A Step Up in Expressiveness of Decidable Fixpoint Logics

Michael Benedikt, Pierre Bourhis, Michael Vanden Boom

A step up in expressiveness of decidable fixpoint logics

Michael Benedikt
University of Oxford

Pierre Bourhis
CNRS CRIStAL, Université Lille 1, INRIA Lille

Michael Vanden Boom
University of Oxford

Abstract

Guardedness restrictions are one of the principal means to obtain decidable logics — operators such as negation are restricted so that the free variables are contained in an atom. While guardedness has been applied fruitfully in the setting of first-order logic, the ability to add fixpoints while retaining decidability has been very limited. Here we show that one of the main restrictions imposed in the past can be lifted, getting a richer decidable logic by allowing fixpoints in which *the parameters of the fixpoint can be unguarded*. Using automata, we show that the resulting logics have a decidable satisfiability problem, and provide a fine study of the complexity of satisfiability. We show that similar methods apply to decide questions concerning the elimination of fixpoints within formulas of the logic.

1. Introduction

We are interested in expressive logics for which static analysis problems such as satisfiability are decidable. One way to achieve decidability is via *guardedness restrictions*. The Guarded Fragment (GF) [2] is a fragment of first-order logic obtained by requiring in existential quantification $\exists \boldsymbol{x}.\phi(\boldsymbol{x})$ that ϕ be of the form $R(\boldsymbol{x}) \wedge \phi'(\boldsymbol{x})$, where $R(\boldsymbol{x})$ is an atom containing all free variables of ϕ', and requiring in universal quantification that ϕ be of the form $R(\boldsymbol{x}) \rightarrow \phi'(\boldsymbol{x})$, where R is as above. The Guarded Negation Fragment (GNF) [5] is an even more expressive decidable language, allowing unrestricted existential quantification but restricting negation to be of the form $R(\boldsymbol{x}) \wedge \neg\phi'(\boldsymbol{x})$, with R as above.

Of course, there are other paradigms for decidability within first-order logic, such as restricting to two variables [28]. A particularly attractive feature of guarded logics is that, unlike with the two variable fragment [20], decidability can be extended to give decidable fragments of *least fixpoint logic* (LFP). LFP is the natural extension of first-order logic with a fixpoint constructor. Given a formula $\phi(x_1 \dots x_m, y_1 \dots y_n)$ over some signature σ in which an m-ary second-order variable X occurs freely and positively in ϕ, and given a σ-structure $\mathfrak{A}$ and some fixed valuation ρ for $\boldsymbol{y}$, we can define a new m-ary relation: the relation is defined as the limit of a monotone sequence $X_0 \dots$, starting with $X_0 = \emptyset$ and then setting X_{i+1} to be the set of $\boldsymbol{a}$ such that ϕ holds when extending ρ with the interpretation of X as X_i and $\boldsymbol{x}$ as $\boldsymbol{a}$. Emphasizing the distinction between $\boldsymbol{x}$ and $\boldsymbol{y}$, we call $\boldsymbol{x}$ the *fixpoint variables* and $\boldsymbol{y}$ the *parameter variables*. Informally, during the fixpoint process, the fixpoint variables change in each iteration, while the parameter variables stay the same. Formulas in LFP can use relations defined using a fixpoint constructor like this, in addition to relations in σ.

Guarded Fixpoint Logic (denoted GFP or μGF) [19] extends GF with a fixpoint operator while maintaining decidability. The fixpoint constructor is restricted in two ways: the parameter variables $\boldsymbol{y}$ must be empty, and the fixpoint relation itself cannot be used as a guard. Guarded Negation Fixpoint Logic (GNFP) [5] adds fixpoint constructors to GNF with similar restrictions.

It is known that the second restriction on these fixpoint logics is essential for decidability [19]. But what about the first? It certainly seemed important to the proofs of decidability; [21] states

> It should be stressed that the presence of extra first-order parameters in fixed-point operations as well as the use of second-order variables and fixed points as guards is disallowed in μGF. These restrictions are essential for keeping the semantics invariant under guarded bisimulation. For instance, with the use of a first-order parameter ... one can define the transitive closure of a binary relation ... However, the transitive closure query is not invariant under guarded bisimulation and it is known that adding transitive closure to GF produces an undecidable logic [17].

In this paper we show that the parameter restriction can indeed be loosened. We introduce a variation of GNFP, denoted GNFP-UP, where the fixpoint variables of any fixpoint need to be guarded, but the fixpoint can carry additional *unguarded parameters*. One can write a GNFP-UP formula holding on the transitive closure of a binary relation. But such a formula cannot be used as a guard, and thus assertions that a binary relation is transitive (the key to undecidability in [17]) cannot be expressed. GNFP-UP can express many properties related to transitivity, such as assertions of paths with certain properties (see the discussion of conjunctive regular path queries with inverse in Section 3).

The decidability of GFP is proven using an elegant high-level argument [18]: one shows that satisfiable formulas must have tree-like models, and thus satisfiability can be reduced to satisfiability of a Monadic Second Order Logic sentence over trees, decidable via Rabin's theorem [26]. A finer argument shows that from a GFP formula ϕ one can effectively create a tree automaton $\mathcal{A}_\phi$ which is non-empty exactly when ϕ is satisfiable. By analyzing the complexity of this automaton construction, Grädel and Walukiewicz derived a 2-ExpTime bound on satisfiability [19].

We begin by showing that the high-level argument easily extends to give decidability of GNFP-UP. The finer analysis of the complexity of GNFP-UP satisfiability requires more work. Because of negation and quantification in our logic, one might expect that the complexity would be a tower of exponentials based on the quantifier alternation. However, we show that the complexity is controlled by the *parameter depth* of the formula: informally, this is a

number that measures the number of times we change parameters while passing from a formula to a subformula. We give elementary bounds for each parameter depth, while proving that the complexity is non-elementary (but still primitive recursive) when the depth is not restricted. Each parameter depth includes formulas of arbitrary quantifier alternation; we avoid unnecessary exponential blowups by identifying pieces of the GNFP-UP formulas that behave like GNFP. We also show that some interesting logics fit within low parameter depth.

Finally, we describe how our translation to automata can be modified to show decidability of the *boundedness problem*, a quantitative analog of satisfiability that asks if a fixpoint can be replaced by a fixed number of unfoldings. Our solution to the boundedness problem can be applied to decide whether certain GNFP-UP-expressible formulas can be rewritten in first-order logic.

2. Preliminaries

We consider signatures σ with a finite set of relations and constants. We write const(σ) for the set of constants in the signature σ.

Least fixed point logic (LFP) over σ is the extension of first-order logic over σ with the following formation rule: if $\phi(\boldsymbol{y}, \boldsymbol{z}, Y, \boldsymbol{Z})$ is a formula in which Y is a free second-order variable of arity $|\boldsymbol{y}|$ that appears only positively, and $\boldsymbol{t}$ is a tuple of variables and constants of length $|\boldsymbol{y}|$, then $[\mathbf{lfp}_{Y,\boldsymbol{y}} \,.\, \phi](\boldsymbol{t})$ is also a formula. Informally, it asserts that $\boldsymbol{t}$ is in the least fixpoint induced by ϕ. The *parameter variables* of this formula are $\boldsymbol{z}$ — the free variables of ϕ other than $\boldsymbol{y}$. In this paper, we emphasize the parameters by writing $[\mathbf{lfp}^{\boldsymbol{z}}_{Y,\boldsymbol{y}} \,.\, \phi]$, and reserve $[\mathbf{lfp}_{Y,\boldsymbol{y}} \,.\, \phi]$ for the case when there are no parameters.

The semantics are standard: since $\phi(\boldsymbol{y}, \boldsymbol{z}, Y, \boldsymbol{Z})$ is positive in Y, it induces a monotone operator

$$U \mapsto O^{\mathfrak{A},\boldsymbol{v},\boldsymbol{V}}_{\phi}(U) := \{\boldsymbol{u} : \mathfrak{A}, \boldsymbol{u}, \boldsymbol{v}, U, \boldsymbol{V} \models \phi(\boldsymbol{y}, \boldsymbol{z}, Y, \boldsymbol{Z})\}$$

on every structure $\mathfrak{A}$ with valuation $\boldsymbol{v}$ for $\boldsymbol{z}$ and $\boldsymbol{V}$ for $\boldsymbol{Z}$. Hence, this operator has a unique *least fixpoint*, denoted $\phi^{\infty}(\mathfrak{A}, \boldsymbol{v}, \boldsymbol{V})$, and this can be obtained as the union of its *fixpoint approximants* $\phi^{\beta}(\mathfrak{A}, \boldsymbol{v}, \boldsymbol{V})$ over all ordinals β, where

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

The semantics of the least fixpoint formulas is defined such that $\mathfrak{A}, \boldsymbol{v}, \boldsymbol{V} \models [\mathbf{lfp}^{\boldsymbol{z}}_{Y,\boldsymbol{y}} \,.\, \phi](\boldsymbol{t})$ iff $\boldsymbol{t} \in \phi^{\infty}(\mathfrak{A}, \boldsymbol{v}, \boldsymbol{V}) = \bigcup_{\beta \in \mathrm{Ord}} \phi^{\beta}(\mathfrak{A}, \boldsymbol{v}, \boldsymbol{V})$.

Free and bound variables The notion of free vs. bound second-order variables is standard. In particular, Y is free in $\phi(\boldsymbol{y}, \boldsymbol{z}, Y, \boldsymbol{Z})$ but bound in $[\mathbf{lfp}^{\boldsymbol{z}}_{Y,\boldsymbol{y}} \,.\, \phi](\boldsymbol{t})$. We assume no second-order variable Y is bound by more than one fixpoint operator, so each bound second-order variable Y identifies a unique fixpoint. If Y identifies a fixpoint with parameters $\boldsymbol{z}$, then params(Y) := $\boldsymbol{z}$, the *parameters associated with the second-order variable* Y.

We use free(ϕ) to denote the *free first-order variables* in ϕ. It is defined recursively. For atoms $R\boldsymbol{t}$ with $R \in \sigma$ and $\boldsymbol{t}$ a tuple consisting of constants and variables, the free first-order variables are just the variables in $\boldsymbol{t}$. For $Y\boldsymbol{t}$ with Y a second-order variable, free($Y\,\boldsymbol{t}$) is the union of the variables in $\boldsymbol{t}$ *and* params(Y). For boolean connectives, free($\phi_1 \wedge \phi_2$) = free($\phi_1 \vee \phi_2$) = free(ϕ_1) $\cup$ free(ϕ_2), and free($\neg\phi$) = free(ϕ). For quantification, free($\exists x.\phi$) = free(ϕ) $\setminus$ $\{x\}$. Finally, for $[\mathbf{lfp}^{\boldsymbol{z}}_{Y,\boldsymbol{y}} \,.\, \phi](\boldsymbol{t})$, the free first-order variables consist of the parameter variables $\boldsymbol{z}$ together with the variables in $\boldsymbol{t}$.

The parameters in ϕ consists of the union of params(Y) for all second-order variables Y occurring in ϕ; we let params(ϕ) denote the *subset of these parameters that occur free in* ϕ.

GNFP-UP *Guarded negation fixpoint logic with unguarded parameters* (GNFP-UP) is the fragment of LFP that allows unguarded parameter variables in fixpoint definitions, but requires fixpoint variables and negation to be guarded. Formally, a GNFP-UP[σ] *formula* ϕ is generated recursively from the following grammar:

$$\begin{aligned}
\phi ::= {} & R\,\boldsymbol{t} \mid Y\,\boldsymbol{t} \mid \phi \wedge \phi \mid \phi \vee \phi \mid \exists y.\phi \mid \\
& \alpha \wedge \neg\phi \text{ where free}(\phi) \subseteq \text{free}(\alpha) \mid \\
& [\mathbf{lfp}^{\boldsymbol{z}}_{Y,\boldsymbol{y}} \,.\, \text{gdd}(\boldsymbol{y}) \wedge \phi(\boldsymbol{y}, \boldsymbol{z}, Y, \boldsymbol{Z})](\boldsymbol{t}) \text{ for } \phi \text{ positive in } Y
\end{aligned}$$

where $\boldsymbol{t}$ is a tuple of variables or constants, $R\boldsymbol{t}$ and α are atoms using a relation in σ or =, and gdd($\boldsymbol{y}$) is defined below.

Guardedness The *guardedness predicate* gdd($\boldsymbol{y}$) asserts $\boldsymbol{y}$ is guarded by an atom in σ or =. It can be understood as an abbreviation for the disjunction of existentially quantified atoms that use a relation from σ or = and involve all of the variables in $\boldsymbol{y}$. Because of this, only guarded relations can be defined using fixpoints in GNFP-UP: i.e. any tuple of elements in the relation defined by the fixpoint formula must already be guarded by an atom in the base signature σ. Note that the relations defined using a fixpoint operator cannot be used as guards.

The parameters $\boldsymbol{z}$ are not required to be guarded in the fixpoint definition. However, for the purposes of negation, parameters are treated like other variables and must be guarded. For example, if $\alpha(\boldsymbol{x})$ is an atomic formula over σ, and Y identifies a fixpoint with parameters $\boldsymbol{z}$, then $\alpha(\boldsymbol{x}) \wedge \neg Y\,\boldsymbol{x}$ is not permitted since Y implicitly uses parameters $\boldsymbol{z}$ and these parameters are not guarded by α (since the free variables in $Y\,\boldsymbol{x}$ are really $\boldsymbol{x}$ and $\boldsymbol{z}$).

A formula ϕ that includes free first-order variables $\boldsymbol{x}$ is $\boldsymbol{x}$-*guarded* if it is logically equivalent to gdd($\boldsymbol{x}$) $\wedge$ ϕ. If free(ϕ) = $\boldsymbol{x}$ and ϕ is $\boldsymbol{x}$-guarded, then we say it is *answer-guarded*. Sentences or formulas with one free variable are always answer-guarded since we can use a trivial guard like $x = x$. For readability purposes, we often omit such trivial guards.

