\mathcal{T}$ such that Eve has a winning strategy in $\mathcal{G}(\mathcal{A}, \mathcal{T}, v_0)$. If v_0 is the root of $\mathcal{T}$, then we often just write $L(\mathcal{A})$.

The *dual* of a 2-way alternating μ-automaton $\mathcal{A}$ is the automaton $\mathcal{A}'$ obtained from $\mathcal{A}$ by switching Q_A and Q_E, and incrementing each priority by 1 (i.e. $\Omega'(q) := \Omega(q) + 1$). This has the effect of switching the roles of the two players, so the resulting automaton accepts the complement of $L(\mathcal{A})$.

We are also interested in a type of automaton on trees with arbitrary branching that operates in a 1-way, nondeterministic fashion. A *μ-automaton* $\mathcal{M}$ is a tuple $\langle \Sigma_p, \Sigma_a, Q, q_0, \delta, \Omega \rangle$. where the transition function now has the form

$$\delta : Q \times \Sigma_p \to \mathcal{P}(\mathcal{P}(\Sigma_a \times Q)).$$

Again, the acceptance condition is a parity condition specified by Ω.

The operation of the automaton is not standard. As before, we view $\mathcal{A}$ running on a tree $\mathcal{T}$ starting at node $v_0 \in \mathrm{dom}(\mathcal{T})$ as a game $\mathcal{G}(\mathcal{A}, \mathcal{T}, v_0)$. The arena is $Q \times \mathrm{dom}(\mathcal{T})$, and the initial position is (q_0, v_0). From a position (q, v), Eve selects some $S \in \delta(q, \mathcal{T}(v))$, and a marking of every successor of v with a set of states such that (i) for all $(a, r) \in S$, there is some a-successor whose marking includes r, and (ii) for all a-successors w of v, if r is in the marking of w, then there is some $(a, r) \in S$. Adam then selects some successor w of v and a state r in the marking of w chosen by Eve, and the game continues from position (r, w). A winning play and strategy is defined as above.

We say a μ-automaton is *action-deterministic* if for all $q \in Q$, $\tau \in \Sigma_p$, and $S \in \delta(q, \tau)$, there is exactly one element of the form $(\rho, r) \in S$ for each $\rho \in \Sigma_a$. In this case, when in position (q, v) in the game, Eve selects some $S \in \delta(q, \mathcal{T}(v))$ and Adam selects some successor w of v; suppose w is a ρ-successor of v. Play continues from (r, w) where (ρ, w) is the unique element in S with action ρ.

These μ-automata are described in more detail in [20] and [8]. The 2-way alternating μ-automata are similar to the automata used in [27].

B. Properties of μ-automata

We first observe that these automata are bisimulation-invariant on trees.

Proposition 29. *Let $\mathcal{A}$ be a 2-way alternating μ-automaton or a μ-automaton. For all trees $\mathcal{T}$, if $\mathcal{T} \in L(\mathcal{A})$ and $\mathcal{T}'$ is bisimilar to $\mathcal{T}$, then $\mathcal{T}' \in L(\mathcal{A})$.*

Both automata models are closed under boolean operations and other language operations. We mention here just the operations that we need in this work.

For 2-way alternating μ-automata, we make use of the following properties.

Proposition 30. *2-way alternating μ-automata are closed under:*

- *Intersection: Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be 2-way alternating μ-automata. Then we can construct a 2-way alternating μ-automaton $\mathcal{A}$ such that $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$, and the size of $\mathcal{A}$ is linear in $|\mathcal{A}_1| + |\mathcal{A}_2|$.*
- *Union: Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be 2-way alternating μ-automata. Then we can construct a 2-way alternating μ-automaton $\mathcal{A}$ such that $L(\mathcal{A}) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$, and the size of $\mathcal{A}$ is linear in $|\mathcal{A}_1| + |\mathcal{A}_2|$.*
- *Complement: Let $\mathcal{A}$ be a 2-way alternating μ-automaton. Then we can construct a 2-way alternating μ-automaton $\mathcal{A}'$ of size at most $|\mathcal{A}|$ such that $L(\mathcal{A}')$ is the complement of $L(\mathcal{A})$.*

Proof: These are standard constructions for alternating automata.

For intersection, we can just take the disjoint union of the two automata, and create a new initial state q_0 controlled by Adam with moves to stay in the same position and go to state $q_0^{\mathcal{A}_1}$, or stay in the same position and go to state $q_0^{\mathcal{A}_2}$. Depending on this initial choice, the automaton then simulates either $\mathcal{A}_1$ or $\mathcal{A}_2$. The construction for the union is similar, but the initial choice is given to Eve, rather than Adam.

For complement, we use the dual automaton described earlier, which requires switching Q_A and Q_E, and incrementing the priority mapping by 1. ■

For μ-automata, we need the following properties.

Proposition 31. *μ-automata are closed under the following operations.*

- *Intersection: Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be μ-automata. Then we can construct a μ-automaton $\mathcal{M}$ such that $L(\mathcal{M}) = L(\mathcal{M}_1) \cap L(\mathcal{M}_2)$. If at least one of $\mathcal{M}_1$ and $\mathcal{M}_2$ is action-deterministic, then the size of $\mathcal{M}$ is at most $|\mathcal{M}_1| \cdot |\mathcal{M}_2|$.*
- *Projection, modulo bisimulation: Let $\mathcal{M}$ be a μ-automaton over propositions Props, and let $P \in$ Props. Then we can construct a μ-automaton $\mathcal{M}'$ over propositions Props $\setminus \{P\}$ of size at most $|\mathcal{M}|$ such that for all trees $\mathcal{T}$, $\mathcal{T} \in L(\mathcal{M})$ iff there exists some $\mathcal{T}' \in L(\mathcal{M}')$ and $\mathcal{T}'$ is (Props $\setminus \{P\}$)-bisimilar to $\mathcal{T}$.*

Proof: In general, closure under intersection follows from the equivalence between the μ-calculus and μ-automata [20], and the fact that the μ-calculus has conjunction. With at least one action-deterministic automaton, the result can be obtained using a product construction.

Closure under projection follows from [8, Theorem 3.3]. ■

We can convert any μ-automaton into a well-structured L_μ-formula.

Proposition 32. *Given a μ-automaton $\mathcal{A}$ over consistent $\tilde{\sigma}_k$ trees we can construct a well-structured $\psi \in \mathrm{L}_\mu[\tilde{\sigma}_k]$ of size polynomial in the size of $\mathcal{A}$ such that $\mathcal{T} \in L(\mathcal{A})$ iff $\mathcal{T} \models \psi$.*

Proof: We can assume without loss of generality that the states of the automaton in interface nodes are disjoint from the states of the automaton in bag nodes, so we will refer to *interface states* and *bag states* as appropriate. Because nodes along a branch in a consistent tree alternate between interface nodes and bag nodes, we can also assume bag states are assigned priority 0, and the initial state is assigned the maximum priority. Converting an automaton into a version satisfying these assumptions results in only a polynomial blow-up.

We can then write in the usual way (see, e.g., [28]) a vectorial L_μ-formula ψ describing the operation of $\mathcal{A}$:

$$\lambda_n X_{q_n} \dots \lambda_1 X_{q_1} . \begin{pmatrix} \delta_{q_1} \\ \vdots \\ \delta_{q_n} \end{pmatrix}$$

where $q_1, \dots, q_n$ is an ordering of the states based on the priority (from least to greatest priority), λ_i is μ (respectively, ν) if q_i has an odd (respectively, even) priority, and δ_q are transition formulas defined below. We can assume that the ordering is chosen so that (for some i) $q_1, \dots, q_i$ consists of bag states, no bag states are present in $q_{i+1}, \dots, q_n$, and q_n is the initial state.

The transition formulas δ_q are defined as follows:

$$\delta_q := \bigvee_{\tau} \left[\left(\bigwedge_{P \in \tau} P \wedge \bigwedge_{P \in Props_k \setminus \tau} \neg P \right) \wedge \delta_{q,\tau} \right]$$

and

$$\delta_{q,\tau} := \bigvee_{S \in \delta(q,\tau)} \left(\bigwedge_{(\rho,r) \in S} \langle \rho \rangle X_r \wedge \bigwedge_{\rho \in Actions_k} [\rho] \bigvee_{(\rho,r) \in S} X_r \right).$$

This captures precisely the meaning of the transition function in a μ-automaton. The idea is that it picks out exactly the label τ at the current node, and then ensures that the successors of this node satisfy the requirements specified by the transition function when in state q and at a position with label τ.

This fits the well-structured form defined in the body of the paper. ■

Appendix C
Initial Algorithm

A. Details on consistent trees

We give here the formal definition of a *consistent $\tilde{\sigma}_k$-tree*, which was described informally in Section III. A $\tilde{\sigma}_k$-tree is *consistent* if it satisfies the following conditions:

- D_0, D_1, D_k partition $\mathrm{dom}(\mathcal{T})$;
- the root ϵ is an empty interface node;
- if v is at an even (respectively, odd) depth then v is an interface node (respectively, bag node);
- if $v \in D_0$, then $v \notin R_{\boldsymbol{i}}$ for all $R \in \sigma$ of arity r and $\boldsymbol{i} \in \{1, \dots, k\}^r$;
- if $v \in D_1$, then $v \notin R_{\boldsymbol{i}}$ for all $\boldsymbol{i} \cap \{2, \dots, k\} \neq \emptyset$;
- if v is an empty interface node, then $(u, v) \in E_\rho$ or $(v, u) \in E_\rho$ implies that ρ is the empty map;
- if v is a non-empty interface node, then $(u, v) \in E_\rho$ (respectively, $(v, u) \in E_\rho$) implies that $\mathrm{rng}(\rho) = \{1\}$ (respectively, $\mathrm{dom}(\rho) = \{1\}$);
- if $(u, v) \in E_\rho$, then for all $R \in \sigma$ of arity r and for all $\boldsymbol{i} \in \mathrm{dom}(\rho)^r$, $u \in R_{\boldsymbol{i}}$ iff $v \in R_{\rho(\boldsymbol{i})}$.

B. Proof of Lemma 12 (Shrewd unravelling)

Recall the statement of the lemma:

For all σ-structures $\mathfrak{A}$, $\mathcal{S}^k(\mathfrak{A})$ is a shrewd consistent $\tilde{\sigma}_k$-tree, and $\mathfrak{D}(\mathcal{S}^k(\mathfrak{A}))$ is $\mathrm{UN}^k[\sigma]$-bisimilar to $\mathfrak{A}$.

We give the proof for the second part, that $\mathfrak{D}(\mathcal{S}^k(\mathfrak{A}))$ is $\mathrm{UN}^k[\sigma]$-bisimilar to $\mathfrak{A}$. Let $\mathfrak{A}' = \mathfrak{D}(\mathcal{S}^k(\mathfrak{A}))$.

Let $\mathrm{elem}_{\mathfrak{A}'}(\pi)$ denote the elements in $\mathfrak{A}'$ represented at node π. Similarly, let $\mathrm{elem}_{\mathfrak{A}'}(\pi, j)$ denote the element in $\mathfrak{A}'$ that is indexed by j in π. Let X' be a set of elements from $\mathfrak{A}'$ of size at most 1. Let $\mathrm{elem}_{\mathfrak{A}}(\pi)$, $\mathrm{elem}_{\mathfrak{A}}(\pi, j)$, X be the analogous definitions for $\mathfrak{A}$. A position $X \mapsto X'$ or $X' \mapsto X$ in the $\mathrm{UN}^k[\sigma]$-bisimulation game is *covered* if there is a node π in $\mathcal{S}^k(\mathfrak{A})$ such that π ends in an interface position with $\mathrm{elem}_{\mathfrak{A}'}(\pi) = X'$ and $\mathrm{elem}_{\mathfrak{A}}(\pi) = X$. Note that if a position f is covered, then f must be a partial isomorphism.

We claim that Duplicator has a winning strategy in the $\mathrm{UN}^k[\sigma]$-bisimulation game between $\mathfrak{A}$ and $\mathfrak{A}'$ starting from any covered position, and in particular from the covered position $\emptyset \mapsto \emptyset$. We present the proof only for the case when we start from a non-empty partial homomorphism; the other cases are similar.

Active structure $\mathfrak{A}'$: Consider first the case when we start in a covered position $a' \mapsto a$. Assume Spoiler chooses $\boldsymbol{b}' = b'_1 \dots b'_k$ of size k in $\mathfrak{A}'$. We can assume without loss of generality that $b'_1 = a'$. Set $b_1 := a$. For each b'_i, let π_i be a node where $b'_i = \mathrm{elem}_{\mathfrak{A}'}(\pi_i, j)$ for some j, and set $b_i := \mathrm{elem}_{\mathfrak{A}}(\pi_i, j)$. Set $\boldsymbol{b} := b_1 \dots b_k$.

We claim $\boldsymbol{b}' \mapsto \boldsymbol{b}$ is a partial unary-rigid homomorphism from $\mathfrak{A}'$ to $\mathfrak{A}$.

Assume that $\mathfrak{A}' \models R b'_{i_1} \dots b'_{i_r}$. Then there is some bag node π ending in (Y, τ') and some indices $j_1, \dots, j_r$ such that $R_{j_1 \dots j_r} \in \tau'$, $b'_{i_m} = \mathrm{elem}_{\mathfrak{A}'}(\pi, j_m)$, and $b_{i_m} = \mathrm{elem}_{\mathfrak{A}}(\pi, j_m)$. Note that this means that b_{i_m} is represented in all nodes along the path from π to π_{i_1} (but this element could be indexed differently in the nodes along this path, and in particular is indexed by j_m at π). Hence, $\{b_{i_1}, \ldots, b_{i_r}\} \subseteq Y$. Since $\tau' \subseteq \tau(Y)$, this implies that $R_{j_1 \ldots j_r} \in \tau(Y)$ and $\mathfrak{A} \models Rb_{i_1} \ldots b_{i_r}$. This ensures that $\boldsymbol{b}' \mapsto \boldsymbol{b}$ is a partial homomorphism.

Now assume that $\mathfrak{A} \models Rb_i \ldots b_i$ (so only one element from $\boldsymbol{b}$ is used). Recall that π_i is the node such that there is some j with $b'_i = \mathrm{elem}_{\mathfrak{A}'}(\pi_i, j)$ and $b_i = \mathrm{elem}_{\mathfrak{A}}(\pi_i, j)$. If π_i ends in an interface position Y, then by definition of $\mathcal{S}^k(\mathfrak{A})$, $\pi_i \in R^{\mathcal{S}^k(\mathfrak{A})}_{j \ldots j}$ and $\mathfrak{A}' \models Rb'_i \ldots b'_i$. Likewise, if π_i ends in a bag postion (Y, τ') then $\tau(Y) \cap \mathit{Props}^1_k = \tau' \cap \mathit{Props}^1_k$, so $R_{j \ldots j} \in \tau'$ and $\mathfrak{A}' \models Rb'_i \ldots b'_i$. Combined with the previous paragraph, this ensures that $\boldsymbol{b}' \mapsto \boldsymbol{b}$ is a partial unary-rigid homomorphism.

We claim that when Spoiler collapses to the empty set or a single element, the corresponding position in the game is covered. If Spoiler moves to the empty partial homomorphism (from $\mathfrak{A}$ to $\mathfrak{A}'$ or $\mathfrak{A}$ to $\mathfrak{A}'$) then we are clearly in a covered position. If Spoiler collapses to b'_i or b_i, then by choice of $\boldsymbol{b}$, we know that there is some node π_i and some j with $b'_i = \mathrm{elem}_{\mathfrak{A}'}(\pi_i, j)$ and $b_i = \mathrm{elem}_{\mathfrak{A}'}(\pi_i, j)$. If π_i is an interface node, then we are done. Otherwise, by construction of $\mathcal{S}^k(\mathfrak{A})$, there is a successor node π' of π such that $\mathrm{elem}_{\mathfrak{A}'}(\pi', 1) = b'_i$ and $\mathrm{elem}_{\mathfrak{A}}(\pi, 1) = b_i$, so we are in a covered position.

Active structure $\mathfrak{A}$: Next consider the case when we start in a covered position $a \mapsto a'$. Then there is some interface node π with $\mathrm{elem}_{\mathfrak{A}}(\pi, 1) = a$ and $\mathrm{elem}_{\mathfrak{A}'}(\pi, 1) = a'$. Assume Spoiler chooses $\boldsymbol{b} = b_1 \ldots b_k$ of size k in $\mathfrak{A}$. There is a successor $\pi' = \pi(\boldsymbol{b}, \tau(\boldsymbol{b}))$ of π in $\mathcal{S}^k(\mathfrak{A})$ such that $\mathrm{elem}_{\mathfrak{A}}(\pi') = \boldsymbol{b}$ and $\mathrm{elem}_{\mathfrak{A}}(\pi', i) = b_i$. For each $b_i \in \boldsymbol{b}$, let $b'_i := \mathrm{elem}_{\mathfrak{A}'}(\pi', i)$. Set $\boldsymbol{b}' := b'_1 \ldots b'_k$.

By definition of $\mathfrak{D}(\mathcal{S}^k(\mathfrak{A}))$, it is easy to see that $\boldsymbol{b} \mapsto \boldsymbol{b}'$ is a partial isomorphism, and hence a partial unary-rigid homomorphism as desired.

If Spoiler moves to the empty partial homomorphism (from $\mathfrak{A}$ to $\mathfrak{A}'$ or $\mathfrak{A}$ to $\mathfrak{A}'$) then we are clearly in a covered position. If Spoiler collapses to b'_i or b_i, then we know by construction of $\mathcal{S}^k(\mathfrak{A})$ that there is some successor π''_i of π' such that $\mathrm{elem}_{\mathfrak{A}}(\pi''_i, 1) = b_i$ and $\mathrm{elem}_{\mathfrak{A}'}(\pi''_i, 1) = b'_i$, so we are again in a covered position.

C. Proof sketch of Theorem 13 (Forward mapping)

Recall the statement:

> Let ψ be a sentence in $\mathrm{GSO}[\sigma]$. For all k, we can construct a sentence $\psi^{\rightarrow} \in \mathrm{MSO}[\tilde{\sigma}_k]$ such that for all consistent $\tilde{\sigma}_k$ trees $\mathcal{T}$, $\mathfrak{D}(\mathcal{T}) \models \psi$ iff $\mathcal{T} \models \psi^{\rightarrow}$.

During the translation, we must deal with formulas with free variables, so we describe how these are coded.

A first-order variable x is encoded by a sequence of second-order variables $x^{\rightarrow} = (Z^x_i)_{i \in \{1, \ldots, k\}}$. The set Z^x_i consists of the nodes v in the coded structure where the i-th element in v corresponds to the element identified by x.

Likewise, a second-order variable X corresponding to an r-ary guarded relation (a relation that only includes tuples guarded in σ) is encoded by a sequence of second-order variables $X^{\rightarrow} = (Z^X_{\boldsymbol{i}})_{\boldsymbol{i} \in \{1, \ldots, k\}^r}$. The set $Z^X_{\boldsymbol{i}}$ corresponds to the set of nodes v in the coded structure where the elements coded by $\boldsymbol{i}$ are in X.

It is straightforward to construct the following auxiliary formulas that check whether some tuple of second-order variables actually encodes a first-order variable or a guarded relation in the way we just described.

Lemma 33. *There is a formula* $\mathrm{correct}(x^{\rightarrow}) \in \mathrm{MSO}[\tilde{\sigma}_k]$ *such that for all consistent $\tilde{\sigma}_k$-trees $\mathcal{T}$ and for all $j^{\rightarrow} = (J_i)_{i \in \{1, \ldots, k\}}$, $\mathcal{T} \models \mathrm{correct}(j^{\rightarrow})$ iff there is some element a in $\mathfrak{D}(\mathcal{T})$ such that for all i, $J_i = \{v \in \mathcal{T} : [v, i] = a\}$.*

There is a formula $\mathrm{correct}_r(X^{\rightarrow}) \in \mathrm{MSO}[\tilde{\sigma}_k]$ *such that for all consistent $\tilde{\sigma}_k$-trees $\mathcal{T}$ and for all $J^{\rightarrow} = (J_{\boldsymbol{i}})_{\boldsymbol{i} \in \{1, \ldots, k\}^r}$, $\mathcal{T} \models \mathrm{correct}_r(J^{\rightarrow})$ iff there is some guarded relation J of arity r on $\mathfrak{D}(\mathcal{T})$ and for all $\boldsymbol{i} = i_1 \ldots i_r$, $J_{\boldsymbol{i}} = \{v \in \mathcal{T} : ([v, i_1], \ldots, [v, i_r]) \in J\}$.*

Using these auxiliary formulas, we can perform the forward translation.

Lemma 34. *Let ψ be a formula in $\mathrm{GSO}[\sigma]$ with free first-order variables among $x_1, \ldots, x_k$, and free second-order variables among $X_1, \ldots, X_m$. We can construct a formula $\psi^{\rightarrow}(x_1{}^{\rightarrow}, \ldots, x_k{}^{\rightarrow}, X_1{}^{\rightarrow}, \ldots, X_m{}^{\rightarrow}) \in \mathrm{MSO}[\tilde{\sigma}_k]$ such that for all consistent $\tilde{\sigma}_k$-trees $\mathcal{T}$, for all elements $a_1, \ldots, a_k$ in $\mathfrak{D}(\mathcal{T})$ encoded by $j_1{}^{\rightarrow}, \ldots, j_k{}^{\rightarrow}$ and for all sets of guarded relations $J_1, \ldots, J_m$ encoded by $J_1{}^{\rightarrow}, \ldots, J_m{}^{\rightarrow}$,*

$$\begin{aligned}&\mathfrak{D}(\mathcal{T}), a_1, \ldots, a_k, J_1, \ldots, J_m \models \psi \textit{ iff}\\ &\mathcal{T}, j_1{}^{\rightarrow}, \ldots, j_k{}^{\rightarrow}, J_1{}^{\rightarrow}, \ldots, J_m{}^{\rightarrow} \models \psi^{\rightarrow}.\end{aligned}$$

Proof: The proof is by induction on the structure of ψ.

- Assume $\psi = Rx_{i_1} \ldots x_{i_n}$. Then

$$\psi^{\rightarrow} := \exists z. \bigvee_{\rho \in \mathit{Actions}_k} \left(z \in R_{\rho(i_1 \ldots i_n)} \wedge \bigwedge_{i \in \{i_1, \ldots, i_n\}} z \in Z^{x_i}_{\rho(i)} \right).$$

  This expresses that there is some node in the coded structure where R holds for elements coded by $\rho(i_1 \ldots i_n)$, and these elements are precisely $x_{i_1} \ldots x_{i_n}$.
  Similarly for $\psi = Xx_{i_1} \ldots x_{i_n}$.
- The translation commutes with $\vee$, $\wedge$, and $\neg$.
- Assume $\psi = \exists x. \chi$. Then

$$\psi^{\rightarrow} := \exists x^{\rightarrow}. \left(\mathrm{correct}(x^{\rightarrow}) \wedge \chi^{\rightarrow} \right).$$

- Assume $\psi = \exists X. \chi$ for X an r-ary relation. Then

$$\psi^{\rightarrow} := \exists X^{\rightarrow}. \left(\mathrm{correct}_r(X^{\rightarrow}) \wedge \chi^{\rightarrow} \right).$$

■

The theorem follows directly from the previous lemma, since the input is a sentence with no free first or second-order variables.

D. Proof of Theorem 15 (Simplification)

Recall the statement of the theorem:

> Let $\psi \in \text{MSO}[\tilde{\sigma}_k]$ be $\tilde{\sigma}_k$-bisimulation invariant over the class of $\tilde{\sigma}_k$-trees. Then we can effectively obtain a well-structured L_μ-formula ψ' such that for all $\tilde{\sigma}_k$-trees $\mathcal{T}$, $\mathcal{T} \models \psi$ iff $\mathcal{T} \models \psi'$.

Apply Theorem 6. Then use [20] to convert ψ to a μ-automaton $\mathcal{A}$, and use Proposition 32 to convert to a well-structured form.

E. Proof of Theorem 17 (Backward mapping)

Recall the statement:

> Let ψ be a well-structured formula in $\text{L}_\mu[\tilde{\sigma}_k]$. We can construct a sentence $\psi^{\leftarrow} \in \text{UNFP}^k[\sigma]$ such that for all σ-structures $\mathfrak{B}$, $\mathfrak{B} \models \psi^{\leftarrow}$ iff $\mathcal{S}^k(\mathfrak{B}) \models \psi$.

Fix well-structured $\psi \in \text{L}_\mu[\tilde{\sigma}_k]$. Our goal in this section is to produce the $\text{UNFP}^k[\sigma]$ sentence $\psi^{\leftarrow}$. We will first convert to a special normal form which we call "UNFP-safe L_μ". This requires a series of transformations, described below. At each stage, we ensure equivalence with ψ, at least over shrewd consistent trees. Step 3 is the most interesting, because it relies on the fact that we are working with shrewd trees.

Before defining UNFP-safety, we need a way to talk about the parts of an L_μ-formula that will lead to free first-order variables in the corresponding UNFP-formula. We need to have an understanding of where these free variables come from, since negation in UNFP can only be applied to formulas with at most one free variable. Roughly speaking, variables in the $\text{UNFP}^k[\sigma]$-formula will come from the indices that are being used by relations $R_{\boldsymbol{i}}$ in the $\text{L}_\mu[\tilde{\sigma}_k]$-formula. For this reason, we define $\text{indices}(\chi)$ for $\chi \in \text{L}_\mu[\tilde{\sigma}_k]$ inductively as follows:

- $\text{indices}(R_{i_1 \dots i_n}) := \{i_1, \dots, i_n\}$;
- $\text{indices}(I) = \text{indices}(D_n) = \text{indices}(X) = \text{indices}(\top) = \text{indices}(\bot) := \emptyset$;
- $\text{indices}(\chi_1 \wedge \chi_2) = \text{indices}(\chi_1 \vee \chi_2) := \text{indices}(\chi_1) \cup \text{indices}(\chi_2)$;
- $\text{indices}(\neg\chi) := \text{indices}(\chi)$;
- $\text{indices}(\langle\rho\rangle\chi) = \text{indices}([\rho]\chi) := \text{dom}(\rho)$;
- $\text{indices}(\lambda X.\chi) := \text{indices}(\chi)$.

We say an $\text{L}_\mu[\tilde{\sigma}_k]$ formula is UNFP-*safe for interface nodes* (respectively, UNFP-*safe for bag nodes*) if

- there are no explicit ν-fixpoints or box-modalities;
- negation is only applied to subformulas χ where $|\text{indices}(\chi)| \leq 1$;
- for every subformula χ in the scope of an even (respectively, odd) number of modalities, $\text{indices}(\chi)$ is contained in $\{1\}$ (respectively, $\{1, \dots, k\}$);
- every fixpoint subformula or fixpoint variable is in the scope of an even (respectively, odd) number of modalities (i.e. fixpoints reference only interface nodes).