Simultaneous fixpoints We will also allow simultaneous fixpoints $[\mathbf{lfp}^{\boldsymbol{z}}_{Y_i,\boldsymbol{y}_i} .S\,](\boldsymbol{t})$ where

$$S = \begin{cases} Y_1, \boldsymbol{y}_1 := \text{gdd}(\boldsymbol{y}_1) \wedge \phi_1(\boldsymbol{y}_1, \boldsymbol{z}, Y_1, \ldots, Y_j, \boldsymbol{Z}) \\ \quad\vdots \\ Y_j, \boldsymbol{y}_j := \text{gdd}(\boldsymbol{y}_j) \wedge \phi_j(\boldsymbol{y}_j, \boldsymbol{z}, Y_1, \ldots, Y_j, \boldsymbol{Z}) \end{cases}$$

is a system of mutually defined equations $Y_i, \boldsymbol{y}_i := \phi_i$ such that $\phi_i \in$ GNFP-UP, and $Y_1, \ldots, Y_j$ occur only positively in ϕ_i. Each ϕ_i utilizes the same parameters $\boldsymbol{z}$, but different fixpoint variables $\boldsymbol{y}_i$. Such a system defines a monotone operation on vectors of relations, and $[\mathbf{lfp}^{\boldsymbol{z}}_{Y_i,\boldsymbol{y}_i} .S\,](\boldsymbol{t})$ expresses that $\boldsymbol{t}$ is a tuple in the i-th component of the least fixpoint defined by this operation; Y_i is the distinguished *goal predicate* in $[\mathbf{lfp}^{\boldsymbol{z}}_{Y_i,\boldsymbol{y}_i} .S\,](\boldsymbol{t})$. Allowing simultaneous fixpoints does not change the expressivity of the logic, since they can be eliminated in favor of traditional fixpoints [3], with a possible exponential blow-up in size. However, it is often more convenient to work directly with these simultaneous fixpoints (see Example 4).

GNFP-UP *vs.* GNFP A good example to keep in mind is that GNFP-UP can express the transitive closure of a binary relation R.

Example 1. *Suppose R is a binary relation in σ. Consider the following* GNFP-UP[σ]*-formula:*

$$\phi(x, z) := [\mathbf{lfp}^{\boldsymbol{z}}_{Y,\boldsymbol{y}} \,.\, Ryz \vee \exists y'.(Ryy' \wedge Yy')](x)\,.$$

Observe that ϕ has two free variables, the variable x being tested in the fixpoint and the parameter variable z. The formula $\phi(x, z)$ expresses that there is some R-path from element x to z, i.e. (x, z) is in the transitive closure of R.

We can express that x participates in an R-cycle by using the formula $\phi(x, x)$.

We cannot express that the structure is strongly R-connected, since this would require unguarded negation, but we can say every pair of guarded elements is R-connected: $\neg\exists xz.(\mathrm{gdd}(x,z) \wedge \neg\phi(x,z)) \in$ GNFP-UP.

The sentence $\neg\exists xz.(\phi(x,z) \wedge \neg Rxz) \vee (Rxz \wedge \neg\phi(x,z))$ *that says R is transitively closed is not in* GNFP-UP*, since we cannot use the fixpoint relation defined by* ϕ *as a guard for* $\neg Rxz$*.*

In LFP, it is always possible to eliminate the use of parameters by increasing the arity of the defined fixpoint predicates and passing the parameters explicitly in the fixpoint. This is not usually possible in our context, because the fixpoint variables are required to be guarded. Indeed, it can be shown that the transitive closure of a binary relation R cannot be expressed in GNFP (the fragment of GNFP-UP in which fixpoints do not use any parameters).

Proposition 2. GNFP-UP *is strictly more expressive than* GNFP*, even over finite structures.*

Normal form A *conjunctive query* (CQ) is $\exists \boldsymbol{y}.\psi$ for ψ a conjunction of atoms. A *union of conjunctive queries* (UCQ) is a disjunction of CQs. Such queries are expressible in GNF. It is helpful to work with GNFP-UP in a normal form that highlights the fact that GNFP-UP formulas can be built up from UCQ-shaped formulas using guarded negation and guarded fixpoints with parameters.

Formally, A *normal form* GNFP-UP[σ] *formula* ϕ or ψ is generated recursively from the following grammars:

$$\begin{aligned}\phi &::= \textstyle\bigvee_i \exists \boldsymbol{y}_i.\, \bigwedge_j \psi_{ij}\\ \psi &::= R\,\boldsymbol{t} \mid Y\,\boldsymbol{t} \mid \alpha \wedge \neg\phi \text{ where } \mathrm{free}(\phi) \subseteq \mathrm{free}(\alpha) \mid\\ &\qquad [\mathbf{lfp}^{\boldsymbol{z}}_{Y_m,\boldsymbol{y}_m}.S](\boldsymbol{t})\end{aligned}$$

where $\boldsymbol{t}$ is a tuple of variables or constants, $R\boldsymbol{t}$ and α are atoms using a relation in σ or $=$, and S is a system with equations of the form $Y, \boldsymbol{y} := \mathrm{gdd}(\boldsymbol{y}) \wedge \phi(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{Y}, \boldsymbol{Z})$, as described earlier.

Any GNFP-UP formula can be converted into normal form in a canonical way. The *width* of a GNFP-UP formula is the maximum number of free variables used in any subformula after the formula is converted into normal form. We write (GNFP-UP)$^k[\sigma]$ for GNFP-UP formulas of width k.

Other guarded logics with parameters We could consider variants of other guarded logics with parameters. The *unary negation fixpoint logic with unguarded parameters* (UNFP-UP) is the fragment of GNFP-UP where only formulas with at most one free variable are negated, and the fixpoint predicates are monadic but may still carry any number of unguarded parameters. The formula $\phi(x,z)$ in Example 1 is in UNFP-UP.

We could also consider GFP-UP, the variant of GFP with parameters in the fixpoints, but where these parameters still need to be guarded if appearing under a quantification (this mimics the way we have added parameters to GNFP). The formula ϕ in Example 1 is not in GFP-UP since the quantification in $\exists y'.(Ryy' \wedge Yy')$ is not correctly guarded: GFP-UP would require something like $\exists y'.(Gzy' \wedge Yy')$, which includes a guard that covers not only y' but also the parameter z implicit in the fixpoint predicate Y. However, adding parameters to GFP in this way does not increase expressivity:

Proposition 3. *For every answer-guarded* GFP-UP *formula* ϕ*, we can construct in linear time an equivalent* GFP *formula* ϕ' *using fixpoint predicates of higher arity.*

This may explain why parameters were not considered further in [19, 21]: when they are introduced to GFP in their full power, it leads to undecidability, and when they are introduced in this more restrictive way, they do not add any expressive power. This is in contrast to GNFP, where we can add parameters in a way that strictly increases expressivity while still retaining decidability.

Organization In Section 3, we give some examples illustrating the expressive power of GNFP-UP. We also define the parameter depth, a key measure of how complicated a GNFP-UP formula is. We then argue in Section 4 that GNFP-UP has tree-like models, which makes satisfiability and boundedness amenable to tree automata techniques. The main technical work is described in Sections 5 and 6 where we describe the automata tools needed for GNFP-UP. Finally, in Sections 7 and 8, we use this automata machinery to study static analysis problems like satisfiability and boundedness.

3. Expressivity examples

In this section, we give some examples showing that GNFP-UP subsumes and extends a wide range of logics. This provides evidence of its power, and explains some of the good properties of these previously-studied logics.

In order to help study the power of GNFP-UP, we first define a way to measure how the parameters are used. Roughly speaking, the *parameter depth* is the maximum number of nested parameter changes. We define $\mathrm{pdepth}_{\boldsymbol{z}}(\eta)$ inductively as follows:

$$\begin{aligned}&\mathrm{pdepth}_{\boldsymbol{z}}(R\,\boldsymbol{t}) = \mathrm{pdepth}_{\boldsymbol{z}}(Y\,\boldsymbol{t}) := 0\\ &\mathrm{pdepth}_{\boldsymbol{z}}(\alpha \wedge \neg\phi) := \mathrm{pdepth}_{\boldsymbol{z}\cap\mathrm{free}(\phi)}(\phi)\\ &\mathrm{pdepth}_{\boldsymbol{z}}(\textstyle\bigvee_i \exists \boldsymbol{x}_i.\, \bigwedge_j \psi_{ij}) := \max_i p_i \text{ s.t.}\\ &\qquad p_i := \begin{cases} 1 + \max_j \mathrm{pdepth}_{\mathrm{params}(\psi_{ij})}(\psi_{ij}) & \text{if } \exists j.\, \mathrm{params}(\psi_{ij}) \not\subseteq \boldsymbol{z}\\ \max_j \mathrm{pdepth}_{\boldsymbol{z}}(\psi_{ij}) & \text{otherwise}\end{cases}\\ &\mathrm{pdepth}_{\boldsymbol{z}}([\mathbf{lfp}^{\boldsymbol{z}'}_{X_m,\boldsymbol{x}_m}.S](\boldsymbol{t})) :=\\ &\qquad \begin{cases} 1 + \max_{\phi_j\in S} \mathrm{pdepth}_{\mathrm{params}(\phi_j)}(\phi_j) & \text{if } \exists j.\, \mathrm{params}(\phi_j) \not\subseteq \boldsymbol{z}\\ \max_{\phi_j\in S} \mathrm{pdepth}_{\boldsymbol{z}}(\phi_j) & \text{otherwise}\end{cases}.\end{aligned}$$

The parameter depth $\mathrm{pdepth}(\phi)$ for normal form $\phi \in$ GNFP-UP is just $\mathrm{pdepth}_{\mathrm{free}(\phi)}(\phi)$. For ϕ not necessarily in normal form, we define it to be the pdepth after converting to normal form.

Observe that a formula that does not use any parameters has pdepth 0. Even a formula that does use parameters can have pdepth 0 if all of its parameters actually come from free variables of the formula. This is because parameters like this can be viewed as constants, since they have a fixed interpretation in any structure. Because of this, if $\phi \in$ GNFP-UP[σ] with $\mathrm{pdepth}(\phi) = 0$ and $\mathrm{params}(\phi) = \boldsymbol{z}$, then we can view ϕ as a GNFP formula without parameters, over the signature σ extended with extra constants $\boldsymbol{z}$.

In general, the pdepth increases when we pass through a subformula that introduces more parameters. This can happen when passing through existential quantification that introduces a variable that is later used as a parameter (see the third case in the pdepth definition), or it can happen when passing through a fixpoint definition that introduces a fixpoint variable that is later used as a parameter (see the fourth case).

Later, we will see that the parameter depth is the major factor impacting the complexity of satisfiability testing.

We now give some examples to illustrate the expressivity of GNFP-UP, and this notion of parameter depth. These examples are drawn mostly from query languages used in databases and knowledge representation. Understanding these different logics and query languages is not important for understanding the main results about GNFP-UP (e.g., Theorem 20 and Theorem 24). However, for readers familiar with some of these previously-studied logics, they may give some insight into the sort of properties that can be expressed in GNFP-UP. Indeed, it is interesting to note that many of the previously-studied logics described below are low in this parameter depth hierarchy.

Traditional guarded logics GNFP-UP subsumes all of the previously mentioned guarded logics (and their fixpoint extensions), including GFP sentences, UNFP formulas, and GNFP formulas. Unsurprisingly, these traditional guarded logics without parameters can be expressed as GNFP-UP formulas of pdepth 0.

Navigational queries There are a number of languages for navigational queries in graph databases, where the signature σ consists only of binary and unary relations. For these languages, a regular expression E over symbols R, R^- coming from binary relations $R \in \sigma$ can be seen as defining a *navigation relation* that holds for (x, y) exactly when there is some path between x and y matching E. A *conjunctive 2-way regular path query* (C2RPQ) [13] is just a CQ over such expressions.

Example 4. *Consider some* C2RPQ *over signature* σ. *Let* $\Sigma := \{R, R^- : R \text{ is a binary relation in } \sigma\}$.

Given a regular expression E *over* Σ, *we can capture the navigation relation defined by it using a* GNFP-UP *formula* E'. *We start with a finite state automaton* $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta, F\rangle$ *for* E *and write a* GNFP-UP$[\sigma]$ *formula with simultaneous fixpoints* $E'(x, y) := [\mathbf{lfp}^{y}_{X_0,x} \ . \ S\,](x)$ *which has a second-order variable* X_i *for each state* $q_i \in Q$, *and the equation for the i-th component* X_i, x_i *in* S *captures the possible transitions from state* q_i*:*

$$\bigvee_{(q_i,T,q_j)\in\Delta} \exists z.(\chi_T(x_i, z) \wedge X_j\, z) \ \vee \ \begin{cases} x_i = y & \text{if } i \in F \\ \bot & \text{if } i \notin F \end{cases}$$

where $\chi_T(x_i, z)$ *is* Rx_iz *if* $T = R$ *and* Rzx_i *if* $T = R^-$.

Once we have E' *in* GNFP-UP *for each regular expression* E *appearing in the* C2RPQ, *it is easy to translate into* GNFP-UP *by replacing each* $E(x, y)$ *in the* C2RPQ *by* $E'(x, y)$. *These* GNFP-UP *formulas have parameter depth 1: the* GNFP-UP *formula* $E'(x, y)$ *for each regular expression predicate* $E(x, y)$ *has pdepth 0; when these are substituted in the CQ, the resulting formula has pdepth at most 1 since the existential quantification may be introducing variables that are used as parameters in the inner formulas.* GNFP-UP *can also express unions of* C2RPQs.