We now proceed with the series of transformations taking ψ to a UNFP-safe version that is equivalent at least over shrewd consistent trees.

Step 1: We are given a well-structured formula $\psi^1 := \psi$. For reference later, we can assume that it is in the following form:

$$\lambda_n X_{q_n} \dots \lambda_1 X_{q_1}. \begin{pmatrix} \delta_{q_1} \\ \vdots \\ \delta_{q_n} \end{pmatrix}$$

such that $\lambda_i \in \{\mu, \nu\}$ are fixpoints,

$$\delta_q := \bigvee_{\tau} \left[\left(\bigwedge_{P \in \tau} P \wedge \bigwedge_{P \in Props_k \setminus \tau} \neg P \right) \wedge \delta_{q,\tau} \right]$$

$$\delta_{q,\tau} := \bigvee_{S \in \delta(q,\tau)} \left(\bigwedge_{(\rho,r) \in S} \langle\rho\rangle X_r \wedge \bigwedge_{\rho \in Actions_k} [\rho] \bigvee_{(\rho,r) \in S} X_r \right).$$

and there is some i such that the fixpoint predicates $X_{q_1}, \dots, X_{q_i}$ only contain bag nodes (call these *bag predicates*), and $X_{q_{i+1}}, \dots, X_{q_n}$ only contain interface nodes (call these *interface predicates*).

Step 2: Consistent tree simplification:

Claim 35. *There is a formula ψ^2 obtained effectively from ψ^1 such that for all interface predicates (respectively, bag predicates) X_q, the vectorial components δ^2_q in ψ^2 satisfy* $\text{indices}(\delta^2_q) \subseteq \{1\}$ *(respectively,* $\text{indices}(\delta^2_q) \subseteq \{1, \dots, k\}$*). Moreover, for all consistent $\tilde{\sigma}_k$-trees $\mathcal{T}$ and for all interface nodes v, we have $\mathcal{T}, v \models \psi^2$ iff $\mathcal{T}, v \models \psi^1$.*

Proof: For all interface predicates X_q, we perform the following simplifications of δ_q in ψ^1.

- Substitute $\bot$ for any unary predicate P with $\text{indices}(P) \cap \{2, \dots, k\} \neq \emptyset$.
- Substitute $\bot$ for any $\langle\rho\rangle X_r$ with $\text{dom}(\rho) \not\subseteq \{1\}$.
- Substitute $\top$ for any $[\rho]\chi$ with $\text{dom}(\rho) \not\subseteq \{1\}$.

Likewise, for all bag predicates X_r, we perform the following simplifications of δ_r in ψ^1.

- Substitute $\bot$ for any $\langle\rho\rangle X_q$ with $\text{rng}(\rho) \not\subseteq \{1\}$.
- Substitute $\top$ for any $[\rho]\chi$ with $\text{rng}(\rho) \not\subseteq \{1\}$.

The definition of consistent $\tilde{\sigma}_k$-trees requires that interface nodes have labels using only indices from $\{1\}$, and edges that respect this domain, so the resulting formula still reflects the operation of the automaton at an interface node. Let ψ^2 be the resulting formula, which has vectorial components δ^2_q. ■

Step 3: Shrewd tree simplification: We would like to eliminate the use of any fixpoint variables in ψ^2 that correspond to bag predicates X_r. We can do this by substituting δ^2_r for any occurrence of X_r in the other transition formulas.

After doing this substitution, the subformulas $\langle\rho\rangle\delta^2_r$ and $[\rho]\bigvee_{(\rho,r)\in S}\delta^2_r$ both may have negations that are not allowed in UNFP-safe formulas. For instance, in $\langle\rho\rangle\delta^2_r$ there may be some $\neg P$ for $P \in \mathit{Props}_k \setminus \tau$ with $|\text{indices}(P)| > 1$. This is because the transition function described by δ^2_r depends on knowing exactly the label τ at the current node. In a shrewd tree, however, this is not really necessary: we only need to know the propositions in Props^1_k exactly. Indeed, by the properties of shrewd trees, we can simplify the formula so that (among other things) we can replace $\bigwedge_{P\in \mathit{Props}_k\setminus\tau} \neg P$ with $\bigwedge_{P\in \mathit{Props}^1_k\setminus\tau} \neg P$.

We will use the following auxiliary formulas to help with this process.

$$\delta^{\Diamond}_r := \bigvee_{\tau}\left[\left(\bigwedge_{P\in\tau} P \wedge \bigwedge_{P\in \mathit{Props}^1_k\setminus\tau} \neg P\right)\wedge \delta_{r,\tau}\right]$$

$$\delta^{\Box}_S := \bigwedge_{\tau}\left[\left(\bigwedge_{P\in\tau} P \wedge \bigwedge_{P\in \mathit{Props}^1_k\setminus\tau} \neg P\right)\to \bigvee_{(\rho,r)\in S} \delta_{r,\tau}\right]$$

Claim 36. *Let v be an interface node in a shrewd consistent $\tilde{\sigma}_k$-tree $\mathcal{T}$. Then*

$$\mathcal{T}, v \models \langle\rho\rangle\delta^2_r \textit{ iff } \mathcal{T}, v \models \langle\rho\rangle\delta^{\Diamond}_r\,, \tag{1}$$

$$\mathcal{T}, v \models [\rho] \bigvee_{(\rho,r)\in S} \delta^2_r \textit{ iff } \mathcal{T}, v \models [\rho]\delta^{\Box}_S\,. \tag{2}$$

Proof of claim: Fix some shrewd consistent $\tilde{\sigma}_k$ tree $\mathcal{T}$, and some interface node v. We write $v \models \psi$ for $\mathcal{T}, v \models \psi$.

(1) Assume $v \models \langle\rho\rangle\delta^2_r$. Then there is some ρ-child w of v. Let τ be the collection of propositions from Props_k that hold at w. Then $w \models \delta_{r,\tau}$. Hence, in particular, $w \models \bigwedge_{P\in\tau} P \wedge \bigwedge_{P\in \mathit{Props}^1_k\setminus\tau} \neg P \wedge \delta_{r,\tau}$, which is enough to conclude that $w \models \delta^{\Diamond}_r$ and $v \models \langle\rho\rangle\delta^{\Diamond}_r$.
Next assume that $v \models \langle\rho\rangle\delta^{\Diamond}_r$. Then there is ρ-child w and some set τ of propositions such that $w \models \bigwedge_{P\in\tau} P \wedge \bigwedge_{P\in \mathit{Props}^1_k\setminus\tau} \neg P \wedge \delta_{r,\tau}$. By the properties of shrewd trees, there is some ρ-child w' of v such that τ describes exactly the propositions that hold at w', and the subtrees rooted at w and w' are isomorphic (ignoring the labels at w and w'). Hence, $w' \models \bigwedge_{P\in\tau} P \wedge \bigwedge_{P\in \mathit{Props}_k\setminus\tau} \neg P \wedge \delta_{r,\tau}$, which implies that $v \models \langle\rho\rangle\delta^2_r$.

(2) Assume $v \models [\rho]\bigvee_{(\rho,r)\in S} \delta^2_r$. Let w be a ρ-child of v. Consider any τ such that $w \models \bigwedge_{P\in\tau} P \wedge \bigwedge_{P\in \mathit{Props}^1_k\setminus\tau} \neg P$. It suffices to show that $w \models \bigvee_{(\rho,r)\in S} \delta_{r,\tau}$. By the property of shrewd trees, there is a ρ-child w' of v such that $w' \models \bigwedge_{P\in\tau} P \wedge \bigwedge_{P\in \mathit{Props}_k\setminus\tau} \neg P$, and (ignoring the labels at the roots), the subtrees rooted at w and w' are isomorphic. By the initial assumption, there is some $(\rho, r) \in S$ such that w' satisfies δ^2_r, and in particular, $w' \models \delta_{r,\tau}$. Since $\delta_{r,\tau}$ is not affected by the label at the current node, this means that $w \models \delta_{r,\tau}$ as desired.
Now assume $v \models [\rho]\delta^{\Box}_S$. Let w be a ρ-child of v, and let τ be the set of propositions that hold at w. Then by the definition of $\delta^{\Box}_S$, there is some $(\rho, r) \in S$, such that $w \models \delta_{r,\tau}$. Hence, $w \models \bigwedge_{P\in\tau} P \wedge \bigwedge_{P\in \mathit{Props}_k\setminus\tau} \neg P \wedge \delta_{r,\tau}$, so $w \models \bigvee_{(\rho,r)\in S} \delta^2_r$. Overall, this means $v \models [\rho]\bigvee_{(\rho,r)\in S} \delta^2_r$ as desired.

■

We are now ready to describe the transformation for this step.

Claim 37. *There is a formula ψ^3 obtained effectively from ψ^2 such that*

- *negation is only applied explicitly to subformulas χ where* $|\text{indices}(\chi)| \le 1$,
- *for every subformula χ in the scope of an even (respectively, odd) number of modalities,* indices(χ) *is contained in* $\{1\}$ *(respectively,* $\{1, \dots, k\}$*),*
- *no bag predicates are used.*

Moreover, for all consistent $\tilde{\sigma}_k$-trees $\mathcal{T}$ and for all interface nodes v, we have $\mathcal{T}, v \models \psi^3$ iff $\mathcal{T}, v \models \psi^2$.

Proof: For all interface predicates X_q, and for all bag predicates X_r:

- substitute $\langle\rho\rangle\delta^{\Diamond}_r$ for $\langle\rho\rangle X_r$ in δ^2_q;
- substitute $[\rho]\delta^{\Box}_S$ for $[\rho]\bigvee_{(\rho,r)\in S} X_r$ in δ^2_q.

Then for all bag predicates X_r,

- remove λX_r and the component δ^2_r.

Let ψ^3 be the resulting formula, with vectorial components δ^3_q.

Claim 36 ensures that we can perform these substitutions while preserving equivalence over shrewd consistent trees. ■

Step 4: Clean up to obtain UNFP-*safe* L_μ*-formula:* The formula ψ^3 obtained in the previous step is almost UNFP-safe. We can now perform a number of clean-up operations to rewrite box modalities and ν-fixpoints in order to get the formula into UNFP-safe form.

Claim 38. *There is a formula ψ^4 obtained effectively from ψ^3 such that ψ^4 is* UNFP-*safe for interface nodes. Moreover, for all consistent $\tilde{\sigma}_k$-trees $\mathcal{T}$ and for all interface nodes v, we have $\mathcal{T}, v \models \psi^4$ iff $\mathcal{T}, v \models \psi^3$.*

Proof: To help eliminate the use of the box modality, we first define

$$\delta^{\neg\Diamond}_S := \bigvee_{\tau}\left[\left(\bigwedge_{P\in\tau} P \wedge \bigwedge_{P\in \mathit{Props}^1_k\setminus\tau} \neg P\right)\wedge \overline{\delta}^4_{r,\tau}\right]$$

where $\overline{\delta}^4_{r,\tau}$ is

$$\bigwedge_{S\in\delta(r,\tau)}\left[\bigvee_{\substack{(\rho',s)\in S \text{ s.t.}\\ \text{rng}(\rho')=\{1\}}} \neg\langle\rho'\rangle X_s \vee \bigvee_{\substack{\rho'\in \mathit{Actions}_k \text{ s.t.}\\ \text{rng}(\rho')=\{1\}}} \langle\rho'\rangle \bigwedge_{(\rho',s)\in S} \neg X_s\right].$$

By using the equivalence $[\rho]\chi \equiv \neg\langle\rho\rangle\neg\chi$ and pushing negations inside, it is easy to observe that $[\rho]\delta_S^{\Box} \equiv \neg\langle\rho\rangle\delta_S^{\neg\Diamond}$, and $\neg\langle\rho\rangle\delta_S^{\neg\Diamond}$ is UNFP-safe.

Finally, in order to obtain ψ^4, we perform the following operations:

- Substitute $\neg\langle\rho\rangle\delta_S^{\neg\Diamond}$ for any occurrence of $[\rho]\delta_S^{\Box}$.
- Substitute $\neg\langle\rho\rangle\bigwedge_{(\rho,s)\in S}\neg X_s$ for any remaining occurrences of $[\rho]\bigvee_{(\rho,s)\in S} X_s$.
- Convert the vectorial fixpoint to a standard L_μ-formula using the Bekič principle.
- Replace ν-fixpoints with μ-fixpoints using the equivalence $\nu X.\chi \equiv \neg\mu X.\neg\chi[\neg X/X]$. These negations are UNFP-safe because of the property that $\mathrm{indices}(\delta_q^3) \subseteq \{1\}$.

■

Putting it together: Based on the previous transformation steps, we have proven the following lemma.

Lemma 39. *The formula $\varphi := \psi^4$ obtained effectively from ψ according to steps* (1)–(4) *is* UNFP-*safe for interface nodes and such that for all shrewd consistent $\tilde{\sigma}_k$-trees $\mathcal{T}$,*

$$\mathcal{T} \models \varphi \text{ iff } \mathcal{T} \models \psi.$$

From a UNFP-safe formula in $\mathrm{L}_\mu[\tilde{\sigma}_k]$ we can produce $\mathrm{UNFP}^k[\sigma]$ formulas described below. Similar to [11], for each fixpoint variable X we introduce two fixpoint variables X_1 and X_0 to handle non-empty and empty interface nodes, respectively; we define $X^{\leftarrow} := (X_0, X_1)$. A set J of nodes in $\mathcal{S}^k(\mathfrak{B})$ is a UNFP-*safe valuation for a free variable X* if it (i) contains only interface nodes, and (ii) if it contains an empty interface node then it contains every empty interface node. We write $J^{\leftarrow}$ for its representation in $\mathfrak{B}$, i.e. $J^{\leftarrow} := (J_0, J_1)$ where J_1 is defined to be $\{\mathrm{elem}(v) : v \in J \text{ and } |\mathrm{elem}(v)| = 1\}$, and J_0 is $\top$ (respectively, $\bot$) if J contains all empty interface node (respectively, contains no empty interface nodes).