We can in fact replace regular expressions in C2RPQs by a variant of *propositional dynamic logic* (PDL). PDL consists of programs (defining binary relations within a labeled graph) and tests (defining unary relations within a graph) defined by mutual recursion. Programs contain all binary relation symbols and are closed under concatenation, union, and Kleene star. Tests contain all unary relation symbols and are closed under boolean operations. Given a test t, we can define a program t? that returns pairs (x, x) such that x is in the unary relation defined by t, and given a program P we can form a test $\langle P\rangle$, defining a relation consisting of pairs (u, u) such that there exists v with (u, v) in the language described by P. We let CQPDL denote the language of *conjunctive queries where binary relations can be* PDL *programs*. Clearly this subsumes C2RPQs, and it also subsumes extensions defined in the description logic literature [9]. If P is restricted to be a traditional regular expression, then the corresponding GNFP-UP formula for $\langle P\rangle$ has pdepth at most 1. By writing expressions with more complicated nesting of these tests, however, these formulas can reach higher parameter depth levels.

Fragments of Datalog Datalog is a syntax for expressing the negation-free fragment of least fixpoint logic. It is heavily used to express database queries that involve some form of recursion. We argue that all the previously-defined fragments of Datalog that have decidable static analysis problems are contained in GNFP-UP.

Formally, a *Datalog query* is specified by

$$\Pi = \langle \text{EDB}^\Pi, \text{IDB}^\Pi, \text{Rules}^\Pi, \text{goal}\rangle$$

where the extensional predicates EDB^Π and intensional predicates IDB^Π are disjoint sets, Rules^Π consists of formulas of the form $R(x_1 \dots x_n) \leftarrow \psi(\boldsymbol{xy})$ where R is an IDB predicate and ψ is a conjunction of atoms, and goal is a distinguished member of IDB^Π. Given some structure $\mathfrak{A}$ we can evaluate goal in the structure obtained from $\mathfrak{A}$ by firing the rules of Π until a fixpoint has been reached. For a structure $\mathfrak{A}$ and query Π we let $\Pi(\mathfrak{A})$ be the value of the predicate goal so obtained. A *boolean* Datalog query is one where the goal predicate is 0-ary, and hence the query defines a boolean function on input structures.

Monadic Datalog restricts the IDBs to have arity 1. In this case, it is possible to express the query using a UNFP formula with simultaneous fixpoints without parameters. *Frontier-guarded Datalog* allows the use of intensional predicates with unrestricted arities, but for each rule $R(x_1 \dots x_n) \leftarrow \psi(\boldsymbol{xy})$, the variables $x_1 \dots x_n$ in the head of the rule, must appear in a single EDB atom appearing in the body ψ. This subsumes monadic Datalog, since the single head variable in the monadic Datalog rules can be trivially guarded. Frontier-guarded Datalog can be expressed in GNFP. No parameters are necessary, so the parameter depth is 0.

The *flag and check queries* introduced in [12, 27] are based on fragments of Datalog queries that have been shown to have decidable analysis problems.

One family consists of the *monadically defined queries* (MQs). These are of the form $\exists \boldsymbol{y}.\Pi$ where Π is a Monadic Datalog query, the goal predicate is nullary, and the rules use special symbols $\boldsymbol{z}$. The answers to the query are (projections of) assignments to the special symbols for which the corresponding Monadic Datalog query evaluates to true. The idea is that the special symbols serve as flags for potential answers to the query, and the Datalog query checks if the flags mark actual answers.

Example 5 (based on [12])**.** *The transitive closure of a binary relation R can be expressed by the* MQ Π *with special symbols* z_1, z_2 *where* Π *is*

$$U(y) \leftarrow R\, z_1 y \qquad U(y) \leftarrow U x \wedge R\, xy \qquad \mathit{hit}() \leftarrow U z_2\ .$$

The answer to the query would consist of all pairs (a_1, a_2) *for which the rules imply hit under the standard Datalog semantics, when interpreting* z_1 *as* a_1 *and* z_2 *as* a_2.

In UNFP-UP, *this is* $\psi(z_1, z_2) := [\mathbf{lfp}^{z_1,z_2}_{\mathit{hit},\emptyset} \ .S\,]()$ *where*

$$S := \begin{cases} U, y & := \ R\, z_1 y \vee \exists x.(U x \wedge R\, xy) \\ \mathit{hit}, \emptyset & := \ U z_2 \end{cases}$$

Notice that the special symbols become parameters. Because hit is a nullary predicate, the fixpoint is nullary too. It expresses the same property as the formula in Example 1, but is written in an alternative way that mimics the MQ. *It has parameter depth 0.*

We can translate an arbitrary MQ $\exists \boldsymbol{y}.\Pi$ with special symbols $\boldsymbol{z}$ using a similar method: the monadic Datalog query becomes a simultaneous fixpoint ψ' in GNFP-UP, with the special symbols $\boldsymbol{z}$ as parameters, and the special nullary hit predicate as the goal predicate. The MQ itself can then be written in GNFP-UP as $\exists \boldsymbol{y}.\psi'(\boldsymbol{z})$. The resulting formula has pdepth at most 1, since $\psi'(\boldsymbol{z})$ has pdepth 0, and $\exists \boldsymbol{y}.\psi'(\boldsymbol{z})$ may project some of the parameters (the previous example only has pdepth 0 because there is no such projection).

In [27], they also consider a nested version of these flag and check queries. An *m-nested* MQ is one where the monadic Datalog query is allowed to use predicates defined by $(m-1)$-nested MQs in the rule bodies (but these predicates cannot be used as guards); a 1-nested MQ is just the MQ defined above. In general, we can translate an m-nested MQ query into a GNFP-UP formula of pdepth at most m. [12] defines other variants of flag and check queries; all of them can be similarly captured in GNFP-UP.

A Datalog query Π_1 is *contained in* a Datalog query Π_2 if for all input structures $\mathfrak{A}$, $\Pi_1(\mathfrak{A}) \subseteq \Pi_2(\mathfrak{A})$. Similarly given a sentence ϕ in some logic, we say Datalog query Π_1 is *contained in* Π_2 *relative to*

ϕ if $\Pi_1(\mathfrak{A}) \subseteq \Pi_2(\mathfrak{A})$ for all $\mathfrak{A}$ satisfying ϕ. GNFP-UP can express the Datalog queries in the fragments above. Moreover, since it is closed under boolean combinations for sentences, it can also express containment of two boolean queries within each fragment, and containment relative to sentences ϕ that are expressible in GNFP-UP.

Transitive closure logic The extension of FO with a transitive closure operator (rather than a full fixpoint operator) was introduced in [23] and has been studied extensively. In a similar vein, we can consider GNF extended with a *(guarded) transitive closure operator* which we denote GNF(TC). The idea is to add to GNF the following formula building rule: if $\phi(\boldsymbol{x}, \boldsymbol{y}) \in$ GNF(TC) for m-tuples $\boldsymbol{x}$ and $\boldsymbol{y}$, then $[\mathbf{TC}_{\boldsymbol{x},\boldsymbol{y}} \ . \ \mathrm{gdd}(\boldsymbol{x}) \wedge \phi(\boldsymbol{x}, \boldsymbol{y})](\boldsymbol{u}, \boldsymbol{v})$ is a formula in GNF(TC). The intended meaning is that $\boldsymbol{u}$ and $\boldsymbol{v}$ are guarded (in the original signature) and $(\boldsymbol{u}, \boldsymbol{v})$ is in the reflexive transitive closure of the binary relation on m-tuples defined by $\mathrm{gdd}(\boldsymbol{x}) \wedge \phi(\boldsymbol{x}, \boldsymbol{y})$.

Example 1 is $\exists x'.(Rxx' \wedge [\mathbf{TC}_{y,y'} \ . \ Ryy'](x', z))$ (the guardedness predicate is omitted since singletons are trivially guarded). CQPDL, and hence C2RPQs, can also be expressed in GNF(TC).

It is straightforward to check that every GNF(TC) formula can be translated in polynomial time to an equivalent GNFP-UP formula, and that such formulas can reach arbitrary pdepth levels.

Why* GNFP-UP*? The previous examples serve to illustrate the variety of logics that GNFP-UP subsumes. GNFP-UP is useful because it serves as a unifying logic for all of these different formalisms that have some recursive nature.

A major advantage of GNFP-UP over the Datalog-based languages is that the logic has some form of negation. Not only does the presence of negation increase the expressivity of the queries that can be written in this language, but it also means that we can express directly query containment problems in this language, which was not possible in many of these earlier formalisms.

Despite the increased expressivity, we will show that GNFP-UP still has many useful model theoretic and computational properties, including decidable satisfiability and boundedness.

4. Tree-like models and tree encodings

Although GNFP-UP fails to have the finite model property (since it embeds the 2-way μ-calculus), it does have the *tree-like model property*. This says that if there is a model, then there is a model with a tree decomposition of some bounded tree width. A *tree decomposition* of a structure $\mathfrak{A}$ is a tree labelled with atomic facts of $\mathfrak{A}$ such that every atom is present in some label, and for each $a \in \mathrm{dom}(\mathfrak{A})$, the set of nodes (often called *bags*) that mention a form a connected part of the tree. The decomposition has *tree-width* $w-1$ if the number of elements represented in each bag is at most w.

Proposition 6. *Every satisfiable* (GNFP-UP)$^k[\sigma]$ *sentence has a model of tree-width at most* $k + |\mathrm{const}(\sigma)| - 1$.

The proof uses a standard technique, involving an unravelling based on a notion of guarded negation bisimulation.

The first route to deciding satisfiability relies on the tree-like model property of Proposition 6 along with the fact that GNFP-UP can be expressed in a fragment of second-order logic called *guarded second-order logic* (GSO) in which second-order quantification is interpreted only over guarded relations, i.e. relations where every tuple is guarded by some atom in the base signature.

Proposition 7. *Given* $\phi \in$ GNFP-UP$[\sigma]$*, we can construct an equivalent* $\phi' \in$ GSO$[\sigma]$ *in linear time.*

Corollary 8. *Satisfiability for* GNFP-UP *is decidable.*

Proof. GNFP-UP embeds in GSO by Proposition 7 and has bounded tree-width by Proposition 6. Using [21], GSO can be translated into an equivalent MSO formula (over encodings of the tree-like models) which is decidable by [26]. □

This is the easiest route to showing decidability of GNFP-UP, but it is not good for extracting complexity bounds. We next show how to make direct use of the tree-like model property and a translation taking a formula in the logic to tree automata that represent tree-like models of the formula, to determine more precisely the complexity of satisfiability and boundedness for GNFP-UP.

Coding structures Structures of bounded tree-width can be encoded as trees over a finite alphabet that depends only on the signature and the tree-width. Fix some signature σ and some width $k \in \mathbb{N}$. Let U_k be a set of size $2(k + |\mathrm{const}(\sigma)|)$. We refer to these as "names", as they name elements coded in a node. The signature $\tilde{\sigma}_k$ for the encodings is defined as follows.

- For all $a \in U_k$, there is a unary relation $D_a \in \tilde{\sigma}_k$ which indicates that a is a name for an element in the bag.
- For every relation $R \in \sigma$ of arity n and every n-tuple $\boldsymbol{a} \in U_k^n$, there is a unary relation $R_{\boldsymbol{a}} \in \tilde{\sigma}_k$, which indicates that R holds for the tuple of elements indexed by $\boldsymbol{a}$.
- For every constant $z \in \sigma$ and $c \in U_k$, there are unary relations $V_{c/z}$ which indicate that z is interpreted by the element named by c in the given bag.

Tree decompositions and the corresponding encodings can generally have unbounded (possibly infinite) degree. We apply the first-child, next-sibling transformation (based on an arbitrary ordering of the children) to the standard encoding, so that we can use binary trees for our encodings. This transformation takes a tree with arbitrary branching degree and constructs a binary tree as follows: it maps the root of the original tree to the root of a new binary tree; then, starting from the root, each node's leftmost child in the original tree is mapped to its left child in the binary tree, and its next sibling to the right in the original tree is mapped to its right child.

Hence, for our binary tree encodings, a node u can be identified by a word in $\{0, 1\}^*$, and the *biological children* of u are the nodes $u01^*$ (these are the nodes that would have been children of u in the tree decomposition before the first-child next-sibling transformation). The *biological parent* of $v \neq \epsilon$ is the unique u such that $v \in u01^*$. A *biological neighbor* is a biological child or biological parent. For these binary tree encodings, we also add to $\tilde{\sigma}_k$ unary predicates P_i for $i \in \{0, 1\}$ which indicate the node is the i-th child of its parent.

This transformation to a binary tree encoding is essential in certain automaton constructions (e.g. Theorem 13) when the automaton needs to record in its state the direction it came from. From now on, we use $\tilde{\sigma}_k$-tree to refer to an infinite full binary tree over $\tilde{\sigma}_k$.

Decoding structures If a $\tilde{\sigma}_k$-tree satisfies certain consistency properties, then it can be decoded into a σ-structure that has tree-width $k + |\mathrm{const}(\sigma)| - 1$.

Let $\mathrm{names}(v) := \{a \in U_k : D_a\, v\}$ denote the set of *names* used for elements in bag v.

A *consistent* $\tilde{\sigma}_k$*-tree* is a $\tilde{\sigma}_k$-tree such that every node v satisfies

- $|\mathrm{names}(v)| \leq k + |\mathrm{const}(\sigma)|$;
- for all $R_{\boldsymbol{a}} \in \tilde{\sigma}_k$, if $R_{\boldsymbol{a}}\, v$ then $\boldsymbol{a} \subseteq \mathrm{names}(v)$;
- $P_i v$ holds iff v is the i-th child of its parent;
- for all constants $z \in \sigma$, there is exactly one node w and one $c \in \mathrm{names}(w)$ such that $V_{c/z}\, w$.

Given a consistent tree $\mathcal{T}$, we say nodes u and v are a-connected if there is a sequence of nodes $u = w_0, w_1, \ldots, w_j = v$ such that w_{i+1} is a biological neighbor of w_i, and $a \in \mathrm{names}(w_i)$ for all $i \in \{0, \ldots, j\}$. We write $[v, a]$ for the equivalence class of a-connected nodes of v. For $\boldsymbol{a} = a_1 \ldots a_n$, we often abuse notation and write $[v, \boldsymbol{a}]$ for the tuple $[v, a_1], \ldots, [v, a_n]$

The *decoding* of $\mathcal{T}$ is the σ-structure $\mathfrak{D}(\mathcal{T})$ with universe $\{[v, a] : v \in \mathrm{dom}(\mathcal{T})$ and $a \in \mathrm{names}(v)\}$ such that for each constant

z, we have $z^{\mathfrak{D}(\mathcal{T})} := [v, c]$ for the unique v, c such that $V_{c/z}\, v$ holds, and for each relation R, we have $R^{\mathfrak{D}(\mathcal{T})}([v_1, a_1], \ldots, [v_j, a_j])$ iff there is $w \in \mathrm{dom}(\mathcal{T})$ such that $R_{\boldsymbol{a}} w$ holds and $[w, a_i] = [v_i, a_i]$ for all i.