Lemma 40. *Let $\varphi \in \mathrm{L}_\mu[\tilde{\sigma}_k]$ be* UNFP-*safe for interface nodes (respectively, bag nodes) with free second-order variables $\boldsymbol{X}$. We can construct* UNFP$[\sigma]$*-formulas $\varphi_0^{\leftarrow}(\boldsymbol{X}^{\leftarrow})$, $\varphi_1^{\leftarrow}(x_1, \boldsymbol{X}^{\leftarrow})$, and $\varphi_k^{\leftarrow}(x_1, \ldots, x_k, \boldsymbol{X}^{\leftarrow})$ of width k such that for all σ-structures $\mathfrak{B}$, for all* UNFP-*safe valuations $\boldsymbol{J}$ of $\boldsymbol{X}$, and for all interface nodes (respectively, bag nodes) v in $\mathcal{S}^k(\mathfrak{B})$ with $|\mathrm{elem}(v)| = m$,*

$$\mathfrak{B}, \mathrm{elem}(v), \boldsymbol{J}^{\leftarrow} \models \varphi_m^{\leftarrow} \text{ iff } \mathcal{S}^k(\mathfrak{B}), v, \boldsymbol{J} \models \varphi.$$

Moreover, if $\mathrm{indices}(\varphi) = \{i_1, \ldots, i_n\}$*, then* $\mathrm{free}(\varphi_m^{\leftarrow}) = \{x_{i_1}, \ldots, x_{i_n}\}$*, and any strict subformula in $\varphi_m^{\leftarrow}$ that begins with an existential quantifier and is not directly below another existential quantifier has at most one free variable.*

Proof sketch: We proceed by induction on the structure of the UNFP-safe formula φ to define $\varphi_0^{\leftarrow}, \varphi_1^{\leftarrow}, \varphi_k^{\leftarrow}$ (assuming φ is UNFP-safe for interface nodes for the definition of $\varphi_0^{\leftarrow}$ and $\varphi_1^{\leftarrow}$, and UNFP-safe for bag nodes for the definition of $\varphi_k^{\leftarrow}$).

- Assume $\varphi = I$. Then $\varphi_0^{\leftarrow} := \top$, $\varphi_1^{\leftarrow} := \top$, and $\varphi_k^{\leftarrow} := \bot$.
- Assume $\varphi = D_j$. Then $\varphi_m^{\leftarrow} := \begin{cases} \top & \text{if } m = j, \\ \bot & \text{otherwise.} \end{cases}$
- Assume $\varphi = R_{i_1 \ldots i_n}$. Then $\varphi_0^{\leftarrow} := \bot$, $\varphi_1^{\leftarrow} := Rx_{i_1} \ldots x_{i_n}$, and $\varphi_k^{\leftarrow} := Rx_{i_1} \ldots x_{i_n}$.
- Assume $\varphi = X$. Then $\varphi_0^{\leftarrow} := X_0$, $\varphi_1^{\leftarrow} := X_1 x_1$, and $\varphi_k^{\leftarrow} := \bot$.
- The translation commutes with $\vee$, $\wedge$, and $\neg$. For the negation case, the definition of UNFP-safety and the inductive hypothesis ensures that the resulting formulas are in UNFP.
- Assume $\varphi = \langle\rho\rangle\chi$. Then
$$\begin{aligned} \varphi_0^{\leftarrow} &:= \exists y_1 \ldots y_k. \left(\chi_k^{\leftarrow}(y_1, \ldots, y_k)\right), \\ \varphi_1^{\leftarrow} &:= \exists y_1 \ldots y_k. \left(\chi_k^{\leftarrow}(y_1, \ldots, y_k)[x_1/y_{\rho(1)}]\right), \\ \varphi_k^{\leftarrow} &:= \begin{cases} \chi_1^{\leftarrow}(x_i) & \text{if } \mathrm{dom}(\rho) = \{i\} \\ \chi_0^{\leftarrow} & \text{if } \mathrm{dom}(\rho) = \emptyset \end{cases}. \end{aligned}$$
- Assume $\varphi = \mu Y.\chi(\boldsymbol{X}, Y)$. Then $\varphi_0^{\leftarrow}$ and $\varphi_1^{\leftarrow}$ are the first and second components of the vectorial fixpoint
$$\varphi^{\leftarrow} := \mathbf{lfp}\begin{pmatrix} Y_0 \\ Y_1, y_1 \end{pmatrix}.\begin{pmatrix} \chi_0^{\leftarrow}(\boldsymbol{X}^{\leftarrow}, Y_0, Y_1) \\ \chi_1^{\leftarrow}(y_1, \boldsymbol{X}^{\leftarrow}, Y_0, Y_1) \end{pmatrix}$$
and $\varphi_k^{\leftarrow} := \bot$.

We omit the proof of correctness for the base cases, and focus on parts of the last two cases. For notational simplicity, we assume there are no free second-order variables in φ.

Case $\varphi = \langle\rho\rangle\chi$: Assume v is a non-empty interface node in $\mathcal{S}^k(\mathfrak{B})$. We prove correctness of $\varphi_1^{\leftarrow}$.

First, assume that $\mathcal{S}^k(\mathfrak{B}), v \models \varphi$ where $\mathrm{elem}(v) = a_1$. Then there is some ρ-child w of v that satisfies χ. Let $\mathrm{elem}(w) = b_1 \ldots b_k$, where $a_1 = b_{\rho(1)}$. By the inductive hypothesis, this means that $\mathfrak{B}, \mathrm{elem}(w) \models \chi_k^{\leftarrow}$, so we have $\mathfrak{B}, \mathrm{elem}(v) \models \varphi_1^{\leftarrow}$ as desired.

Next, assume that $\mathfrak{B}, \mathrm{elem}(v) \models \varphi_1^{\leftarrow}$ where $\mathrm{elem}(v) = a_1$. Then there are elements $b_1, \ldots, b_k$ in $\mathfrak{B}$ such that $a_1 = b_{\rho(1)}$ and $\mathfrak{B} \models \chi_k^{\leftarrow}(b_1, \ldots, b_k)$. By definition of $\mathcal{S}^k(\mathfrak{B})$, there is a ρ-successor w of v based on exactly these elements (in fact, there are many successors based on these elements). Hence, $\mathfrak{B}, \mathrm{elem}(w) \models \chi_k^{\leftarrow}$, so by the inductive hypothesis, $\mathcal{S}^k(\mathfrak{B}), w \models \chi$. This is enough to conclude that $\mathcal{S}^k(\mathfrak{B}), v \models \varphi$.

By the inductive hypothesis, any strict subformula in $\chi_k^{\leftarrow}$ that begins with an existential quantifier and is not directly below another existential quantifier has at most one free variable. This implies that even if $\chi_k^{\leftarrow}$ begins with an existential quantifier, $\varphi_1^{\leftarrow}$ also satisfies this property. Moreover, the only free variable is x_1.

Case $\varphi = \mu Y.\chi(Y)$: By UNFP-safety, we must only consider the case when v is an interface node.

For ordinals β, we write χ^β and $(\chi^{\leftarrow})^\beta$ for the β-approximant of the outer fixpoint in φ and $\varphi^{\leftarrow}$. We first show the result for the fixpoint approximants. That is, for all interface nodes v with $|\mathrm{elem}(v)| = m$, $\mathfrak{B}, \mathrm{elem}(v) \models (\chi^{\leftarrow})^{\beta}_{m}$, iff $\mathcal{S}^k(\mathfrak{B}), v \models \chi^{\beta}$. We proceed by induction on the fixpoint approximant β.

For $\beta = 0$, the result follows by the outer inductive hypothesis applied to the formulas that result from substituting $\bot$ for Y in χ, and $\bot$ for Y_0 and $Y_1 y$ in $\chi^{\leftarrow}$.

Now assume $\beta > 0$ is a successor ordinal $\beta = \delta + 1$.

Assume $\mathcal{S}^k(\mathfrak{B}), v \models \chi^{\beta}$ for v an interface node with $|\mathrm{elem}(v)| = m$. Then $\mathcal{S}^k(\mathfrak{B}), v, J_\delta \models \chi$ where

$$J_\delta := \{w : w \text{ is an interface node and } \mathcal{S}^k(\mathfrak{B}), w \models \chi^{\delta}\}$$

is a valuation for X. Note that this is a UNFP-safe valuation: any subtrees rooted at an empty interface node in $\mathcal{S}^k(\mathfrak{B})$ are isomorphic to $\mathcal{S}^k(\mathfrak{B})$, which means that if any empty interface nodes are included, all empty interface nodes must be included. By the outer inductive hypothesis, this implies that $\mathfrak{B}, \mathrm{elem}(v), J^{\leftarrow}_{\delta} \models \chi^{\leftarrow}_{m}$.

However, by the inner inductive hypothesis,

$$J^{\leftarrow}_{\delta} = (J_0, \{\mathrm{elem}(w) : w \text{ is an interface node and } \mathfrak{B}, \mathrm{elem}(w) \models (\chi^{\leftarrow})^{\delta}_{|\mathrm{elem}(w)|}\}).$$

This means that $\mathfrak{B}, \mathrm{elem}(v), J^{\leftarrow}_{\delta} \models \chi^{\leftarrow}_{m}$ iff $\mathfrak{B}, \mathrm{elem}(v) \models (\chi^{\leftarrow})^{\beta}_{m}$, so $\mathfrak{B}, \mathrm{elem}(v) \models (\chi^{\leftarrow})^{\beta}_{m}$ as desired.

Next, assume $\mathfrak{B}, \mathrm{elem}(v) \models (\chi^{\leftarrow})^{\beta}_{m}$ for v and interface node with $|\mathrm{elem}(v)| = m$. Then $\mathfrak{B}, \mathrm{elem}(v), J_0, J_1 \models \chi^{\leftarrow}_{m}$ where J_0 is $\top$ (respectively, $\bot$) if $\mathfrak{B} \models (\chi^{\leftarrow})^{\delta}_{0}$ (respectively, $\mathfrak{B} \models \neg(\chi^{\leftarrow})^{\delta}_{0}$) and $J_1 := \{b : \mathfrak{B}, b \models (\chi^{\leftarrow})^{\delta}_{1}\}$.

Define

$$J_\delta := \{w : w \text{ is an interface node and } \mathcal{S}^k(\mathfrak{B}), w \models \chi^{\delta}\}$$

Note that every element in $\mathfrak{B}$ appears as the element in an interface node of $\mathcal{S}^k(\mathfrak{B})$, and every subtree of $\mathcal{S}^k(\mathfrak{B})$ rooted at any empty interface node is isomorphic to $\mathcal{S}^k(\mathfrak{B})$. Using these properties and the inner inductive hypothesis, it can be shown that $J^{\leftarrow}_{\delta} = (J_0, J_1)$. Hence, we can apply the outer inductive hypothesis to see that $\mathcal{S}^k(\mathfrak{B}), v, J_\delta \models \chi$, and this holds iff $\mathcal{S}^k(\mathfrak{B}), v \models \chi^{\beta}$.

The proof is similar when β is a limit ordinal.

The overall result for this case follows by appealing to the fact that the least fixpoint corresponds to some β-approximant, and using the Bekič principle to eliminate vectorial fixpoints.

Width: The constructed formulas use at most k free variables in any subformula. It can be checked that the constructed formulas have width k (that is, the conversion into normal form does not increase the maximum number of free variables in the subformulas; see Proposition 42). ■

Theorem 17 follows by applying Lemma 39 to get a UNFP-safe L_μ-formula φ, and then using Lemma 40, taking $\psi^{\leftarrow} := \varphi^{\leftarrow}_{0}$ as the desired $\mathrm{UNFP}^k[\sigma]$ sentence (since we want a sentence equivalent to φ at the root of $\mathcal{S}^k(\mathfrak{B})$, an empty interface node).

Appendix D
Improved Algorithm

A. *Proof of Theorem 20 (Forward mapping)*

Recall the statement of Theorem 20:

> Let φ be a sentence in $\mathrm{UNFP}^k[\sigma]$. We can construct a 2-way alternating μ-automaton $\mathcal{A}_\varphi$ such that for all consistent $\tilde{\sigma}_k$ trees $\mathcal{T}$, $\mathfrak{D}(\mathcal{T}) \models \varphi$ iff $\mathcal{T} \models L(\mathcal{A}_\varphi)$ and the size of $\mathcal{A}_\varphi$ is doubly exponential in $|\varphi|$, but the number of states of $\mathcal{A}_\varphi$ is at most singly exponential in $|\varphi|$, and the number of priorities is at most $|\varphi|$.

Before we give the construction, we want to define the relevant formulas that will need to be considered by the automaton in order to determine whether a $\mathrm{UNFP}^k[\sigma]$-formula holds in some structure. It turns out that we need more information than just the usual subformula closure, namely "specializations" of CQ-shaped subformulas. Consider a CQ-shaped $\phi(\boldsymbol{y})$ of the form $\exists \boldsymbol{x}. \bigwedge_i \psi_i(\boldsymbol{xy})$. A *specialization* of ϕ is a formula ϕ' obtained from ϕ by the following operations:

- select a subset $\boldsymbol{z}$ of $\boldsymbol{x}$ (call variables from $\boldsymbol{yz}$ the *inside variables* and variables from $\boldsymbol{x} \setminus \boldsymbol{z}$ the *outside variables*);
- select a partition $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ of the outside variables, with the property that for every ψ_j, either ψ_j has no outside variables or all of its outside variables are contained in some partition element $\boldsymbol{x}_i$;
- let χ_0 be the conjunction of the ψ_i using only inside variables, and let χ_j be the conjunction of the ψ_i using outside variables and satisfying $\mathrm{free}(\psi_i) \subseteq \boldsymbol{x}_j \boldsymbol{yz}$;
- set $\phi'(\boldsymbol{yz})$ to be $\chi_0(\boldsymbol{yz}) \wedge \bigwedge_{j \in \{1,\ldots,k\}} \exists \boldsymbol{x}_j . \chi_j(\boldsymbol{x}_j \boldsymbol{yz})$.

Roughly speaking, a specialization describes a way in which the original CQ-shaped subformula could be satisfied in a tree-like model. A specialization is *non-trivial* if either there are no outside variables (thus the specialization is only χ_0), or χ_0 is non-empty, or the partition of the outside variables is non-trivial ($j > 0$).

Lemma 41. *Let $\phi(\boldsymbol{y}) \in$ UNFP be a CQ-shaped formula $\exists \boldsymbol{x}. \bigwedge_i \psi_i(\boldsymbol{xy})$.*

- *Given a structure $\mathfrak{M}$ and a tree decomposition T of $\mathfrak{M}$, if there exists a node v with $\boldsymbol{b} \subseteq T(v)$ and $\mathfrak{M}, \boldsymbol{b} \models \phi(\boldsymbol{y})$, then there is a non-trivial specialization $\phi'(\boldsymbol{yz})$ of ϕ and a node w with a tuple $\boldsymbol{c}$ such that $\boldsymbol{bc} \in T(w)$ and $\mathfrak{M}, \boldsymbol{bc} \models \phi'(\boldsymbol{yz})$.*
- *For all structures $\mathfrak{M}$ and for all specializations $\phi'(\boldsymbol{yz})$ of ϕ, if $\mathfrak{M}, \boldsymbol{bc} \models \phi'(\boldsymbol{yz})$, then $\mathfrak{M}, \boldsymbol{b} \models \phi(\boldsymbol{y})$.*

Proof: We prove the first part of this lemma.

For elements $\boldsymbol{d}$ and a node w with a neighboring node w' in a tree decomposition T we say $\boldsymbol{d}$ *is contained in the direction of* w' if (i) w' is the parent of w and $\boldsymbol{d}$ appears in the tree resulting from removing from T the subtree rooted at w, or (ii) w' is a child of w and $\boldsymbol{d}$ appears in the subtree rooted at w'.

We can now prove the first part of the lemma. There must be some tuple $\boldsymbol{a} = a_1 \dots a_m$ of elements (corresponding to $\boldsymbol{x} = x_1 \dots x_m$) such that $\mathfrak{M}, \boldsymbol{ab} \models \bigwedge_i \psi_i(\boldsymbol{xy})$.

If there is a node w in the tree decomposition T with $\boldsymbol{ab} \subseteq T(w)$, then it is easy (take the specialization where all variables $\boldsymbol{x}$ are inside variables).

Otherwise, there is a node w in T with $\boldsymbol{b} \subseteq T(w)$ and such that $\boldsymbol{ab}$ is not contained in the direction of any neighbor of w (if not, then starting at the node v containing $\boldsymbol{b}$, we could eventually reach a node w' with $\boldsymbol{ab} \subseteq T(w')$, which we are assuming is not possible).

Let $\boldsymbol{z}$ be the tuple of variables from $\boldsymbol{x}$ corresponding to elements in $\boldsymbol{c} := T(w) \cap \boldsymbol{a}$ (i.e. $x_i \in \boldsymbol{z}$ iff $a_i \in T(w)$). Take $\boldsymbol{yz}$ to be the inside variables, corresponding to elements $\boldsymbol{bc}$ (the elements inside $T(w)$.

Let O be the nonempty set of elements from $\boldsymbol{a}$ that are not in $T(w)$ (i.e. the elements that correspond to outside variables). Let $O_{w'}$ be the set of elements from O that are contained in the direction of a neighbor w' of w. Because $o \in O$ do not appear in $T(w)$ and $|O| \leq k$, $\{O_{w'} : w' \text{ is a neighbor of } w\}$ is a partition of O into at most k partition elements. This induces a partition of the outside variables based on what variables belong together because the witnesses are contained in the same direction from w.

Taking the resulting non-trivial specialization ϕ', we have $\mathfrak{M}, \boldsymbol{bc} \models \phi'(\boldsymbol{yz})$. ■

We are now ready to define the closure set. For $\varphi \in \text{UNFP}^k[\sigma]$ in normal form, let $\text{cl}_+(\varphi)$ be the smallest set C of formulas containing φ and satisfying the following closure conditions:

- if $\neg\psi \in C$, then $\psi \in C$;
- if $\bigvee_i \psi_i \in C$ or $\bigwedge_i \psi_i \in C$, then $\psi_i \in C$ for all i;
- if $\exists \boldsymbol{x}.\psi(\boldsymbol{xy}) \in C$, then for every specialization $\chi_0(\boldsymbol{xy}) \wedge \bigwedge_j \exists \boldsymbol{z}_j.\chi_j(\boldsymbol{xyz}_j)$ of $\exists \boldsymbol{x}.\psi(\boldsymbol{xy})$, $\chi_0 \in C$ and $\exists \boldsymbol{z}_j.\chi_j \in C$;
- if $[\mathbf{lfp}\, Y, y.S](x) \in C$, then $\psi \in C$ for each ψ in the system of fixpoint formulas in S.

Let $\text{cl}_+(\varphi, U_k)$ be the set of formulas obtained by substituting names from $U_k = \{1, \dots, k\}$ for all of the free variables in the formulas in $\text{cl}_+(\varphi)$.

It is straightforward to check that the size of $\text{cl}_+(\varphi, U_k)$ is exponential in the size of φ. We are interested in the size of the closure set, since we will see that this parameter controls the size of the automata constructed. It turns out that the size of the closure set is still exponential in the size of the input formula, even when we first have to convert to normal form.

Proposition 42. *Let ψ be a formula in* UNFP *with $k = |\psi|$. We can construct an equivalent formula $\varphi \in$* UNFP^k *in normal form such that*

- $|\varphi| \leq 2^{f(k)}$,
- $|\text{cl}_+(\varphi, U_k)| \leq 2^{f(k)}$,

where f is a polynomial function independent of ψ.

Proof: Starting from the innermost formulas, we apply the following transformation rules to convert an arbitrary formula into normal form:

$$\begin{array}{rrcl} & \exists x.(\phi \vee \psi) & \to & (\exists x.\phi) \vee (\exists x.\psi) \\ & \phi \wedge (\psi \vee \chi) & \to & (\phi \wedge \psi) \vee (\phi \wedge \chi) \\ \text{if } |\text{free}(\exists x.\phi)| > 1: & (\exists x.\phi) \wedge \psi & \to & \exists x'.(\phi[x'/x] \wedge \psi) \end{array}$$

where $\phi[x'/x]$ is the result of substituting some fresh x' for x in ϕ. It is straightforward to check that this results in a formula in normal form. The new formula φ is of size at most $M := 2^{f(k)}$ for $k = |\psi|$ and f some polynomial function, but has CQ-shaped subformulas with at most k conjuncts. Moreover, the number of free variable names needed in any subformula is at most k, so $\varphi \in \text{UNFP}^k$.

There are at most M subformulas of φ, all of which appear in $\text{cl}_+(\varphi)$. However, each CQ-shaped subformula ψ can contribute additional formulas to the closure set due to specializations. Fix some CQ-shaped subformula χ of φ. There are at most 2^k additional CQ-shaped subformulas χ' obtained by choosing some subset of the conjuncts in χ. In each χ', there are at most 2^k choices of the inside variables, and k^k ways to partition the outside variables resulting in specializations χ'' of χ'. Notice that these specializations χ'' only have CQ-shaped subformulas resulting from taking some subset of the conjuncts in the original CQ-shaped formula χ, so these formulas have already been accounted for. Hence, χ contributes at most $2^k 2^k k^k$ new CQ-shaped subformulas to $\text{cl}_+(\varphi)$. Overall, this means $|\text{cl}_+(\varphi)| \leq M \cdot 2^k 2^k k^k$. Finally, in $\text{cl}_+(\varphi, U_k)$, each of the at most k free variables in formulas from $\text{cl}_+(\varphi)$ is mapped to a name in $U_k = \{1, \dots, k\}$, so $|\text{cl}_+(\varphi, U_k)| \leq k^k \cdot \text{cl}_+(\varphi)$, which is exponential in k as claimed. ■

Hence, we can safely assume that $\varphi \in \text{UNFP}^k[\sigma]$ is in normal form. We will also assume that no fixpoint variable is bound by more than one fixpoint operator, so each fixpoint variable X identifies a unique subformula $D_\phi(X)$ of ϕ in which X only appears positively. If X occurs positively (respectively, negatively) in ϕ then it is a $\mathbf{lfp}$-*variable* (respectively, $\mathbf{gfp}$-*variable*). We say X depends on Y if Y occurs free in $D_\phi(X)$. The *dependency order* $\sqsubset_\phi$ is the transitive closure of this relation. The *alternation level* $\text{al}_\phi(X)$ of X is the maximum number of alternations between $\mathbf{lfp}$ and $\mathbf{gfp}$ variables on the $\sqsubset_\phi$-paths descending from X.

Fix such a $\varphi \in \text{UNFP}^k[\sigma]$ in normal form. We view φ as a query about the decoding of the input tree $\mathcal{T}$. The idea behind the construction of the automaton for Theorem 20 is to allow Eve to guess an annotation of $\mathcal{T}$ with information about which unary subqueries of φ hold, and then run an automaton that checks φ with the help of these annotations. In order to prevent Eve from cheating with her guesses about the subqueries, Adam is allowed to launch inductively-defined automata in order to check Eve's claims about the subqueries.

Testing $\mathfrak{D}(\mathcal{T}) \models [\mathbf{lfp}\, Y, y.\psi(y, Y)](a)$ can be viewed as a game between Adam and Eve which starts with $y = a$ and proceeds as follows:

- Eve chooses some valuation for Y such that $\mathfrak{D}(\mathcal{T}) \models \psi(y, Y)$ (she loses if this is not possible), then

- Adam chooses some new $y \in Y$ (he loses if this is not possible), and then the game proceeds to the next turn.

If the game never terminates then Adam is declared the winner. We can implement this game as an automaton running on $\mathcal{T}$, where Eve guesses an annotation of $\mathcal{T}$ with the valuation of Y in the current round, and then simulates the inductively-defined automaton checking $\psi(y, Y)$. Adam can challenge any b in the set Y chosen by Eve by launching another copy of the automaton checking ψ starting from the node carrying b. Correctness is enforced by using the parity condition: an odd priority is used if Adam challenges a **lfp** fixpoint.

We prove two technical lemmas leading to Theorem 20.

As a first step, it is straightforward to construct an automaton that checks a UCQ-shaped query when running on trees annotated with subqueries with at most one free variable (we call these *unary subqueries*). The idea is that the automaton guesses a specialization of each CQ, and uses the annotations to check that this specialization is actually realized.

Lemma 43. *Let $\psi(a_1, \dots, a_k, \boldsymbol{X}, \boldsymbol{S})$ be a UCQ-shaped formula from $\mathrm{cl}_+(\varphi, U_k)$ with every unary subquery χ replaced by a new predicate S_χ (we write $\boldsymbol{S}$ for this set of new predicates). We can construct a 2-way alternating μ-automaton $\mathcal{C}_\psi$ of size exponential in $|\mathrm{cl}_+(\psi, U_k)|$ such that for all consistent $\tilde{\sigma}_k$-trees $(\mathcal{T}, \boldsymbol{Z}^{\rightarrow}, \boldsymbol{S}^{\rightarrow})$ and for all nodes $v \in \mathrm{dom}(\mathcal{T})$,*

$$\mathfrak{D}(\mathcal{T}), [v, a_1], \dots, [v, a_k], \boldsymbol{Z}, \boldsymbol{S} \models \psi \textit{ iff } (\mathcal{T}, \boldsymbol{Z}^{\rightarrow}, \boldsymbol{S}^{\rightarrow}) \in L_v(\mathcal{C}_\psi).$$

In $\mathcal{C}_\psi$, the number of states is polynomial in $|\mathrm{cl}_+(\psi, U_k)|$, and only two priorities are used.

Proof: Assume we have constructed automata for each CQ in ψ, of size polynomial in $|\mathrm{cl}_+(\psi, U_k)|$. Then it is straightforward to create an automaton for ψ of the desired size by taking the disjoint union of at most $|\mathrm{cl}_+(\psi, U_k)|$-many automata, one for each CQ in ψ. Hence, it suffices to show that for each CQ-shaped formula ψ, we can construct an automaton of size polynomial in $|\mathrm{cl}_+(\psi, U_k)|$.

Fix such a ψ. The states of $\mathcal{C}_\psi$ are CQ-shaped formulas in $\mathrm{cl}_+(\psi, U_k)$.