Free variables For formulas with free variables, the trees are extended with additional information about the valuations for these free variables. These trees use an extended signature where for each free first-order variable z and each $c \in U_k$, we introduce a predicate $V_{c/z}$, and for each second-order variable Z of arity n and each $\boldsymbol{a} \in U_k^n$, we introduce a predicate $Z_{\boldsymbol{a}}$. We refer to these as free variable markers or encodings of valuations for free variables.

We write $z^{\rightarrow}$ for the sets of predicates associated with z. For a set of free variables $\boldsymbol{z} = \{z_1 \ldots z_n\}$, we write $\boldsymbol{z}^{\rightarrow}$ for $z_1{}^{\rightarrow} \cup \cdots \cup z_n{}^{\rightarrow}$. We sometimes abuse notation and write $\boldsymbol{z}^{\rightarrow}$ for both the predicates and the valuation of those predicates. Similar conventions apply to the free second-order variables.

Given a $\tilde{\sigma}_k$-tree $\mathcal{T}$, we write $(\mathcal{T}, \boldsymbol{z}^{\rightarrow}, \boldsymbol{Z}^{\rightarrow})$ for a tree over the signature $\tilde{\sigma}_k \cup \boldsymbol{z}^{\rightarrow} \cup \boldsymbol{Z}^{\rightarrow}$. We abuse the terminology and say $(\mathcal{T}, \boldsymbol{z}^{\rightarrow}, \boldsymbol{Z}^{\rightarrow})$ is a consistent $\tilde{\sigma}_k$-tree if $\mathcal{T}$ is a consistent $\tilde{\sigma}_k$-tree, and for each $z \in \boldsymbol{z}^{\rightarrow}$, there is exactly one v and one $c \in \mathrm{names}(v)$ such that $V_{c/z}\, v$ holds, and for each $Z_{\boldsymbol{a}} \in \boldsymbol{Z}^{\rightarrow}$, if $Z_{\boldsymbol{a}} v$ then $\boldsymbol{a} \subseteq \mathrm{names}(v)$.

5. Automata tools

We make use of automata running on infinite binary trees. In this section, we briefly recall some definitions and properties (please consult, e.g., [24, 30] for more information). We will need to use 2-way automata that can move both up and down as they process the tree, so we highlight some less familiar properties about the relationship between 2-way and 1-way versions of these automata.

Trees The input structures are infinite full binary trees $\mathcal{T}$ over a finite set of propositions Σ. In other words, these are structures over a signature with binary relations for the left and right child relation, and unary relations for the propositions. We also assume there are propositions indicating whether each node is a left child, right child, or the root. We write $\mathcal{T}(v)$ for the set of propositions that hold at node v.

Tree automata An *alternating parity tree automaton* $\mathcal{A}$ is a tuple $\langle \Sigma, Q, q_0, \delta, \Omega \rangle$ where Σ is a finite set of propositions, Q is a finite set of states, $q_0 \in Q$ is the initial state, $\delta : Q \times \mathcal{P}(\Sigma) \to \mathcal{B}^+(\mathrm{Dir} \times Q)$ is the transition function with directions $\mathrm{Dir} \subseteq \{\mathbf{l}, \mathbf{r}, \uparrow\}$, and $\Omega : Q \to P$ is the priority function with a finite set of *priorities* $P \subseteq \mathbb{N}$.

The transition function maps a state and input letter to a positive boolean formula over propositions $\mathrm{Dir} \times Q$, denoted $\mathcal{B}^+(\mathrm{Dir} \times Q)$. This formula indicates possible next moves for the automaton. We can assume that these formulas are written in disjunctive normal form. Running the automaton $\mathcal{A}$ on some input tree $\mathcal{T}$ is best thought of in terms of an *acceptance game*. Positions in the game are of the form $(q, v) \in Q \times \mathrm{dom}(\mathcal{T})$. In position (q, v), Eve chooses a disjunct θ in $\delta(q, \mathcal{T}(v))$. Then Adam chooses a conjunct (d, q') in θ and the game continues from position (q', v'), where v' is the node in direction d from v. In other words, the disjunct θ chosen by Eve specifies the possible copies of the automaton that could be launched by Adam from v or a neighbor of v.

A play $(q_0, v_0)(q_1, v_1) \ldots$ in the game is winning for Eve if it satisfies the *parity condition*: the maximum priority occurring infinitely often in $\Omega(q_0)\Omega(q_1) \ldots$ is even. A *strategy* for Eve is a function that given the history of the play and the current position in the game, determines Eve's choice in the game. Note that we allow the automaton to be started from arbitrary positions v_0 in the tree, rather than just the root; we will often indicate this by saying that the automaton is *launched* from v_0. We say that $\mathcal{A}$ *accepts* $\mathcal{T}$ *starting from* v_0 if Eve has a strategy such that all plays consistent with the strategy starting from (q_0, v_0) are winning. $L(\mathcal{A})$ denotes the *language* of trees accepted by $\mathcal{A}$ starting from the root.

A 1-way alternating automaton is an automaton that uses only directions $\mathbf{l}$ and $\mathbf{r}$. A (1-way) nondeterministic automaton is a 1-way alternating automaton such that every transition function formula is of the form $\bigvee_i (\mathbf{l}, q_{\mathbf{l}}^i) \wedge (\mathbf{r}, q_{\mathbf{r}}^i)$.

Closure properties We recall some closure properties of these automata (omitting the standard proofs).

First, the automata that we are using are closed under union and intersection (of their languages).

Proposition 9. *2-way alternating parity tree automata and 1-way nondeterministic parity tree automata are closed under union and intersection, with only a polynomial blow-up in the number of states and overall size.*

For example, this means that if we are given 2-way alternating parity tree automata $\mathcal{A}_1$ and $\mathcal{A}_2$, then we can construct in PTime a 2-way alternating parity tree automaton $\mathcal{A}$ such that $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$. In automaton constructions, when we say, e.g., "take the intersection of $\mathcal{A}_1$ and $\mathcal{A}_2$", we mean take this automaton $\mathcal{A}$ such that $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.

Another important language operation is projection. Let L' be a language of trees over propositions $\Sigma \cup \{P\}$. The *projection* of L' with respect to P is the language of trees $\mathcal{T}$ over Σ such that there is some $\mathcal{T}' \in L'$ such that $\mathcal{T}$ and $\mathcal{T}'$ agree on all propositions in Σ. Projection is easy for nondeterministic automata since the valuation for the projected proposition can be guessed by Eve.

Proposition 10. *1-way nondeterministic parity tree automata are closed under projection, with no change in the number of states and overall size.*

Finally, complementation is easy for alternating automata by taking the *dual* automaton, obtained by switching conjunctions and disjunctions in the transition function, and incrementing all of the priorities by one.

Proposition 11. *2-way alternating parity tree automata are closed under complementation, with no change in the number of states and overall size.*

Connections between 2-way and 1-way automata It was shown by Vardi [31] that 2-way alternating parity tree automata can be converted to equivalent 1-way nondeterministic automata, with an exponential blow-up.

Theorem 12 ([31])**.** *Let $\mathcal{A}$ be a 2-way alternating parity tree automaton. We can construct a 1-way nondeterministic parity tree automaton $\mathcal{A}'$ such that $L(\mathcal{A}) = L(\mathcal{A}')$. The number of states of $\mathcal{A}'$ is exponential in the number of states of $\mathcal{A}$, but the number of priorities of $\mathcal{A}'$ is linear in the number of priorities of $\mathcal{A}$.*

1-way nondeterministic tree automata can be seen as a special case of 2-way alternating automata, so the previous theorem shows that 1-way nondeterministic and 2-way alternating parity automata are equivalent, in terms of their ability to recognize trees starting from the root.

We need another conversion from 1-way nondeterministic to 2-way alternating automata that we call *localization*. This is the process by which a 1-way nondeterministic automaton that is running on trees with extra information about some predicate annotated on the tree is converted to an equivalent 2-way alternating automaton that operates on trees without these annotations, but under the assumption that these predicates hold only locally at the position the 2-way automaton is launched from. A similar localization idea is present in prior work (see, e.g., [10, 12]).

Theorem 13. *Let $\Sigma' := \Sigma \cup \{P_1, \ldots, P_j\}$. Let $\mathcal{A}'$ be a 1-way nondeterministic parity automaton on Σ'-trees. We can construct a 2-way alternating parity automaton $\mathcal{A}$ on Σ-trees such that for*

all Σ-trees $\mathcal{T}$ and $v \in \mathrm{dom}(\mathcal{T})$,

$$\mathcal{A}' \text{ accepts } \mathcal{T}' \text{ from the root iff } \mathcal{A} \text{ accepts } \mathcal{T} \text{ from } v,$$

where $\mathcal{T}'$ is the Σ'-tree obtained from $\mathcal{T}$ by setting $P_1^{\mathcal{T}'} = \cdots = P_j^{\mathcal{T}'} = \{v\}$. The number of states of $\mathcal{A}$ is linear in the number of states of $\mathcal{A}'$, and the overall size is linear in the size of $\mathcal{A}'$.

Proof sketch. $\mathcal{A}$ simulates $\mathcal{A}'$ by guessing in a backwards fashion an initial part of a run of $\mathcal{A}'$ on the path from v to the root and then processing the rest of the tree in a normal downwards fashion. The subtlety is that the automaton $\mathcal{A}$ is reading a tree without valuations for $P_1, \ldots, P_j$ so once the automaton leaves node v, if it were to cross this position again, it would be unable to correctly simulate $\mathcal{A}'$. To avoid this, we only send downwards copies of the automaton in directions that are not on the path from the root to v. □

We remark that this construction can be adapted for an alternating parity automaton as input, but $\mathcal{A}$ is exponential in the size of the input automaton $\mathcal{A}'$, rather than linear.

Emptiness testing Finally, we make use of the well-known fact that language emptiness of tree automata is decidable.

Theorem 14 ([16],[31])**.** *For 1-way nondeterministic parity tree automata, emptiness is decidable in time polynomial in the number of states and exponential in the number of priorities. For 2-way alternating parity automata, it is decidable in time exponential in the number of states and priorities.*

6. Automata for GNFP-UP

In this section, we construct automata for θ in GNFP-UP[σ]. Before we give some details of the construction, it is helpful to consider how automata can be used to analyze fixpoints.

Using localized automata for fixpoints Testing whether some tuple $\boldsymbol{t}$ is in the least fixpoint $[\mathbf{lfp}^{\boldsymbol{z}}_{Y,\boldsymbol{y}} . \phi]$ in some structure $\mathfrak{N}$ and for some fixed valuation of the parameters (and any other free variables) can be viewed as a game. Positions in this game consist of the current tuple $\boldsymbol{y}$ being tested in the fixpoint, with the initial position being $\boldsymbol{t}$. In general, in position $\boldsymbol{y}$, one round of the game consists of the following:

- Eve chooses some valuation for Y such that $\phi(\boldsymbol{y}, Y)$ holds (if it is not possible, she loses), then
- Adam chooses tuple $\boldsymbol{y}' \in Y$ (if it is not possible, he loses), and the game proceeds to the next round in position $\boldsymbol{y}'$.

Adam wins if the game continues forever.

The idea is that if $\boldsymbol{t}$ is really in the least fixpoint, then it must be added in some fixpoint approximant. This gives Eve a strategy for choosing Y at each stage in the game, in such a way that after finitely many challenges by Adam, she should be able to guess the empty valuation.

When the fixpoint can consist of only guarded tuples, there is a version of this game on a tree encoding $\mathcal{T}$ of a structure, that can be implemented using tree automata. We start with an automaton $\mathcal{A}_\phi$ for the body ϕ of the fixpoint. In fact, we start with *localized versions* of this automaton because we need to launch different versions based on Adam's challenges. A *local assignment* $\boldsymbol{b}/\boldsymbol{y}$ for $\boldsymbol{b} = b_1 \ldots b_n \in U_k^n$ and $\boldsymbol{y} = y_1 \ldots y_n$ is a mapping such that $y_i \mapsto b_i$. A node v in $\mathcal{T}$ with $\boldsymbol{b} \subseteq \mathrm{names}(v)$ and a local assignment $\boldsymbol{b}/\boldsymbol{y}$, specifies a valuation for $\boldsymbol{y}^{\rightarrow}$. We say it is local since the free variable markers for $\boldsymbol{y}$ would all appear locally in v. If we have an automaton $\mathcal{A}$ running on trees with free variable markers for $\boldsymbol{y}$, we say that we *localize* $\mathcal{A}$ *to* $\boldsymbol{b}/\boldsymbol{y}$ if we apply the localization theorem (Theorem 13) to the predicates V_{b_i/y_i}, and then eliminate the dependence on any other V_{c/y_i} for $c \neq b_i$ by always assuming these predicates do not hold. This results in an automaton that simulates $\mathcal{A}$ under the assumption that the free variables $\boldsymbol{y}$ correspond to the elements $[v, \boldsymbol{b}]$, but it no longer relies on free variable markers for $\boldsymbol{y}$. These localized automata are important because they can be launched to test that a tuple of elements that appear together in a node satisfy some property — without having the markers for this tuple explicitly written on the tree.

We now describe the version of the fixpoint game using localized automata. Initially, Eve navigates to a node in $\mathcal{T}$ carrying $\boldsymbol{t}$, and launches the appropriate localized $\mathcal{A}_\phi$ from there. In general, the game proceeds as follows:

- Eve and Adam simulate some localized version of $\mathcal{A}_\phi$. During the simulation Eve can guess a valuation for Y (recall that $\mathcal{A}_\phi$ runs on trees with an annotation describing the valuation for the second-order variable Y, and that information is missing from $\mathcal{T}$). Because Y can only contain guarded tuples, this amounts to guessing an annotation of the tree with this valuation.
- When Eve guesses some $\boldsymbol{y}' \in Y$, Adam can either continue the simulation, or challenge her on this assertion. A challenge corresponds to launching a new localized copy of $\mathcal{A}_\phi$ from the node carrying $\boldsymbol{y}'$ (again, we know that $\boldsymbol{y}'$ must be present locally in a node, since any tuple in the fixpoint must be guarded).

After each challenge, the game continues as before (with the new copy of $\mathcal{A}_\phi$ being simulated, Eve guessing a new valuation for Y, etc.). Adam wins if he challenges infinitely often, or if the game stabilizes in some simulation of $\mathcal{A}_\phi$ where he wins.

Assuming we have localized automata for ϕ, we can implement this game using a 2-way alternating parity automaton. We assign a large odd priority (larger than the priorities in $\mathcal{A}_\phi$) to the states where Adam challenges, so that he wins if he is able to challenge infinitely many times; the other priorities are just inherited from $\mathcal{A}_\phi$. Simultaneous fixpoints can be handled in a similar way.

In order to analyze the fixpoints like this, our inductive automaton construction must produce 2-way localized automata at each stage — if we did not, then each time we reached a fixpoint and needed localized automata for the body of the fixpoint, we would get an exponential blow-up. For GFP and GNFP, we can define directly the localized versions of the automata using a state set of size at most singly exponential in the size of the input formula. However, by adding parameters in GNFP-UP, this direct definition of a localized version becomes more challenging. We are forced to construct non-localized automata at some points — namely, for subformulas that introduce new parameters — and then apply Theorems 12 and 13, resulting in an exponential blow-up. The parameter depth is a measure of how many of these blow-ups occur.

Construction We now describe more details of the construction of an automaton for normal form $\theta \in$ GNFP-UP[σ]. First, it is straightforward to construct an automaton that checks consistency:

Proposition 15. *There is a 2-way alternating parity tree automaton C that checks whether or not a $\tilde{\sigma}_k$-tree (possibly extended with additional free variable markers for $\mathbf{z}$ and $\mathbf{Z}$) is consistent. The size of C is at most exponential in $(|\sigma| + |\mathbf{z}| + |\mathbf{Z}|) \cdot |U_k|^k$.*

Hence, we can concentrate on defining an automaton for θ that runs on consistent trees and accepts iff the decoding of the consistent input tree actually satisfies θ.

The main theorem states that the size of the automaton for θ is a tower of exponentials whose height depends on the pdepth. Given a function f, we write $\exp_f^n(m)$ for a tower of exponentials of height n based on f, i.e. $\exp_f^0(m) = m$ and $\exp_f^n(m) = 2^{f(\exp_f^{n-1}(m))}$.

Theorem 16. *For normal form $\theta \in$ (GNFP-UP)^k[σ] with $pdepth(\theta) \geq 1$, we can construct a 2-way alternating parity tree automaton $\mathcal{A}_\theta$ such that for all consistent $\tilde{\sigma}_k$-trees $\mathcal{T}$, $\mathfrak{D}(\mathcal{T}) \models \theta$ iff $\mathcal{T} \in L(\mathcal{A}_\theta)$, and the size of $\mathcal{A}_\theta$ is at most $(pdepth(\theta)+1)$-exponential in $|\theta|$.*

More precisely, there is a polynomial function f independent of θ such that the size is at most $exp_f^{pdepth(\theta)}(f(m)\cdot 2^{f(klr)})$ where $m = |\theta|$, $l = |\text{const}(\sigma)|$, and $r = \text{rank}_{\text{CQ}}(\sigma)$ (see definitions below).

For brevity, in this theorem and in the remainder of the paper, we usually give only bounds on the output size, not the running time of the algorithms. However the proofs will show that the worst-case running time is bounded by a polynomial in the output size, i.e. the running time of Theorem 16 is (pdepth(θ) + 1)-ExpTime. We emphasize that this means that *for fixed pdepth, the construction can be done in elementary time*.

The main factor affecting the output size is the pdepth, since this determines the height of the tower of exponentials. However, for more precise bounds, the other factors affecting the size are the *size of the formula* θ, the *width* k, the number of *constants* in σ, and the *CQ-rank* (the maximum number of conjuncts ψ_i in any CQ-shaped subformula $\exists \boldsymbol{x}.\ \bigwedge_i \psi_i$ for non-empty $\boldsymbol{x}$).

The proof of Theorem 16 is by induction on $|\theta|$, and constructs localized 2-way automata for subformulas of θ. For GNFP subformulas, it is known from [8] how to construct 2-way automata:[1]

Lemma 17 ([8])**.** *Let $\psi(\boldsymbol{y}, \boldsymbol{Z}) \in \text{GNFP}^k[\sigma']$ in normal form. Then for every local assignment $\boldsymbol{b}/\boldsymbol{y}$, we can construct a 2-way alternating parity tree automaton $\mathcal{A}_\psi^{\boldsymbol{b}/\boldsymbol{y}}$ such that for all consistent $\tilde{\sigma}'_k$-trees $(\mathcal{T}, \boldsymbol{Z}^{\rightarrow})$ and for all nodes $w \in \text{dom}(\mathcal{T})$ with $\boldsymbol{b} \subseteq \text{names}(w)$,*

$$\mathfrak{D}(\mathcal{T}), [w, \boldsymbol{b}], \boldsymbol{Z} \models \psi \text{ iff } \mathcal{A}_\psi^{\boldsymbol{b}/\boldsymbol{y}} \text{ accepts } (\mathcal{T}, \boldsymbol{Z}^{\rightarrow}) \text{ from } w.$$

There is a polynomial function f independent of ψ such that the number of states for all such localized automata is at most $N := f(m)\cdot 2^{f(klr)}$ where $m = |\psi|$, $l = |\text{const}(\sigma')|$, and $r = \text{rank}_{\text{CQ}}(\psi)$. The number of priorities in each automaton is linear in $|\psi|$. The overall size is at most exponential in $|\sigma'|\cdot N$.

We use these automata for GNFP as building blocks for our GNFP-UP construction. Recall that pdepth 0 formulas can always be viewed as GNFP formulas. We can also transform parts of the formula into GNFP formulas over a slightly different signature.

For this purpose, given $\psi \in (\text{GNFP-UP})^k[\sigma]$ with params(ψ) $\subseteq \boldsymbol{z}$, define the *augmented signature* $\sigma_{\boldsymbol{z},\psi}$ to be the signature σ together with additional constants $z \in \boldsymbol{z}$ and subformula predicates F_η for subformulas η with params(η) $\subseteq \boldsymbol{z}$. For such η, the arity of F_η is usually $|\text{free}(\eta) \setminus \text{params}(\eta)|$; in the special case that η is a fixpoint formula, then the arity of F_η is the arity of this fixpoint predicate. Then we can transform the outer part of a GNFP-UP formula to a GNFP formula over this augmented signature. We can only perform this transformation on the outer part of the formula that uses the same set of parameters. Consider $\eta \in (\text{GNFP-UP})^k[\sigma]$ with free(η) $\subseteq \boldsymbol{yz}$ and params(η) $\subseteq \boldsymbol{z}$. We define $\text{transform}_{\boldsymbol{z}}(\eta) \in \text{GNFP}^k[\sigma_{\boldsymbol{z},\eta}]$ inductively as follows:

$$\text{transform}_{\boldsymbol{z}}(R\,\boldsymbol{t}) := R\,\boldsymbol{t} \qquad \text{transform}_{\boldsymbol{z}}(Y\,\boldsymbol{t}) := Y\,\boldsymbol{t}$$

$$\text{transform}_{\boldsymbol{z}}(\alpha \wedge \neg\phi) := \alpha \wedge \neg\text{transform}_{\boldsymbol{z}\cap\text{free}(\phi)}(\phi)$$

$$\text{transform}_{\boldsymbol{z}}([\mathbf{lfp}^{\boldsymbol{z}'}_{X,\boldsymbol{x}}.S](\boldsymbol{t})) :=$$

$$\begin{cases} F_{[\mathbf{lfp}^{\boldsymbol{z}'}_{X,\boldsymbol{x}}.S](\boldsymbol{t})}\,\boldsymbol{t} \text{ if there is } \phi_j \in S \text{ with params}(\phi_j) \not\subseteq \boldsymbol{z} \\ [\mathbf{lfp}_{X,\boldsymbol{x}}.S'](\boldsymbol{t}) \text{ o.w.} \end{cases}$$

where S' is the result of applying $\text{transform}_{\boldsymbol{z}}$ to each $\phi_j \in S$

$$\text{transform}_{\boldsymbol{z}}(\textstyle\bigvee_i \exists \boldsymbol{x}_i.\ \bigwedge_j \psi_{ij}) :=$$

$$\begin{cases} F_{\bigvee_i \exists \boldsymbol{x}_i.\ \bigwedge_j \psi_{ij}}\,\boldsymbol{y} \text{ if there is } i, j \text{ such that params}(\psi_{ij}) \not\subseteq \boldsymbol{z} \\ \bigvee_i \exists \boldsymbol{x}_i.\ \bigwedge_j \text{transform}_{\boldsymbol{z}\cap\text{free}(\psi_{ij})}(\psi_{ij}) \text{ o.w.} \end{cases}$$

[1] [8] used a different encoding of the tree-like models, but the adaptation to the encoding here requires only minor technical changes.

The GNFP formula obtained using this transformation is "equivalent" to the GNFP-UP formula, under the assumption that the additional predicates in the augmented signature are interpreted in the expected way. It does not increase the width, CQ-rank, or the size of the formula.

If the transformation applied to η only introduces $F_{\eta'}$ for strict subformulas η' of η, then we say the transformation is *helpful*. In a helpful transformation, all occurrences of these new predicates $F_{\eta'}$ appear under a guard of free(η') $\setminus$ params(η'). Another way to understand the parameter depth is to say that the parameter depth measures the number of unhelpful breakpoints we reach as we try to transform the entire formula using this operation.

The main idea in the construction, described in Lemma 18 below, is to transform the outer part of the formula into a GNFP formula. If the transformation is helpful, we can then use the GNFP automaton for the outer part of the formula, and plug in inductively defined automata checking the subformulas. When this is not possible, we must use different techniques which result in an exponential blow-up at these stages.

Lemma 18. *Let $\phi(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{Z})$ be a subformula of $\theta \in (\text{GNFP-UP})^k[\sigma]$ with params(ϕ) $\subseteq \boldsymbol{z}$. For each local assignment $\boldsymbol{b}/\boldsymbol{y}$, we can construct a 2-way alternating parity tree automaton $\mathcal{B}_\phi^{\boldsymbol{b}/\boldsymbol{y}}$ such that for all consistent $\tilde{\sigma}_k$-trees $(\mathcal{T}, \boldsymbol{z}^{\rightarrow}, \boldsymbol{Z}^{\rightarrow})$ and for all nodes $w \in \text{dom}(\mathcal{T})$ with $\boldsymbol{b} \subseteq \text{names}(w)$,*

$$\mathfrak{D}(\mathcal{T}), [w, \boldsymbol{b}], \boldsymbol{z}, \boldsymbol{Z} \models \phi \text{ iff } \mathcal{B}_\phi^{\boldsymbol{b}/\boldsymbol{y}} \text{ accepts } (\mathcal{T}, \boldsymbol{z}^{\rightarrow}, \boldsymbol{Z}^{\rightarrow}) \text{ from } w.$$

For $\text{pdepth}_{\boldsymbol{z}}(\phi) \geq 1$, there is a polynomial function f independent of ϕ such that the size of all such localized automata is at most $exp_f^{\text{pdepth}_{\boldsymbol{z}}(\phi)}(f(mn)\cdot 2^{f(klr)})$ where $m = |\phi|$, $n = |\sigma|$, $l = |\text{const}(\sigma)|$, and $r = \text{rank}_{\text{CQ}}(\phi)$. The number of priorities is linear in $|\phi|$. For $\text{pdepth}_{\boldsymbol{z}}(\phi) = 0$, the bounds match Lemma 17.

Proof sketch. The proof is by induction on $|\phi|$.

Assume $\phi' := \text{transform}_{\boldsymbol{z}}(\phi)$ is helpful. This always holds for the smallest (atomic) formulas, so this covers the base of the induction. We construct $\mathcal{B}_\phi^{\boldsymbol{b}/\boldsymbol{y}}$ to simulate the automaton $\mathcal{A}_{\phi'}^{\boldsymbol{b}/\boldsymbol{y}}$ from Lemma 17, while allowing Eve to guess valuations for the F_η relations from ϕ'. Since ϕ' is helpful, we know that every F_η relation in ϕ' is for some formula η that is strictly smaller than ϕ, and hence the inductive hypothesis ensures there is a corresponding automaton for every suitable local assignment. During the simulation of $\mathcal{A}_{\phi'}^{\boldsymbol{b}/\boldsymbol{y}}$, if Eve asserts $F_{\eta(\boldsymbol{x})}\,\boldsymbol{a}$ at w for some $\boldsymbol{a} \subseteq \text{names}(w)$, then Adam can challenge this by launching the localized automaton for the intersection of $\mathcal{B}_{\eta(\boldsymbol{x})}^{\boldsymbol{a}/\boldsymbol{x}}$ and $\mathcal{B}_{\text{gdd}(\boldsymbol{x})}^{\boldsymbol{a}/\boldsymbol{x}}$ from w; likewise, if Eve does not assert $F_{\eta(\boldsymbol{x})}\,\boldsymbol{a}$ at w for some $\boldsymbol{a} \subseteq \text{names}(w)$, then Adam can challenge this by launching the localized automaton for the dual of the intersection of $\mathcal{B}_{\eta(\boldsymbol{x})}^{\boldsymbol{a}/\boldsymbol{x}}$ and $\mathcal{B}_{\text{gdd}(\boldsymbol{x})}^{\boldsymbol{a}/\boldsymbol{x}}$ from w. Correctness follows from Lemma 17, and the fact that the inductive hypothesis ensures the subautomata for F_η in ϕ' are correct. There is no exponential blow-up in this case.