Assume Eve is in state $\psi(\boldsymbol{a})$ (a CQ-shaped formula) at position $w \in \mathrm{dom}(\mathcal{T})$. Eve immediately loses if w does not contain $\boldsymbol{a}$.

If ψ does not begin with existential quantifiers, then the automaton checks locally if this part of the formula is satisfied, based on the facts from the tree encoding, and the annotations provided by $\boldsymbol{Z}^{\rightarrow}$ and $\boldsymbol{S}^{\rightarrow}$.

Otherwise, the automaton outputs priority 1 and proceeds as follows:

- Eve guesses a specialization $\chi_0(\boldsymbol{ay}) \wedge \bigwedge_{j \in \{1,\dots,l\}} \exists \boldsymbol{z}_j . \chi_j(\boldsymbol{ayz}_j)$ of ψ and some names $\boldsymbol{b}$ appearing in w.
- Adam chooses some $j \in \{0, \dots, l\}$.
- If $j = 0$, then the automaton remains in the same position and moves to state $\chi_0(\boldsymbol{ab})$. Otherwise, Eve selects a neighboring node w' (where ρ describes the relationship between the names in w and w'), and the automaton moves to state $\exists \boldsymbol{z}_j . \chi_j(\rho(\boldsymbol{ab})\boldsymbol{z}_j)$ in w'.

Correctness of the construction follows from Lemma 41.

This concludes the proof of Lemma 43. ■

Using this lemma, we define the 2-way alternating μ-automaton that checks φ when running on tree encodings $(\mathcal{T}, \boldsymbol{Z}^{\rightarrow})$. In order to correctly handle negation, we need to consider the *polarity* of subformulas $\psi \in \mathrm{cl}_+(\varphi)$, i.e. whether ψ occurred positively (in the scope of an even number of negations) or negatively (in the scope of an odd number of negations) in φ. For polarity $p \in \{+, -\}$, we write $p\psi$ to denote ψ (respectively, $\neg\psi$) if $p = +$ (respectively, $p = -$). The definition of the automaton for ψ depends on the polarity of ψ.

Lemma 44. *Let $\psi(a_1, \dots, a_k, \boldsymbol{Z}) \in \mathrm{cl}_+(\varphi, U_k)$. If ψ appears with polarity p in φ, then there exists a 2-way alternating μ-automaton $\mathcal{B}^p_\psi$ of size exponential in $|\mathrm{cl}_+(\psi, U_k)|$ such that for all consistent $\tilde{\sigma}_k$-trees $(\mathcal{T}, \boldsymbol{Z}^{\rightarrow})$ and for all nodes $v \in \mathrm{dom}(\mathcal{T})$,*

$$\mathfrak{D}(\mathcal{T}), [v, a_1], \dots, [v, a_k], \boldsymbol{Z} \models \psi \;\textit{ iff }\; \mathcal{T}, \boldsymbol{Z}^{\rightarrow} \in L_v(\mathcal{B}^p_\psi).$$

The number of states and priorities in $\mathcal{B}^p_\psi$ is polynomial in $|\mathrm{cl}_+(\psi, U_k)|$.

Proof: The proof uses ideas familiar from work on the μ-calculus and GFP (see, e.g., [27]).

We first describe the construction of $\mathcal{B}^p_\psi$, which proceeds by induction on the structure of ψ.

- Assume ψ is a UCQ-shaped subformula. If $p = +$ (respectively, $p = -$), then let $\mathcal{B}^p_\psi$ be the automaton where Eve (respectively, Adam) chooses a valuation $\boldsymbol{S}^{\rightarrow}$, $\mathcal{C}_\psi$ (respectively, the dual of $\mathcal{C}_\psi$) from Lemma 43 is simulated, and Adam (respectively, Eve) can choose to launch $\mathcal{B}^p_{\psi''}$ from a node w if $S_{\psi''}(w)$.
- Assume ψ is a unary subquery of the form $\neg\psi'$. If $p = +$ (respectively, $p = -$), then simulate $\mathcal{B}^-_{\psi'}$ respectively, $\mathcal{B}^+_{\psi'}$.
- Assume ψ is a fixpoint subformula $[\mathbf{lfp}\, X, x.\psi'(x, X, \boldsymbol{Z})](a)$ where X has alternation level j relative to the other fixpoint variables in φ. If $p = +$ (respectively, $p = -$), then let $\mathcal{B}^p_\psi$ be the automaton where Eve (respectively, Adam) chooses valuation $X^{\rightarrow}$, $\mathcal{B}^p_{\psi'(a,X,\boldsymbol{Z})}$ is simulated, and Adam (respectively, Eve) can choose to launch $\mathcal{B}^p_{\psi'(b,X,\boldsymbol{Z})}$ from w if $X_b(w)$. In case $X_b(w)$ is challenged, the priority output is $2j - 1$ (respectively, $2j$).

We now give the proof of correctness for one of the interesting cases when ψ is a fixpoint subformula $[\mathbf{lfp}\, X, x.\eta](a)$ and $p = +$. We assume that there are no free second-order variables in ψ, since these do not affect the arguments below.

Let $\mathcal{G} := \mathcal{G}(\mathcal{B}^p_{\psi(a)}, \mathcal{T}, v)$. In the argument below, we also consider *approximant games* $\mathcal{G}(\mathcal{B}^p_{\eta(a)}, \mathcal{T}, X^{\rightarrow}, v)$, which are intermediate games based on evaluating the body of the fixpoint with some valuation $X^{\rightarrow}$. For ordinals α, let $(X^\alpha)^{\rightarrow}$ be the valuation such that $X_b(w)$ holds iff $\mathfrak{D}(\mathcal{T}), [w, b] \models \eta^\alpha$.

Assume $\mathfrak{D}(\mathcal{T}), [v, a] \models \psi$. We must define a winning strategy ζ for Eve in $\mathcal{G}$. There is some least ordinal α such that $\mathfrak{D}(\mathcal{T}) \models \eta^{\alpha}([v, a])$. Moreover, α is a successor ordinal and $\mathfrak{D}(\mathcal{T}) \models \eta([v, a])[\eta^{\alpha-1}(y)/X(y)]$. By the inductive hypothesis, this means there is a winning strategy $\zeta^{\alpha-1}$ for Eve in the approximant game $\mathcal{G}^{\alpha-1} := \mathcal{G}(\mathcal{B}^{p}_{\eta(a)}, \mathcal{T}, (X^{\alpha-1})^{\rightarrow}, v)$. Eve begins by selecting the valuation $(X^{\alpha-1})^{\rightarrow}$, and using the strategy $\zeta^{\alpha-1}$ from $\mathcal{G}^{\alpha-1}$. If Adam never challenges an annotation, then the play is winning for Eve by assumption on $\zeta^{\alpha-1}$. If Adam challenges some $X_b(w)$ in $(X^{\alpha-1})^{\rightarrow}$, then there must be some least ordinal $\beta \leq \alpha - 1$ such that $\mathfrak{D}(\mathcal{T}) \models \eta^{\beta}([w, b])$, by definition of $(X^{\alpha-1})^{\rightarrow}$. Hence, using similar reasoning as above, Eve can switch to selecting the valuation $(X^{\beta-1})^{\rightarrow}$ and playing the inductively defined winning strategy $\zeta^{\beta-1}$ in $\mathcal{G}^{\beta-1} := \mathcal{G}(\mathcal{B}^{p}_{\eta(b)}, \mathcal{T}, (X^{\beta-1})^{\rightarrow}, w)$. Eve continues updating the approximant and her strategy in the approximant game like this after each challenge by Adam. On any play, there can be only a finite number of challenges by Adam (otherwise, $\alpha > \beta > \ldots$ would be an infinite descending chain, contradicting the well-foundedness of ordinals). This means that any play in the constructed strategy will eventually stabilize in some particular approximant game, where Eve can play her inductively-defined winning strategy. Hence, Eve has a winning strategy.

Now assume $\mathfrak{D}(\mathcal{T}), [v, a] \not\models \eta$. We must define a winning strategy for Adam in $\mathcal{G}$. Consider the valuation $X^{\rightarrow}$ such that $X_b(w)$ holds iff $\mathfrak{D}(\mathcal{T}), [w, b] \models \psi$. Let ζ' be the inductively defined winning strategy for Adam in the approximant game $\mathcal{G}' := \mathcal{G}(\mathcal{B}^{p}_{\eta(a)}, \mathcal{T}, X^{\rightarrow}, v)$, based on this valuation. We construct Adam's strategy ζ in $\mathcal{G}$ so that he starts by playing from ζ'. If Eve deviates from the valuation for X defined above by guessing some tuple $X_b(w)$ such that $\mathfrak{D}(\mathcal{T}), [w, b] \not\models \psi$ then Adam challenges this tuple, and starts using a new inductively defined winning strategy in the approximant game $\mathcal{G}(\mathcal{B}^{p}_{\eta(b)}, \mathcal{T}, X^{\rightarrow}, w)$ (but with the same valuation). If Eve eventually stabilizes in a particular approximant game, then Adam wins using his inductively-defined strategy. If Eve deviates infinitely-many times, then the play will be winning for Adam, since the maximum priority occurring infinitely often will be odd (corresponding to the priority output at each challenge). Hence, Adam has a winning strategy in $\mathcal{G}$ as desired.

This concludes Lemma 44. ■

The desired 2-way alternating μ-automaton $\mathcal{A}_{\varphi}$ in Theorem 20 is the result of running the automaton $\mathcal{B}^{+}_{\varphi}$ starting from the root of $\mathcal{T}$. This concludes the proof of Theorem 20.

B. Proof of Theorem 21 (Simplification)

Recall the statement:

> Let $\mathcal{A}$ be a 2-way alternating μ-automaton on $\tilde{\sigma}_k$ trees. We can construct a well-structured $\mathrm{L}_{\mu}[\tilde{\sigma}_k]$-formula ψ such that for all consistent $\tilde{\sigma}_k$ trees $\mathcal{T}$, $\mathfrak{D}(\mathcal{T}) \models \psi$ iff $\mathcal{T} \in L(\mathcal{A})$. The size of ψ is at most $|\mathcal{A}|^{f(m)}$ where m is the number of states and priorities of $\mathcal{A}$ and f is a polynomial function independent of $\mathcal{A}$.

As mentioned in the body, we actually convert to a μ-automaton, and then use Proposition 32 to convert the μ-automaton to a well-structured L_{μ}-formula. Hence, we prove the following theorem:

Theorem 45. *Let $\mathcal{A}$ be a 2-way alternating μ-automaton. There is a μ-automaton $\mathcal{M}$ such that $L(\mathcal{A}) = L(\mathcal{M})$. The size of $\mathcal{M}$ is at most $|\mathcal{A}|^{f(m)}$, where m is the number of states and priorities of $\mathcal{A}$ and f is a polynomial function independent of $\mathcal{A}$. The number of states in $\mathcal{M}$ is exponential in the number of states and priorities of $\mathcal{A}$, and the number of priorities in $\mathcal{M}$ is linear in the number of states of $\mathcal{A}$.*

This is similar to the result in [22]. In [22], however, the result is about trees of some fixed finite branching k: a 2-way alternating μ-automaton $\mathcal{A}$ using directions up (-1), stay (0), and down in some direction $1, \ldots, k$, is converted to an equivalent 1-way nondeterministic automaton $\mathcal{M}$ using only directions $1, \ldots, k$.

We are dealing with trees with arbitrary branching, so we cannot refer to specific children; instead, we start with a 2-way alternating μ-automaton $\mathcal{A}$ using directions up $(\uparrow)$, stay (0), and down $(\downarrow)$, and convert to an equivalent (1-way) μ-automaton.

As in [22], the idea is that $\mathcal{M}$ on input $\mathcal{T}$ guesses an annotation of $\mathcal{T}$ with a (positional) strategy for Eve in the acceptance game of $\mathcal{M}$ on $\mathcal{T}$, and then checks whether this strategy is actually winning. In order to check this in a one-way fashion, $\mathcal{M}$ also guesses additional information about loops that are possible when using this strategy.

A *strategy annotation* γ of $\mathcal{T}$ is a labelling of the tree $\mathcal{T}$ with information about a positional strategy for Eve in the acceptance game of $\mathcal{A}$ on $\mathcal{T}$. Formally, γ is a mapping from $\mathrm{dom}(\mathcal{T})$ to $\mathcal{P}(Q \times \mathrm{Dir} \times \Sigma_a \times Q)$ satisfying the following properties:

(i) if q_0 is the initial state of $\mathcal{A}$, then there must be some $(q_0, d, a, q') \in \gamma(\epsilon)$ (the initial state must be present at the root ϵ);
(ii) if $(q, d, a, q') \in \gamma(v)$ and $q \in Q_E$, then $(d, a, q') \in \delta(q, \mathcal{T}(v))$;
(iii) if $(q, d, a, q') \in \gamma(v)$ and $q \in Q_A$, then $(d, a, q') \in \delta(q, \mathcal{T}(v))$ and $\delta(q, \mathcal{T}(v)) \in \gamma(v)$;
(iv) if $(q, d, a, q') \in \gamma(v)$ and $q \in Q_E$, then there is some a-neighbor w of v in direction d with some $(q', d', a', q'') \in \gamma(w)$;
(v) if $(q, d, a, q') \in \gamma(v)$ and $q \in Q_A$, then for all a'-neighbors w of v in direction d', there is some $(q', d', a', q'') \in \gamma(w)$.

Every play in $\mathcal{A}$ using the strategy γ can be decomposed into a downwards path together with loops that come back to this path, possibly in a different state. Given a tree $\mathcal{T}$ and a strategy annotation γ of $\mathcal{T}$, a *loop annotation* η summarizes the loops that are possible using γ. Loops at v are summarized by tuples (q, p, q'), which indicate that (using the strategy γ) there is a finite path leading from q at v to q' at v such that the

the maximum priority encountered on this path is p, and every node on this path is in the subtree rooted at v. Formally, a loop annotation η is a mapping from $\mathrm{dom}(\mathcal{T})$ to to $\mathcal{P}(Q \times \mathrm{Pri} \times Q)$ that satisfies some natural closure properties for all nodes $v \in \mathrm{dom}(\mathcal{T})$:

(i) if $(q, 0, a, q') \in \gamma(v)$, then $(q, \Omega(q'), q') \in \eta(v)$;
(ii) if $(q, p, q') \in \eta(v)$ and $(q', p', q'') \in \eta(v)$, then $(q, \max\{p, p'\}, q'') \in \eta(v)$;
(iii) if $(q, \downarrow, a, q') \in \gamma(v)$ and $(q', \uparrow, a, q'') \in \gamma(w)$ for an a-successor w of v, then $(q, \max\{\Omega(q'), \Omega(q'')\}, q'') \in \eta(v)$;
(iv) if $(q, \downarrow, a, q') \in \gamma(v)$ and $(q', p, q'') \in \eta(w)$ for an a-successor w of v and $(q'', \uparrow, a, q''') \in \gamma(w)$, then $(q, \max\{\Omega(q'), p, \Omega(q''')\}, q''') \in \eta(v)$.

It is straightforward to construct a μ-automaton $\mathcal{C}$ that reads an annotated tree and checks whether these annotations are strategy annotations and loop annotations. The state of the automaton stores the previous annotation labels in order to help perform these checks, so the size of the automaton is exponential in the number of states and number of priorities of $\mathcal{A}$.

Lemma 46. *There is a μ-automaton $\mathcal{C}$ that reads an annotated tree and checks that the annotations actually correspond to a strategy and loop annotation. The size of $\mathcal{C}$ is at most $|\mathcal{A}|^{f(m)}$ where m is the number of states and priorities of $\mathcal{A}$ and f is a polynomial function independent of $\mathcal{A}$.*

Now consider a branch of a tree with these strategy and loop annotations. This branch induces a word consisting of the annotations along this branch. Given such a word w, a *downward path* based w on is an infinite sequence of states that describe a possible play on this branch based on the strategy annotations and loop annotations — but not making use of any upward moves. Note that it is possible that there are no downward paths consistent with w. We say a strategy and loop annotation of some tree $\mathcal{T}$ is winning if every downward path satisfies the parity condition.

Proposition 47. *A 2-way alternating μ-automaton accepts an input tree $\mathcal{T}$ iff there is a strategy annotation γ of $\mathcal{T}$ and a loop annotation of γ that are winning.*

The proof is similar to [22, Propositions 5 and 6] so we omit it here; it relies on the positional determinacy of parity games [29].

We now construct a μ-automaton $\mathcal{N}$ that reads a tree with strategy and loop annotations, and checks (in a 1-way fashion) whether the strategy is actually winning for Eve.

Lemma 48. *There is an action-deterministic μ-automaton $\mathcal{N}$ that reads a tree with strategy and loop annotations, and checks whether the annotations are winning. The size of $\mathcal{N}$ is at most $|\mathcal{A}|^{f(m)}$ where m is the number of states and priorities of $\mathcal{A}$ and f is a polynomial function independent of $\mathcal{A}$.*

Proof sketch: This is constructed by first observing that there is a nondeterministic parity automaton—on words—that accepts if a sequence of annotation labels describe a downward path that is losing for Eve. This automaton guesses such a downward path and some point on this path after which the highest priority occurring infinitely often is odd. The size of the state set is linear in the size of $\mathcal{A}$, and it only uses two priorities (i.e. it is a Büchi automaton). Hence, there is a deterministic parity automaton $\mathcal{D}$ that accepts the complement of this language (see, e.g., [26]). That is, $\mathcal{D}$ accepts sequences w of annotations such that all downward paths based on w are winning for Eve. The number of states of $\mathcal{D}$ is exponential in the number of states of $\mathcal{A}$, and the number of priorities is linear in the number of states of $\mathcal{A}$. We take $\mathcal{N}$ to be the action-deterministic μ-automaton that simulates $\mathcal{D}$ on every branch of the tree. The state set and priorities of $\mathcal{N}$ are the same as in $\mathcal{D}$. ■

We are now ready to construct $\mathcal{M}$, the μ-automaton equivalent to $\mathcal{A}$ for Theorem 45. Let $\mathcal{M}$ be the μ-automaton obtained by taking the intersection of $\mathcal{C}$ and $\mathcal{N}$, and then projecting the strategy and loop annotations (using the closure properties in Proposition 31). The size of $\mathcal{M}$ is at most $|\mathcal{A}|^{f(m)}$ where m is the number of states and priorities of $\mathcal{A}$, and f is a polynomial function independent of $\mathcal{M}$. The correctness of this construction relies on bisimulation-invariance of tree languages for 2-way alternating μ-automata.

If $\mathcal{A}$ accepts some tree $\mathcal{T}$, then by Proposition 47 there is some strategy annotation and loop annotation describing the winning strategy. Since it is winning, every downward path satisfies the parity condition of $\mathcal{A}$. Hence, the μ-automaton can use this strategy and loop annotation to guide the choices in $\mathcal{M}$.

If $\mathcal{M}$ accepts some tree $\mathcal{T}$, then there is some tree $\mathcal{T}'$ that is bisimilar to $\mathcal{T}$ (with respect to the original alphabet, not the alphabet with annotations) that is accepted by both $\mathcal{N}$ and $\mathcal{C}$. This means there is some strategy and loop annotation for $\mathcal{T}'$, which by Proposition 47 means that $\mathcal{T}'$ (with the annotations omitted) is accepted by $\mathcal{A}$. But $L(\mathcal{A})$ is closed under bisimulation by Proposition 29, so $\mathcal{T}$ is also accepted by $\mathcal{A}$.

This concludes the proof of Theorem 45.

Appendix E
Formulas

A. Failure of strong form of interpolation with respect to constants

Recall that in Section IV (in the discussion about the extension of uniform interpolation to signatures with constants) it is stated that UNFP formulas do not have the stronger form of interpolation where the constants are restricted to the common signature. We justify this here.

Suppose for the sake of contradiction that we had this stronger form of interpolation for UNFP formulas with one free variable.

Consider the following UNF formulas with constants c and d:

$$\begin{aligned} \psi_{\mathrm{L}} &:= (x = c) \wedge \exists y. \neg(y = c) \\ \psi_{\mathrm{R}} &:= \neg(x = d) \vee \exists y. \neg(y = d) \end{aligned}$$

Let $\theta \in$ UNFP be the uniform interpolant for ψ_{L}, restricting to the subsignature with no constants. Since θ cannot use any constants and ψ_{L} entails ψ_{R}, θ must be equivalent to $\neg(x = y)$. But this is not expressible in UNFP, so we have reached a contradiction.

Appendix F
Extension for Equality

Our aim is to extend the uniform interpolation algorithm to allow sentences with equality. We accomplish this by showing that we can convert to formulas and tree encodings with a very limited use of equality that ensures that we can use the algorithm described before, simply treating equality like any other relation.

A. Equality normalization for GSO

Let us say that a formula ϕ in GSO using constants $\boldsymbol{e}$ and free variables $\boldsymbol{x}$ is *equality-normalized* if

(i) every subformula beginning with a first-order existential quantifier and using free variables $\boldsymbol{z}$ is of the form $\exists y.(\bigwedge_{t \in (\boldsymbol{z} \cup \boldsymbol{e})} \neg(y = t) \wedge \chi(y, \boldsymbol{z}))$ and

(ii) every occurrence of equality in ϕ is either an equality over its free variables and constants, or comes from (i).

Let $\equiv$ be any equivalence relation over $\boldsymbol{x} \cup \boldsymbol{e}$. We denote by $\xi_{\equiv}$ the formula

$$\bigwedge_{s \equiv t} (s = t) \wedge \bigwedge_{s \not\equiv t} \neg(s = t),$$

that is, the conjunction of all equalities and inequalities corresponding to $\equiv$.

The following lemma describes a way to bring a formula into equality normal form.

Lemma 49. *Let $\phi(\boldsymbol{x})$ be a* GSO *formula. We can construct an equality-normalized ϕ' equivalent to ϕ.*

Proof: Let $\boldsymbol{x}$ be the variables used in ϕ and $\boldsymbol{e}$ be the constants used in it.

We first construct an equality-normalized formula $\phi'_{\equiv}$ that is equivalent to ϕ assuming that $\xi_{\equiv}$ holds. For each $\equiv$-equivalence class, we fix an arbitrary representative — a constant whenever possible. We replace all occurrences of each variable and constant in $\boldsymbol{x} \cup \boldsymbol{e}$ by the representative of its equivalence class. Next, we replace subformulas of the form $s = t$, with $s, t \in \boldsymbol{x} \cup \boldsymbol{e}$, by $\top$ if $s = t$ and by $\bot$ otherwise.

Now consider any subformula of the form

$$\psi(\boldsymbol{z}) = \exists y. \chi(y, \boldsymbol{z})$$

We will essentially do a case distinction of the possible values that y may take. More precisely, we replace $\psi(\boldsymbol{z})$ by the disjunction, for each map $f : y \to (y \cup \boldsymbol{z} \cup \boldsymbol{e})$, of the formula ψ_f obtained from ψ by (i) replacing y by $f(y)$, (ii) replacing $y = y$ by $\top$, (iii) replacing $y = t$ or $t = y$ for $t \in \boldsymbol{z} \cup \boldsymbol{e}$ by $\top$ if $f(y) = t$ and by $\bot$ otherwise (and dropping the quantifier if the quantified variable no longer occur in the formula). Finally, for any ψ_f where y is still quantified, we conjoin it with $\bigwedge_{t \in (\boldsymbol{z} \cup \boldsymbol{e})} \neg(y = t)$. This clearly preserves the semantics of the formula over structures satisfying $\xi_{\equiv}$. We obtain the desired equality-normalized $\phi'_{\equiv}$ by performing this rewriting in a bottom-up fashion starting with the innermost quantifier.

Then $\phi' := \bigvee_{\equiv} \xi_{\equiv} \wedge \phi_{\equiv}$, where $\equiv$ ranges over equivalence relations over $\boldsymbol{x} \cup \boldsymbol{e}$. ∎

B. *Equality normalization for* UNFP

Let us say that a formula $\phi(x)$ in UNFP using constants $\boldsymbol{e}$ and at most one free variable x is *equality-normalized* if

(i) every occurrence of $R\,\boldsymbol{t}$ or $X\,\boldsymbol{t}$ in ϕ appears in conjunction with
$$\bigwedge_{t\in\boldsymbol{t},e\in\boldsymbol{e},t\neq e}\neg(t=e),$$
and

(ii) every occurrence of equality in ϕ is either over $\{x\}\cup\boldsymbol{e}$, or comes from (i).

In particular, this means that there are no assertions like $y=z$ for y and z distinct variables.

Now let $\equiv$ be any equivalence relation over $\{x\}\cup\boldsymbol{e}$. We denote by $\xi_{\equiv}$ the formula
$$\bigwedge_{s\equiv t}(s=t)\wedge\bigwedge_{s\not\equiv t}\neg(s=t),$$
that is, the conjunction of all equalities and inequalities corresponding to $\equiv$ (note that this is in UNF, since x is the only variable).

The following lemma describes a way to eliminate equalities to bring a formula into equality normal form, similar to the result for GSO. The blow-up resulting from this normalization is exponential in the *width*.

Lemma 50. *Let $\phi(x)$ be a* UNFP *formula in normal form containing constants $\boldsymbol{e}$. We can construct a DAG-representation of an equivalent equality-normalized $\phi'(x)$ in normal form such that*

- *the size of the DAG-representation of ϕ' is at most $p(|\phi|^k)$, where k is the width of ϕ, and p is a polynomial function independent of ϕ;*
- $\mathrm{width}(\phi')\leq\mathrm{width}(\phi)$.

Proof: We proceed in a similar fashion as Lemma 49. Let x be the free variable in ϕ and $\boldsymbol{e}$ be the constants used in it.

We first construct an equality-normalized formula $\phi'_{\equiv}$ that is equivalent to ϕ assuming that $\xi_{\equiv}$ holds. For each $\equiv$-equivalence class, we fix an arbitrary representative — a constant whenever possible. We replace all occurrences of each variable and constant in $\{x\}\cup\boldsymbol{e}$ by the representative of its equivalence class. Next, we replace subformulas of the form $s=t$, with $s,t\in\{x\}\cup\boldsymbol{e}$, by $\top$ if $s=t$ and by $\bot$ otherwise. Finally, we conjoin every relational atom containing distinct $s,t\in\{x\}\cup\boldsymbol{e}$ with $\neg(s=t)$.