Next, assume that $\phi' := \text{transform}_{\boldsymbol{z}}(\phi)$ is unhelpful. There are two possible cases.

The first case is when ϕ is a UCQ-shaped formula $\bigvee_i \exists \boldsymbol{x}_i.\ \bigwedge_j \psi_{ij}$ where variables from some $\boldsymbol{x}_i$ are used as parameters in $\bigwedge_j \psi_{ij}$. To start, we consider each CQ-shaped formula separately, so fix some $\exists \boldsymbol{x}_i.\ \bigwedge_j \psi_{ij}$. We use the inductive hypothesis to obtain 2-way alternating automata for each conjunct ψ_{ij}, using the empty local assignment. This is possible since the size of these conjuncts must be strictly less than the size of ϕ. Because we are using the empty local assignment, these automata operate on trees with markers for all of the free variables: $\boldsymbol{x}_i \cup \boldsymbol{y} \cup \boldsymbol{z}$. Take the intersection of these automata using Proposition 9. Then take the intersection with the automaton from Proposition 15 checking consistency. This yields a 2-way alternating automaton corresponding to $\bigwedge_j \psi_{ij}$. We then convert this

automaton to a nondeterministic version using Theorem 12 and project away the information about $\boldsymbol{x}_i$ using Proposition 10. This yields an equivalent (1-way) nondeterministic parity automaton for the CQ. Next, we localize this automaton to the desired variables $\boldsymbol{y}$ using Theorem 13. Finally, to construct the automaton $\mathcal{B}_{\phi}^{\boldsymbol{b}/\boldsymbol{y}}$, we take the union of the individual CQ automata using Proposition 9. The conversion from a 2-way alternating automaton to a nondeterministic automaton is the costly step in this process, resulting in an exponential blow-up. This matches the claimed size bound since $\mathrm{pdepth}(\psi_{ij}) < \mathrm{pdepth}_{\boldsymbol{z}}(\phi)$.

The second case is when ϕ is a fixpoint formula where fixpoint variables are used as parameters in the body of the fixpoint. Suppose it is of the form $[\mathbf{lfp}^{\boldsymbol{z}'}_{X,\boldsymbol{x}} . \mathrm{gdd}(\boldsymbol{x}) \wedge \chi(\boldsymbol{x}\boldsymbol{z}', X\boldsymbol{Z})](\boldsymbol{t})$ where $\mathrm{params}(\chi) \cap \boldsymbol{x} \neq \emptyset$, so $\mathrm{params}(\chi) \not\subseteq \boldsymbol{z}$; the construction is similar for a simultaneous fixpoint. The formula χ in the body of the fixpoint is strictly smaller, so we can apply the inductive hypothesis to get an automaton for this part. We want this automaton to be localized to $\boldsymbol{x}$ so we can test for tuples in the fixpoint by launching copies of the automaton for some local assignment. However, the inductive hypothesis does not directly yield this, since some variables from $\boldsymbol{x}$ are used as parameters. Hence, we must use the inductive hypothesis to get a 2-way automaton for $\mathcal{B}_{\chi'}^{\emptyset/\emptyset}$, apply Theorem 12 to get an equivalent 1-way nondeterministic automaton (resulting in an exponential blow-up), and then localize to some $\boldsymbol{a}/\boldsymbol{x}$ using Theorem 13. Once we have these localized automata $\mathcal{B}_{\chi}^{\boldsymbol{a}/\boldsymbol{x}}$ for the body of the fixpoint, we can construct an automaton that captures the fixpoint game described at the beginning of this section. □

Theorem 16 easily follows from this lemma.

Using exactly the same conversion rules from [5], it can be shown that an arbitrary GNFP-UP sentence θ' can be converted to an equivalent normal form θ with size exponential in $|\theta|$, but width and CQ-rank linear in $|\theta|$. Hence, we have the following corollary.

Corollary 19. *For $\theta' \in$ GNFP-UP$[\sigma]$ (not necessarily in normal form), we can construct an automaton $\mathcal{A}_{\theta'}$ of $(pdepth(\theta') + 1)$-exponential size.*

7. Satisfiability and containment

Because of the tree-like model property for GNFP-UP, we can use the automaton construction and ExpTime emptiness testing (Theorem 14) to decide satisfiability of GNFP-UP.

Theorem 20. *Satisfiability for $\theta \in$ GNFP-UP is decidable in $(pdepth(\theta) + 2)$-ExpTime.*

Applications We can apply Theorem 20 to obtain results about the query languages described in Section 3: e.g., containment of m-nested MQs is in $(m + 2)$-ExpTime. This result was known already from [12]. However, an advantage of this GNFP-UP framework is that we can introduce additional features such as relativizing the results to sentences in our logics, without affecting the complexity.

Corollary 21. *Containment of m-nested* MQ*s relative to boolean frontier-guarded Datalog queries (or any boolean queries translatable to* GNFP *in* PTime*) is decidable in $(m + 2)$-*ExpTime.

We can extend our approach to even allow some unguarded logics on the left-hand side of the containment (as is done in [12], which considers only unrelativized containment).

Corollary 22. *Containment of a Datalog query in an m-nested* MQ *is in $(m + 2)$-*ExpTime*, even relative to a boolean frontier-guarded Datalog query or a* GNFP *sentence.*

We can also derive results about satisfiability of CQPDL with respect to GNFP sentences. Although CQPDL can reach arbitrary pdepth levels, we can construct an equi-satisfiable formula of low pdepth. This yields the following new result, extending results about (nested) C2RPQs from [9].

Corollary 23. *Satisfiability of* CQPDL *sentences (or boolean combinations of* CQPDL *sentences and* GNFP *sentences) is decidable in* 2-ExpTime.

Lower bounds This connection with flag and check queries also demonstrates that our general $(m + 2)$-ExpTime bound on satisfiability for GNFP-UP formulas of pdepth m is optimal, since containment of boolean Datalog queries in m-nested MQs is actually $(m + 2)$-ExpTime hard [12].

8. Extending to boundedness

Thus far, we have concentrated on showing that satisfiability is decidable for GNFP-UP, using techniques based on automata. In this section, we point out that we can extend this automata machinery to help answer additional questions like boundedness.

The *boundedness problem* for a logic $\mathcal{L}$ over σ asks:

> Given a formula $\phi(\boldsymbol{x}, X) \in \mathcal{L}[\sigma]$ positive in some second-order variable X of arity $|\boldsymbol{x}|$, is there a natural number n such that for all σ-structures $\mathfrak{A}$, $\phi^n(\mathfrak{A}) = \phi^{n+1}(\mathfrak{A})$?

In other words, the boundedness problem asks whether there is a uniform natural number bound on the number of iterations needed to reach the least fixpoint induced by $\phi(\boldsymbol{x}, X)$. For formulas $\phi(\boldsymbol{x}, \boldsymbol{z}, X)$ with some designated set of parameters $\boldsymbol{z}$, it is also natural to ask the following variant of the boundedness problem: is there a natural number n such that for all σ-structures $\mathfrak{A}$ and for all valuations $\boldsymbol{z} \mapsto \boldsymbol{c}$, does $\phi^n(\mathfrak{A}, \boldsymbol{c}) = \phi^{n+1}(\mathfrak{A}, \boldsymbol{c})$?

There is a large body of work studying boundedness for various logics, particularly Datalog queries (see [1, 15, 22]). Boundedness is undecidable when ϕ is a negation-free first-order formula. However, for many guarded logics, boundedness has been shown to be decidable [6, 8, 11] (some of the decidability results rely on unpublished work due to Colcombet, referred to as ILT in [6]).

Using similar techniques, we can analyze boundedness for GNFP-UP. As was the case for our analysis of satisfiability, the key is to take advantage of the fact that we can restrict to structures of bounded tree-width, and then use tree automata to help solve the problem. For boundedness a variant of the prior construction (with the same bounds) can generate a special type of tree automaton with counters called a *cost automaton* (see [8, 14] for more information). This cost automaton defines a function that maps a consistent tree $\mathcal{T}$ to the least $n \in \mathbb{N} \cup \{\infty\}$ such that every tuple in $\phi^{\infty}(\mathfrak{D}(\mathcal{T}))$ is in $\phi^n(\mathfrak{D}(\mathcal{T}))$ — there is a single counter that is incremented each time the fixpoint is unfolded. The range of the function defined by this cost automaton is bounded by a natural number across all consistent trees iff ϕ is bounded.

In general, it is not known how to decide if the range of the function defined by a cost automaton over infinite trees is bounded. However, for special types of cost automata — like the automata that come from analyzing boundedness for guarded logics — this is known to be decidable [8] (in cases where there is a reliance on cost automaton results that have been claimed before but not published, we mark this with ILT as in [6]).[2] Hence, by combining our automaton construction with results from [8] and ILT, we can show:

Theorem 24. *Assuming ILT, the boundedness problem is decidable for $\boldsymbol{x}$-guarded $\phi(\boldsymbol{x}, X) \in$ GNFP-UP$[\sigma]$ in non-elementary time (elementary time for fixed pdepth).*

In the special case of $\boldsymbol{x}$-guarded $\phi(\boldsymbol{x}, \boldsymbol{z}, X) \in$ GNF$[\sigma]$, boundedness is decidable in elementary time (without assuming ILT).

[2] ILT stands for "infinite limitedness theorem", a statement about cost automata on infinite trees. More details can be found in [8].

FO *definability* For certain logics, boundedness coincides with FO definability; e.g., we have the following corollary of Theorem 24.

Corollary 25. *It is decidable whether* $[\mathbf{lfp}^{\boldsymbol{z}}_{X,\boldsymbol{x}}.\phi]$ *can be written in* FO *for* $\boldsymbol{x}$*-guarded* $\phi(\boldsymbol{x},\boldsymbol{z},X) \in \text{GNF}[\sigma]$.

Furthermore, whenever this holds the fixpoint $[\mathbf{lfp}^{\boldsymbol{z}}_{X,\boldsymbol{x}}.\phi]$ *can in fact be written in* GNF.

Proof. If $\phi \in$ GNF is bounded, then the fixpoint is equivalent to the n-th approximant, for some $n \in \mathbb{N}$. We can write out the formula ϕ^n for this approximant, where $\phi^0 := \bot$, and $\phi^n := \phi[\phi^{n-1}(\boldsymbol{y})/X\boldsymbol{y}]$. This is in GNF, and witnesses the FO definability of $[\mathbf{lfp}^{\boldsymbol{z}}_{X,\boldsymbol{x}}.\phi]$. The other direction follows from the Barwise-Moschovakis theorem [7]. □

A similar result is not known for $\phi \in$ GNFP-UP$[\sigma]$, because of the presence of additional fixpoints in ϕ — the boundedness problem only concerns the elimination of the outermost fixpoint.

As another application, we can combine automata techniques from this paper with other cost automata results in [8, 10] to prove the following result about negation-free CQPDL and FO definability.

Theorem 26. *It is decidable whether or not a negation-free* CQPDL *sentence relative to a* GNF *sentence can be expressed in* FO*.*

In particular, we can decide whether a conjunctive regular path query can be expressed in FO*, and similarly for their nested and two-way variants (e.g.* C2RPQ*s).*

The ability to reduce C2RPQ querying to first-order querying is interesting, since when this occurs we can use standard database techniques to evaluate graph queries with recursion.

9. Conclusion

We have explained how fixpoint logics can be increased in expressivity while retaining decidability, by allowing *unguarded parameters*. We have also undertaken a fine-grained analysis of the complexity of static analysis problems for the resulting logics.

A limitation of our analysis is that it restricts to reasoning over *all* structures, both finite and infinite. In contrast, [4, 29] have shown that unparameterized guarded fixpoint logics have decidable satisfiability problems over finite structures. It is not clear if the technique of [4] extends to deal with unguarded parameters.

The results about testing first-order definability of fixpoint logics and recursive queries (e.g. Theorem 26) do not include complexity bounds. We conjecture that the cost automaton results could be analyzed and refined further (as was done in [8]) to extract at least an elementary bound (and, ideally, a 2-ExpTime bound). For cases without negation, like FO definability of C2RPQs, it may well be possible to extend the more elementary automata-theoretic approach used to decide boundedness for monadic Datalog [15], avoiding the use of cost automata altogether.

Acknowledgments

Benedikt's work was sponsored by the Engineering and Physical Sciences Research Council of the United Kingdom (EPSRC), grants EP/M005852/1 and EP/L012138/1. Vanden Boom was partially supported by EPSRC grant EP/L012138/1. Bourhis was supported by CPER Nord-Pas de Calais/FEDER DATA Advanced Data Science and Technologies 2015-2020 and the ANR Aggreg Project ANR-14-CE25-0017, INRIA Northern European Associate Team Integrated Linked Data.

References

[1] S. Abiteboul. Boundedness is undecidable for datalog programs with a single recursive rule. *IPL*, 32(6):281–287, 1989.

[2] H. Andréka, I. Németi, and J. van Benthem. Modal languages and bounded fragments of predicate logic. *JPL*, 27(3):217–274, 1998.

[3] A. Arnold and D. Niwiński. *Rudiments of mu-calculus*. North Holland, 2001.

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

[5] V. Bárány, B. ten Cate, and L. Segoufin. Guarded negation. In *ICALP*, 2011.

[6] V. Bárány, B. ten Cate, and M. Otto. Queries with guarded negation. *PVLDB*, 5(11):1328–1339, 2012.

[7] J. Barwise and Y. N. Moschovakis. Global inductive definability. *JSL*, 43(3):521–534, 1978.

[8] M. Benedikt, B. ten Cate, T. Colcombet, and M. Vanden Boom. The complexity of boundedness for guarded logics. In *LICS*, 2015.