We take advantage of the fact that we are in normal form, and now consider CQ-shaped subformulas of the form
$$\psi(\boldsymbol{z})=\exists\boldsymbol{y}.\chi(\boldsymbol{y},\boldsymbol{z}).$$
We now do a case distinction, for each quantified variable $y_i\in\boldsymbol{y}$, of the possible values that y_i may take. More precisely, we replace $\psi(\boldsymbol{z})$ by the disjunction, for each map $f:\boldsymbol{y}\to(\boldsymbol{y}\cup\boldsymbol{z}\cup\boldsymbol{e})$, of the formula ψ_f obtained from ψ by (i) replacing each $y_i\in\boldsymbol{y}$ by $f(y_i)$, (ii) replacing $y_i=y_j$ by $\top$ if $f(y_i)=f(y_j)$ and by $\bot$ otherwise, (iii) replacing $y_i=t$ or or $t=y_i$ for $t\in\boldsymbol{z}\cup\boldsymbol{e}$ by $\top$ if $f(y_i)=t$ and by $\bot$ otherwise (and dropping the quantifiers corresponding to quantified variables that no longer occur in the formula). Finally, we conjoin every atom $R\boldsymbol{t}$ or $X\boldsymbol{t}$ with $\bigwedge_{t\in\boldsymbol{t},e\in\boldsymbol{e},t\neq e}\neg(t=e)$. Because of the restrictions on negation in ψ, this preserves the semantics of the formula over structures satisfying $\xi_{\equiv}$. The size blowup involved in this procedure is at most exponential in the width of ϕ.

We obtain the desired equality-normalized $\phi'_{\equiv}$ in normal form by performing this rewriting in a bottom-up fashion starting with the innermost quantifier. By using a DAG-representation, the overall size of this equality-normalized form is at most exponential in k (we remark that using a tree representation, it would be exponential in both k and the maximal nesting depth of quantifiers in the formula). Furthermore, even though we introduce disjunctions, the resulting formula is easily seen to be still in normal form. The bound on the width is immediate from the construction. ■

Using this lemma, we can prove the following generalization of Proposition 42.

Proposition 51. *Let $\psi(x)$ be a formula in* UNFP *with $k=|\psi|$. We can construct a DAG-representation of an equivalent equality-normalized $\varphi\in$* UNFP^k *in normal form such that*

- $|\varphi|\leq 2^{f(k)}$,
- $|\mathrm{cl}_+(\varphi,U_k)|\leq 2^{f(k)}$,

where f is a polynomial function independent of ψ.

The important thing to note is that going from a UNFP sentence to an equality-normalized and normal form formula is no more costly than just going to normal form.

Thus, we have shown that for sentences in GSO and UNFP, we can convert to an equality-normalized form with very restricted use of equality.

C. *Equality-trivial coded structures*

The next step is to ensure that the coded structures use only a limited form of equality too. We say that a consistent tree $\mathcal{T}$ is *equality-trivial* if for all nodes v in $\mathcal{T}$ (i) $x=x'$ is asserted at v only if x and x' are the same variable name, and (ii) $\neg(x=x')$ is asserted at v only if x and x' are distinct names. If $\mathcal{T}$ is an equality-trivial consistent tree then distinct terms are realized by distinct elements in $\mathfrak{D}(\mathcal{T})$. Moreover, we can modify the definition of the shrewd unravelling to enforce that $\mathcal{S}^k(\mathfrak{B})$ is an equality-trivial consistent tree: this is trivial if $\mathfrak{B}$ has at least k elements (since we can choose k distinct elements in each bag node). But even if the elements from $\mathfrak{B}$ represented at some bag node are not distinct, in $\mathcal{S}^k(\mathfrak{B})$ we can declare that these copies are distinct, and the structure $\mathfrak{D}(\mathcal{S}^k(\mathfrak{B}))$ obtained like this is still UN^k-bisimilar to $\mathfrak{B}$.

The most important property, however, is that for equality-normalized sentences, equality-trivial consistent trees contain all of the information necessary to evaluate whether some sentence holds in the corresponding structure, just treating equality as any other relation.

For instance, the new version of the forward mapping theorem (extending Theorem 20) would be:

Theorem 52. *Let φ be a sentence in equality-normalized* UNFP$^k[\sigma]$ *in normal form. We can construct a 2-way alternating μ-automaton $\mathcal{A}_\varphi$ such that for all equality-trivial consistent $\tilde{\sigma}_k$ trees $\mathcal{T}$, $\mathfrak{D}(\mathcal{T}) \models \varphi$ iff $\mathcal{T} \models L(\mathcal{A}_\varphi)$ and the size of $\mathcal{A}_\varphi$ is doubly exponential in $|\varphi|$, but the number of states of $\mathcal{A}_\varphi$ is at most singly exponential in $|\varphi|$, and the number of priorities is at most $|\varphi|$. These bounds hold even if we start with a sentence in* UNFP^k *that is not in normal form or equality normal form.*

D. Equality extension of Theorem 3

We can now extend Theorem 3 for sentences with equality and constants. Given some UNFP sentence with equality and constants, we can convert into normal form and equality-normalized form using Proposition 51, and then use the algorithm described in Section IV and Appendix D, working over coded structures that are equality-trivial and treating equality like any other relation. Proposition 51 ensures that the conversion to this equality-normalized version does not change the complexity of the forward mapping, or the complexity of the algorithm as a whole.

Note that Theorem 24 can be extended in a similar way, using Lemma 49 in the forward mapping step.

Appendix G
Failure of Interpolation

Proof of failure of uniform interpolation for GSO *(Proposition 26)*

Recall the statement:

> Uniform interpolation fails for GSO. In particular, there is a GF antecedent with no uniform interpolant in GSO, even when the consequents are restricted to sentences of GF (or UNF) of width 2.

Proof: Let $\varphi \in \text{GF}[\sigma]$ for $\sigma = \{G, P, Q, R_1, R_2, S\}$ be

$$\forall z.[Qz \rightarrow \exists xy.(Gzzxy \wedge Sxy \wedge R_1zx \wedge R_2zy)] \wedge$$
$$\forall xy.[Sxy \rightarrow \exists x'y'.(Gxyx'y' \wedge Sx'y' \wedge R_1xx' \wedge R_2yy' \wedge ((Px' \wedge Py') \vee (\neg Px' \wedge \neg Py')))]$$

which implies that there is an infinite “ladder” starting at every Q-node (where S connects pairs of elements on the same rung, and R_i connects corresponding elements on different rungs) and the pair of elements on each rung agree on P. The relation G is used as a dummy guard to ensure that the formula is in GF.

Then for each n, we can define over $\sigma' = \{P, Q, R_1, R_2\}$ a formula ψ_n

$$(\exists x.(Qx \wedge \forall x_1 \ldots x_n.((\textstyle\bigwedge_i R_1x_ix_{i+1} \wedge x_1 = x) \rightarrow Px_n))) \rightarrow$$
$$(\exists y.(Qy \wedge \exists y_1 \ldots y_n.(\textstyle\bigwedge_i R_2y_iy_{i+1} \wedge y_1 = y \wedge Py_n)))$$

which expresses that if there is some Q-position x such that every R_1-path of length n from x ends in a position satisfying P, then there is an R_2-path of length n from some Q-position y that ends in a position satisfying P. Note that for all n, ψ_n can be written in either GF or UNF of width 2, and $\varphi \models \psi_n$.

Assume for the sake of contradiction that there is some uniform interpolant θ in GSO.

Over trees, GSO coincides with MSO ([24], as cited in [11]). Hence, there is an equivalent θ' in MSO over tree structures. This means we can construct from θ' a nondeterministic parity tree automaton $\mathcal{A}$ that recognizes precisely the language of trees (with branching degree at most 2, say) where θ' holds.

Let m be the number of states in $\mathcal{A}$. Consider the ladder structure $\mathfrak{A}_m$ consisting of a single element a from which there is an infinite R_1-chain of distinct elements and an infinite R_2-chain of distinct elements, where the i-th elements on each chain are connected by S, elements on level i and $i+1$ are guarded by G, P holds only at the $(m+1)$-st element in each chain, and Q holds only at a.

Because $\mathfrak{A}_m \models \varphi$, we have $\mathfrak{A}_m \models \theta$. But over σ', $\mathfrak{A}_m$ is a tree with branching degree at most 2, so $\mathfrak{A}_m \models \theta'$. Hence, there is an accepting run of $\mathcal{A}$ on $\mathfrak{A}_m$. Using a pumping argument, we can pump a section of the R_2 branch before the P-labelled element in order to generate an accepting run of $\mathcal{A}$ on a new tree $\mathfrak{A}'_m$ where P holds at the $(m+1)$-st element in the R_1-chain and P does not hold at that position in the R_2 chain. Hence, this new tree $\mathfrak{A}'_m$ is a model for both θ' and θ. But $\mathfrak{A}'_m \not\models \psi_m$, contradicting the fact that θ is a uniform interpolant. ■

Proof of failure of interpolation for GNFP *(Proposition 27)*

Recall the statement:

> Craig interpolation fails for GNFP. In particular, there is a GFP antecedent and GFP consequent with no GNFP interpolant, even over finite structures.

Proof: Define the GFP[σ] sentence φ over signature $\sigma = \{G, Q, R\}$ to be $\forall x.(Qx \to \varphi')$ where φ' is

$$[\mathbf{lfp}\, X, xy.Gxyy \wedge (Rxy \vee \exists y'.(Gxy'y \wedge Rxy' \wedge Xy'y)](xx)$$

which implies that there is an R-loop from every Q-labelled element.

Define the GFP[σ'] sentence ψ over signature $\sigma' = \{P, Q, R\}$ to be

$$\forall x.((Qx \wedge Px) \to [\mathbf{lfp}\, X, x.\exists y.(Rxy \wedge (Py \vee Xy))](x))$$

which expresses that for all Q and P labelled elements x, there is an R-path from x leading to some node y with Py.

We claim $\varphi \models \psi$.

Now suppose for the sake of contradiction that there is a GNFP[$\sigma \cap \sigma'$]-interpolant χ for $\varphi \models \psi$. Let k be the width of χ (in GN-normal form).

Let $\mathfrak{A}$ be the σ-structure with a single R-loop of length $k+1$, with every element in the loop satisfying Q, and a single element satisfying P. Let $\mathfrak{B}$ be the structure built by starting from an R-chain of three elements, and then adding an R-loop of length $k+1$ to the first and third elements, and labelling all elements with Q, and the second element with P. In other words, $\mathfrak{B}$ has two lassos, one terminating at the second element, and the other starting from the second element. Notice that $\mathfrak{A}$ satisfies φ and ψ, but $\mathfrak{B}$ does not because the second element is not part of an R-cycle.

We claim that $\mathfrak{A}$ and $\mathfrak{B}$ are indistinguishable by GNFP[$\sigma \cap \sigma'$] sentences of width k. We now define a winning strategy for Duplicator in the GN^k-bisimulation game between $\mathfrak{A}$ and $\mathfrak{B}$. This is the game associated with the sublogic GNF^k from [15]. It is an infinite game played on a pair of structures $\mathfrak{A}, \mathfrak{B}$ by two players: Spoiler and Duplicator. The game has two kinds of positions:

i) pairs of guarded tuples $(\bar{m}, \bar{n})$, such that $\bar{m} \mapsto \bar{n}$ is a partial isomorphism from $\mathfrak{A}$ to $\mathfrak{B}$; and
ii) partial homomorphisms $h : \mathfrak{A}\restriction_X \to \mathfrak{B}\restriction_Y$ or $g : \mathfrak{B}\restriction_Y \to \mathfrak{A}\restriction_X$, where $X \subset \mathfrak{A}$ and $Y \subset \mathfrak{B}$, both finite and $|X|, |Y| \leq k$.

From a position $(\bar{m}, \bar{n})$ Spoiler must choose a finite subset $X \subset \mathfrak{A}$ or a finite subset $Y \subset \mathfrak{B}$, in either case of size at most k, upon which Duplicator must respond by a homomorphism with domain X or Y accordingly, mapping it into the other structure in a manner consistent with $\bar{m} \mapsto \bar{n}$. From a position $h : X \to Y$ Spoiler chooses a guarded tuple $\bar{m}$ inside X taking the play to the position $(\bar{m}, h(\bar{m}))$. Similarly, from a position $g : Y \to X$ Spoiler chooses a guarded tuple $\bar{n}$ in Y taking the play to the position $(g(\bar{n}), \bar{n})$. Spoiler wins if he can force the play into a position from which Duplicator cannot respond, and Duplicator wins if she can continue to play indefinitely.

Duplicator's strategy will maintain the property that any group of R-connected elements that Spoiler selects in the active structure are R-connected in the other structure, and any elements that are R-connected to the starting position in the active structure are R-connected to the starting position in the other structure. Notice that any position in the game consists of at most two R-connected elements.

If the active structure is $\mathfrak{A}$, then when Spoiler selects his set of elements, there must be at least one element in the loop that is not selected. For pebbles that are forward (respectively, backward) connected to the starting position, we place these in the corresponding forward (respectively, backward) connected positions in $\mathfrak{B}$. Any other blocks of pebbles that are not connected to the starting position can be placed arbitrarily (as long as R-connected blocks stay together).

If the active structure is $\mathfrak{B}$, then the most interesting case is when Spoiler plays pebbles both inside a lasso and outside of it. For instance, if the starting position is the first element in the chain, and Spoiler selects both R-successors of this position (i.e. the second element in the chain), and the first element in the R-loop starting from the first element), then Duplicator maps both of these pebbles to the same element in $\mathfrak{A}$ (the single successor of the starting position in $\mathfrak{A}$). This is acceptable, because it is not possible for Spoiler to select all of the elements in a single lasso, which would be needed to distinguish between these two different successors. Using this sort of strategy, Duplicator can always choose her pebble positions in $\mathfrak{A}$ so that R-connected blocks of elements are preserved.

Since $\mathfrak{A} \models \varphi$, we have $\mathfrak{A} \models \chi$. Hence, $\mathfrak{B} \models \chi$. But this implies that $\mathfrak{B} \models \psi$, which is a contradiction. ■