[9] M. Bienvenu, D. Calvanese, M. Ortiz, and M. Simkus. Nested regular path queries in description logics. In *KR*, 2014.

[10] A. Blumensath, T. Colcombet, D. Kuperberg, P. Parys, and M. Vanden Boom. Two-way cost automata and cost logics over infinite trees. In *CSL-LICS*, 2014.

[11] A. Blumensath, M. Otto, and M. Weyer. Decidability results for the boundedness problem. *LMCS*, 10(3), 2014.

[12] P. Bourhis, M. Krötzsch, and S. Rudolph. Reasonable highly expressive query languages. In *IJCAI*, 2015.

[13] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Containment of conjunctive regular path queries with inverse. In *KR*, 2000.

[14] T. Colcombet and C. Löding. Regular cost functions over finite trees. In *LICS*, 2010.

[15] S. S. Cosmadakis, H. Gaifman, P. C. Kanellakis, and M. Y. Vardi. Decidable optimization problems for database logic programs. In *STOC*, 1988.

[16] E. A. Emerson and C. S. Jutla. The complexity of tree automata and logics of programs (extended abstract). In *FOCS*, 1988.

[17] E. Grädel. On the restraining power of guards. *JSL*, 64(4):1719–1742, 1999.

[18] E. Grädel. Guarded fixed point logics and the monadic theory of countable trees. *TCS*, 288(1):129 – 152, 2002.

[19] E. Grädel and I. Walukiewicz. Guarded fixed point logic. In *LICS*, 1999.

[20] E. Grädel, M. Otto, and E. Rosen. Undecidability results on two-variable logics. In *STACS*, 1997.

[21] E. Grädel, C. Hirsch, and M. Otto. Back and forth between guarded and modal logics. *ACM TOCL*, 3(3):418–463, 2002.

[22] G. G. Hillebrand, P. C. Kanellakis, H. G. Mairson, and M. Y. Vardi. Undecidable boundedness problems for datalog programs. *J. Log. Program.*, 25(2):163–190, 1995.

[23] N. Immerman. Relational queries computable in polynomial time. *Information and Control*, 68(1-3):86–104, 1986.

[24] C. Löding. Automata on infinite trees. Available at http://www.automata.rwth-aachen.de/ loeding/inf-tree-automata.pdf.

[25] M. Otto. The boundedness problem for monadic universal first-order logic. In *LICS*, 2006.

[26] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141:1–35, 1969.

[27] S. Rudolph and M. Krötzsch. Flag & check: data access with monadically defined queries. In *PODS*, 2013.

[28] D. Scott. A decision method for validity of sentences in two variables. *JSL*, 27(477), 1962.

[29] L. Segoufin and B. ten Cate. Unary negation. *LMCS*, 9(3), 2013.

[30] W. Thomas. Languages, Automata, and Logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*. 1997.

[31] M. Y. Vardi. Reasoning about the past with two-way automata. In *ICALP*, 1998.

A. Missing proofs from Section 2

A.1 Proof of Proposition 2 (GNFP-UP strictly more expressive than GNFP)

We prove:

> GNFP-UP is strictly more expressive than GNFP, even over finite structures. In particular, it cannot express the transitive closure of a binary relation R.

Proof. Let $\sigma = \{Q, R\}$. Consider the UNFP-UP$[\sigma]$ sentence

$$\phi := \neg\exists x.\left(Qx \wedge \neg[\mathbf{lfp}^{x}_{Y,y}.Ryx \vee \exists y'.(Ryy' \wedge Yy')](x)\right)$$

which expresses that there is an R-loop from every Q-labelled element. Note that $[\mathbf{lfp}^{x}_{Y,y}.Ryx \vee \exists y'.(Ryy' \wedge Yy')]$ defines the transitive closure of the binary relation R.

We claim this cannot be expressed in GNFP, and hence the transitive closure of R is not expressible in GNFP.

Assume there were a GNFP sentence ψ of width k (when in normal form) equivalent to ϕ. Let $\mathfrak{A}$ be the structure with a single R-loop of length $k+1$, with every element in the loop satisfying Q. 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. 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, but $\mathfrak{B}$ does not because the second element is not part of an R-cycle.

We now define a winning strategy for Duplicator in the k-width all-at-once game between $\mathfrak{A}$ and $\mathfrak{B}$. This is the bisimulation game associated with normal form GNFP^k from [5]. 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 $(\boldsymbol{m}, \boldsymbol{n})$, such that $\boldsymbol{m} \mapsto \boldsymbol{n}$ 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 $(\boldsymbol{m}, \boldsymbol{n})$ 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 $\boldsymbol{m} \mapsto \boldsymbol{n}$. From a position $h : X \to Y$ Spoiler chooses a guarded tuple $\boldsymbol{m}$ inside X taking the play to the position $(\boldsymbol{m}, h(\boldsymbol{m}))$. Similarly, from a position $g : Y \to X$ Spoiler chooses a guarded tuple $\boldsymbol{n}$ in Y taking the play to the position $(g(\boldsymbol{n}), \boldsymbol{n})$. 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.

A winning strategy for Duplicator implies agreement between the structures on GNFP^k sentences, so proving the existence of a winning strategy for Duplicator in the game between the structures $\mathfrak{A}$ and $\mathfrak{B}$ defined above will yield the desired contradiction.

Duplicator's winning 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. □

A.2 Proof of Proposition 3 (GFP-UP is not more expressive than GFP)

Recall the statement:

> For every answer-guarded GFP-UP$[\sigma]$ formula ϕ, we can construct an equivalent GFP$[\sigma]$ formula ϕ' using fixpoint predicates of higher arity such that $|\phi'|$ is linear in $|\phi|$.

Proof. Consider an answer-guarded GFP-UP formula

$$\mathrm{gdd}(\boldsymbol{xz}) \wedge [\mathbf{lfp}^{z}_{Y,\boldsymbol{y}}.\,\mathrm{gdd}(\boldsymbol{y}) \wedge \psi(\boldsymbol{yz})](\boldsymbol{x}).$$

We claim that we can replace this with

$$\mathrm{gdd}(\boldsymbol{xz}) \wedge [\mathbf{lfp}_{Y',\boldsymbol{yz}}.\,\mathrm{gdd}(\boldsymbol{yz}) \wedge \psi'(\boldsymbol{yz})](\boldsymbol{xz})$$

where ψ' is the result of replacing $Y\boldsymbol{w}$ with $Y'\boldsymbol{wz}$ in ψ.

Fix some structure $\mathfrak{A}$, and a valuation $z \mapsto \boldsymbol{c}$. We need to argue that although there may be tuples $\boldsymbol{b}$ in the fixpoint such that $\boldsymbol{bc}$ is not guarded, for any $\boldsymbol{a}$ in the fixpoint for ψ such that $\boldsymbol{ac}$ is guarded, we can witness this in the fixpoint only using other tuples that are also guarded with $\boldsymbol{c}$. We define an alternative fixpoint semantics $\hat{\psi}$ based on this idea:

$$\begin{aligned}\hat{\psi}^0(\mathfrak{A}) &:= \emptyset\\ \hat{\psi}^{\beta+1}(\mathfrak{A}) &:= \hat{\psi}_{\mathfrak{A}}(\hat{\psi}^{\beta}(\mathfrak{A}))\\ \hat{\psi}^{\beta}(\mathfrak{A}) &:= \bigcup_{\beta'<\beta}\hat{\psi}^{\beta'}(\mathfrak{A}) \text{ for a limit ordinal } \beta\end{aligned}$$

where $\hat{\psi}_{\mathfrak{A}}(V) := \{\boldsymbol{a} : \boldsymbol{ac} \text{ is guarded and } \mathfrak{A}, V \models \psi(\boldsymbol{ac}, Y)\}$. We can prove by induction on the fixpoint approximant β, that for all $\boldsymbol{a}$ such that $\boldsymbol{ac}$ is guarded, $\boldsymbol{a} \in \psi^{\beta}(\mathfrak{A})$ iff $\boldsymbol{a} \in \hat{\psi}^{\beta}(\mathfrak{A})$. The base and limit ordinal cases of the induction are trivial. For a successor ordinal $\beta+1$, consider some $\boldsymbol{a} \in \psi^{\beta+1}(\mathfrak{A})$ such that $\boldsymbol{ac}$ is guarded. Then $\mathfrak{A} \models \psi(\boldsymbol{ac}, \psi^{\beta}(\mathfrak{A}))$. Consider occurrences of $Y\boldsymbol{w}$ in ψ. If $\boldsymbol{w}$ contains only variables from $\boldsymbol{xz}$, then this tuple must be guarded by assumption on $\boldsymbol{ac}$, so we can replace $Y\boldsymbol{w}$ with $\mathrm{gdd}(\boldsymbol{wz}) \wedge Y\boldsymbol{w}$ in ψ and still have a logically equivalent formula. If $\boldsymbol{w}$ contains variables outside of $\boldsymbol{xz}$, then $Y\boldsymbol{w}$ must occur under some quantification that guards $\boldsymbol{wz}$, so we can replace $Y\boldsymbol{w}$ with $\mathrm{gdd}(\boldsymbol{wz}) \wedge Y\boldsymbol{w}$ and still have a logically equivalent formula. In other words, $\boldsymbol{a}$ being in the $(\beta+1)$-approximant of the fixpoint only depends on tuples in the β-approximant that are guarded with $\boldsymbol{c}$. Overall, this means that $\mathfrak{A} \models \psi(\boldsymbol{ac}, \hat{\psi}^{\beta}(\mathfrak{A}))$, so $\boldsymbol{a} \in \hat{\psi}^{\beta+1}(\mathfrak{A})$ as desired. The other direction is trivial. This argument justifies the substitution described in the first paragraph.

In an answer-guarded GFP-UP formula, all fixpoints must be answer-guarded. Hence, we can apply the substitution described in the first paragraph to all fixpoints to yield an equivalent GFP formula, with fixpoints of possibly higher arity. □

B. Missing proofs from Section 4

B.1 Proof of Proposition 6 (Tree-like model property for GNFP-UP)

We prove:

> Every satisfiable (GNFP-UP)$^k[\sigma]$ sentence has a model of tree-width at most $k + |\text{const}(\sigma)| - 1$.

Bisimulation game In order to prove the tree-like model property, we begin by defining a GN-bisimulation game between structures $\mathfrak{A}$ and $\mathfrak{B}$, that captures when two structures agree on certain guarded negation formulas.

We write $f : \boldsymbol{a} \mapsto \boldsymbol{b}$ to indicate a partial map f from $\mathfrak{A}$ to $\mathfrak{B}$ with domain set($\boldsymbol{a}$), range set($\boldsymbol{b}$), and $f(a_i) = b_i$, where set($\boldsymbol{c}$) denotes the set of elements in the tuple $\boldsymbol{a}$.

A position in the GN$^k[\sigma]$ *bisimulation game* between $\mathfrak{A}$ and $\mathfrak{B}$ (relative to some signature σ, possibly with constants) is a partial map $f : \boldsymbol{a} \mapsto \boldsymbol{b}$ or $f : \boldsymbol{b} \mapsto \boldsymbol{a}$ such that $|\text{dom}(f)| \leq k + |\text{const}(\sigma)|$. dom($f$) must include the interpretation of the constants in σ. Moreover, we require that f is a *partial rigid homomorphism*: that is, f is a partial homomorphism and for any guarded tuple $\boldsymbol{c}$ in dom(f), $f\restriction_{\boldsymbol{c}}$ (the restriction of f to set($\boldsymbol{c}$)) is a partial isomorphism. The structure containing dom(f) is called the *active structure*.

Starting in position f, one round of the game consists of the following:

Restrict Spoiler can choose some tuple $\boldsymbol{c}$ such that set($\boldsymbol{c}$) $\subseteq$ dom(f) contains the interpretation of all constants, and then the game proceeds from position $f\restriction_{\boldsymbol{c}}$; or

Switch if dom(f) is a guarded set, then Spoiler can choose to switch structures, and the game proceeds from position f^{-1}; or

Add if $|\text{dom}(f)| < k + |\text{const}(\sigma)|$, Spoiler can select an element c such that c is in the active structure, Duplicator must choose d in the inactive structure such that $f[c \mapsto d]$ is a partial rigid homomorphism, and then the game proceeds from position $f[c \mapsto d]$. In the special case that $c \in \text{dom}(f)$, then d must be $f(c)$.

Spoiler wins if Duplicator gets stuck (i.e. Spoiler chooses some c but Duplicator is unable to select d such that $f[c \mapsto d]$ is a partial rigid homomorphism). If the game continues forever, then Duplicator wins.

We say $\mathfrak{A}$ and $\mathfrak{B}$ are GN$^k[\sigma]$ *bisimilar* if Duplicator has a winning strategy in the GN$^k[\sigma]$ bisimulation game starting from the empty position.

Infinitary logic A winning strategy for Duplicator implies agreement between $\mathfrak{A}$ and $\mathfrak{B}$ on certain guarded negation formulas in GNF$_\infty$. Formulas in GNF$_\infty$ are built up from atoms using conjunction, guarded negation, and infinitary disjunction $\bigvee \Phi$ (for Φ a possibly infinite set of formulas). This is like GNFP-UP, except the least fixpoint case is replaced by infinitary disjunction. We write GNF$^k_\infty$ for the k-width fragment (i.e. the maximum number of free variables in any subformula is at most k).

Proposition 27. *Assume Duplicator has a winning strategy in the* GN$^k[\sigma]$*-bisimulation game between* $\mathfrak{A}$ *and* $\mathfrak{B}$ *starting from position* $\boldsymbol{a} \mapsto \boldsymbol{b}$*. If* $\phi(\boldsymbol{x}) \in \text{GNF}^k_\infty[\sigma]$ *and* $\mathfrak{A}, \boldsymbol{a} \models \phi(\boldsymbol{x})$*, then* $\mathfrak{B}, \boldsymbol{b} \models \phi(\boldsymbol{x})$*.*

Proof. The proof is a straightforward induction on the structure of $\phi(\boldsymbol{x})$. Let $f : \boldsymbol{a} \mapsto \boldsymbol{b}$ be the initial position.

The base case is trivial, since the position $\boldsymbol{a} \mapsto \boldsymbol{b}$ is a partial rigid homomorphism.

Suppose ϕ is an infinitary disjunction of the form $\bigvee \Phi$. Since $\mathfrak{A}, \boldsymbol{a} \models \phi(\boldsymbol{x})$, there is some $\psi \in \Phi$ such that $\mathfrak{A}, \boldsymbol{a}' \models \psi$, where $\boldsymbol{a}' \subseteq \boldsymbol{a}$ are the elements corresponding to free(ψ) $\subseteq \boldsymbol{x}$. Consider the move where Spoiler restricts to $\boldsymbol{a}'$. Since this is a valid move in the game, we know that Duplicator still has a winning strategy from $f' := f\restriction_{\boldsymbol{a}'}$, so the inductive hypothesis implies that $\mathfrak{B}, f'(\boldsymbol{a}') \models \psi$, and hence $\mathfrak{B}, f(\boldsymbol{a}) \models \phi$. The argument is similar for ϕ with the outermost connective a conjunction.

Suppose ϕ is of the form $\exists y.\psi(\boldsymbol{x}y)$. Since the width of ϕ is at most k, we know that $|\boldsymbol{x} \cup \{y\}| \leq k$, and hence it is possible for Spoiler to add an element in the game. Consider the move in the game corresponding to Spoiler adding the witness c such that $\mathfrak{A}, \boldsymbol{a}c \models \psi(\boldsymbol{x}y)$. Since Duplicator has a winning strategy, there is some d in $\mathfrak{B}$ such that $f' := f[c \mapsto d]$ is a partial rigid homomorphism. Hence by the inductive hypothesis, $\mathfrak{B}, f'(\boldsymbol{a}d) \models \psi(\boldsymbol{x}y)$, and $\mathfrak{B}, f(\boldsymbol{a}) \models \phi$.

Finally, suppose ϕ is of the form $\alpha(\boldsymbol{x}) \wedge \neg\psi$. We have $\mathfrak{A}, \boldsymbol{a} \models \alpha$ and and hence $\mathfrak{B}, \boldsymbol{b} \models \alpha$ since $\boldsymbol{a} \mapsto \boldsymbol{b}$ is a partial rigid homomorphism. This ensures that $\boldsymbol{a}$ is guarded in $\mathfrak{A}$, so it is possible for Spoiler to switch structures (and possibly restrict further to the elements corresponding to free(ψ)), and move to position $\boldsymbol{b} \mapsto \boldsymbol{a}$ (or some restriction of this). Hence, the inductive hypothesis shows that $\mathfrak{B}, \boldsymbol{b} \models \psi$ implies $\mathfrak{A}, \boldsymbol{a} \models \psi$. Since $\mathfrak{A}, \boldsymbol{a} \models \neg\psi$, this means that it must be the case that $\mathfrak{B}, \boldsymbol{b} \models \neg\psi$. Hence, $\mathfrak{B}, \boldsymbol{b} \models \phi$ as desired. □

Over structures of some fixed cardinality, GNFP-UP can be expressed in GNF$_\infty$.

Proposition 28. *For all* $\phi \in$ (GNFP-UP)$^k[\sigma]$ *and for all cardinals* κ*, there is* $\phi' \in \text{GNF}^k_\infty[\sigma]$ *such that for all structures* $\mathfrak{A}$ *of cardinality at most* κ*,* $\mathfrak{A} \models \phi$ *iff* $\mathfrak{A} \models \phi'$*.*

Proof. Consider $\psi := [\mathbf{lfp}^{\tau}_{Y,\boldsymbol{y}} . \text{gdd}(\boldsymbol{y}) \wedge \psi'](\boldsymbol{x})$ for $\psi' \in \text{GNF}^k_\infty[\sigma]$. Then for each ordinal β, the β-approximant to the fixpoint defined by ψ can be expressed in GNF$_\infty$, while preserving the width. This is easily shown by transfinite induction on β.

Now given $\phi \in$ (GNFP-UP)$^k[\sigma]$ and κ, we work from the inside out, replacing each fixpoint definition with its $\kappa + 1$ approximant, an upper bound on the closure ordinal where the fixpoint is reached in structures of cardinality at most κ. □

Unravelling The tree-like models for GNFP-UP (and indeed, for GNF and GNFP) can be obtained using an unravelling construction based on the GN bisimulation game.

Fix a structure $\mathfrak{A}$. Consider the set Π of sequences of the form $X_0X_1 \ldots X_n$, where $X_0 = \text{const}(\sigma)^{\mathfrak{A}}$, $X_i \subseteq \text{dom}(\mathfrak{A})$ with $|X_i| \leq k$, and $X_{i+1} = X_i \cup \{a\}$ for some $a \in \text{dom}(\mathfrak{A}) \setminus X_i$, or $X_{i+1} \subseteq X_i$. We can arrange these sequences in a tree based on the prefix order. Each sequence $\pi = X_0X_1 \ldots X_n$ identifies a unique node in the tree; we say $a \in \text{dom}(\mathfrak{A})$ is represented at node π if $a \in X_n$. For $a \in \text{dom}(\mathfrak{A})$, we say π and π' are a-equivalent if a is represented at every node on the unique minimal path between π and π' in this tree. For a represented at π, we write $[\pi, a]$ for the a-equivalence class. The GN$^k[\sigma]$-*unravelling of* $\mathfrak{A}$ is the structure $\mathfrak{A}^k$ with universe $\{[\pi, a] : \pi \in \Pi, a \text{ represented at } \pi\}$ and such that $e^{\mathfrak{A}^k} := [X_0, e^{\mathfrak{A}}]$ for $X_0 := \text{const}(\sigma)^{\mathfrak{A}}$ and $R^{\mathfrak{A}^k}([\pi_1, a_1], \ldots, [\pi_j, a_j])$ iff $R^{\mathfrak{A}}a_1 \ldots a_j$ and there is some $\pi \in \Pi$ such that for all i, $[\pi, a_i] = [\pi_i, a_i]$. There is a natural tree decomposition of width $k + |\text{const}(\sigma)| - 1$ for $\mathfrak{A}^k$ induced by the tree of sequences from Π. Note that this tree may have unbounded, infinite degree.

Proposition 29. *The* GN$^k[\sigma]$*-unravelling* $\mathfrak{A}^k$ *of* $\mathfrak{A}$ *is* GN$^k[\sigma]$*-bisimilar to* $\mathfrak{A}$*.*

Proof. We distinguish *safe positions* f in the GN$^k[\sigma]$-bisimulation game between $\mathfrak{A}$ and $\mathfrak{A}^k$: if the active structure is $\mathfrak{A}^k$, then f is safe if for all $[\pi, a] \in \text{dom}(f)$, $f([\pi, a]) = a$; if the active structure is $\mathfrak{A}$, then f is safe if there is some π such that $f(a) = [\pi, a]$ for all $a \in \text{dom}(f)$.

We now argue that starting from a safe position f, Duplicator has a strategy to move to a new safe position f'. This is enough to conclude that Duplicator has a winning strategy in the $\mathrm{GN}^k[\sigma]$-bisimulation game between $\mathfrak{A}$ and $\mathfrak{A}^k$ starting from any safe position.

First, assume that the active structure is $\mathfrak{A}^k$.

- If Spoiler restricts to some subset, then the restriction of f is clearly still safe.
- If Spoiler adds an element $[\pi', a']$, then Duplicator should choose a' in $\mathfrak{A}$ to maintain safety. This new map $f' = f[[\pi', a'] \mapsto a']$ is still a partial homomorphism since any relation holding for a tuple of elements $[\pi_1, a_1], \dots, [\pi_n, a_n]$ from $\mathrm{dom}(f')$ must hold for the tuple of elements $a_1, \dots, a_n$ in $\mathfrak{A}$ by definition of $\mathfrak{A}^k$. It is possible that there is some $[\pi, a']$ in $\mathrm{dom}(f')$ with $[\pi, a'] \neq [\pi', a']$; however, $[\pi, a']$ and $[\pi', a']$ are not guarded in $\mathfrak{A}^k$. Hence rigidity of f implies that any restriction f'' of f' to a guarded set of elements is a bijection. Moreover, such an f'' is a partial isomorphism: consider some $a_1, \dots, a_n \in \mathrm{rng}(f'')$ for which some relation R holds in $\mathfrak{A}$; since $(f'')^{-1}(a_1), \dots, (f'')^{-1}(a_n)$ must be guarded, we know that there is some π such that $[\pi, a_1] = (f'')^{-1}(a_1), \dots, [\pi, a_n] = (f'')^{-1}(a_n)$, so by definition of $\mathfrak{A}^k$, R holds of $(f'')^{-1}(a_1), \dots, (f'')^{-1}(a_n)$ as desired. Hence, f' is a safe rigid partial homomorphism.
- If Spoiler switches structures, then we know that $\mathrm{dom}(f)$ is guarded by some relation R, so there is some π with $f(a) = [\pi, a]$ for all $a \in \mathrm{dom}(f)$ by construction of $\mathfrak{A}^k$. Hence, $f' = f^{-1}$ is still safe.

Now assume that the active structure is $\mathfrak{A}$. Since f is safe, there is some π such that $f(a) = [\pi, a]$ for all $a \in \mathrm{dom}(f)$. Let X be the set of elements represented at π.

- If Spoiler restricts to some subset, then the restriction of f is still safe.
- If Spoiler adds an element a', then Duplicator should choose $[\pi', a']$, where $\pi' = \pi(X \cup \{a\}')$. By construction of the unravelling, $\pi' \in \Pi$ and the resulting partial mapping f' still satisfies the safety property with π' as witness. Now consider some tuple $\boldsymbol{a} = a_1 \dots a_n$ of elements from $\mathrm{dom}(f')$ that are in some relation R. We know that $f(a_i) = [\pi', a_i]$, hence R must hold for $f(\boldsymbol{a})$ in $\mathfrak{A}^k$. Moreover, for any guarded set $\boldsymbol{a} = \{a_1, \dots, a_n\}$ of distinct elements from $\mathrm{dom}(f')$, we know that $f(\boldsymbol{a})$ yields a set of distinct elements $\{f(a_1), \dots, f(a_n)\}$, and these elements can only participate in some fact in $\mathfrak{A}^k$ if the underlying elements from $\boldsymbol{a}$ participate in the same fact in $\mathfrak{A}$. Hence, f' is a safe partial rigid homomorphism.
- If Spoiler switches structures, then $f' = f^{-1}$ is still safe.

This is enough to conclude that Duplicator has a winning strategy starting from any safe position (in particular, the empty position), so $\mathfrak{A}^k$ and $\mathfrak{A}$ are $\mathrm{GN}^k[\sigma]$-bisimilar. □

We can now conclude the proof of Proposition 6. Assume that $\mathfrak{A}$ is a model of $\phi \in (\text{GNFP-UP})^k[\sigma]$, and let κ be the cardinality of $\mathfrak{A}^k$. By Proposition 28, there is an equivalent $\phi' \in \mathrm{GNF}^k_\infty[\sigma]$, at least on structures of cardinality at most κ. By Propositions 27 and 29, this means that $\mathfrak{A}^k$ is also a model of ϕ'. Hence, we can conclude that the unravelling $\mathfrak{A}^k$ is the desired tree-like model of ϕ.

B.2 Proof of Proposition 7 (GNFP-UP to GSO)

We prove:

> Given $\phi \in \text{GNFP-UP}[\sigma]$, we can construct an equivalent $\phi' \in \text{GSO}[\sigma]$.

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

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

where second-order quantifiers range over guarded relations. This uses the characterization of the least fixpoint as the intersection of all prefixpoints. Although the body of the fixpoint may use parameters $\boldsymbol{z}$, the tuples in the fixpoint are all guarded, so it suffices to range only over prefixpoints that are guarded relations (i.e. only contain tuples that are guarded by relations in σ). □

C. Missing proofs from Section 5

C.1 Proof of Theorem 13 (Localization)

Recall the statement:

> Let $\Sigma' := \Sigma \cup \{P_1, \dots, P_j\}$. Let $\mathcal{A}'$ be a 1-way nondeterministic parity automaton on Σ'-trees. We can construct a 2-way alternating parity automaton $\mathcal{A}$ on Σ-trees such that for all Σ-trees $\mathcal{T}$ and $v \in \mathrm{dom}(\mathcal{T})$,
>
> $$\mathcal{A}' \text{ accepts } \mathcal{T}' \text{ from the root iff } \mathcal{A} \text{ accepts } \mathcal{T} \text{ from } v,$$
>
> where $\mathcal{T}'$ is the Σ'-tree obtained from $\mathcal{T}$ by setting $P_1^{\mathcal{T}'} = \dots = P_j^{\mathcal{T}'} = \{v\}$. The number of states of $\mathcal{A}$ is linear in the number of states of $\mathcal{A}'$, and the overall size is linear in the size of $\mathcal{A}'$.

Proof. Roughly speaking, $\mathcal{A}$ simulates $\mathcal{A}'$ by guessing in a backwards fashion an initial part of a run of $\mathcal{A}'$ on the path from v to the root and then processing the rest of the tree in the normal downwards fashion. The subtlety is that the automaton $\mathcal{A}$ is reading a tree without the valuation for $P_1, \dots, P_j$ so once the automaton leaves node v, if it were to cross this position again, it would be unable to correctly simulate $\mathcal{A}'$. To avoid this problem, we only send downwards copies of the automaton in directions that are not on the path from the root to v.

The state set of $\mathcal{A}$ is $\{q_0\} \cup Q_{\mathcal{A}'} \cup (\{\mathbf{l}, \mathbf{r}\} \times Q_{\mathcal{A}'})$ consisting of initial state q_0, *downwards mode states* of the form $q \in Q_{\mathcal{A}'}$ and *upwards mode states* of the form $(d, q) \in (\{\mathbf{l}, \mathbf{r}\} \times Q_{\mathcal{A}'})$.

We describe the acceptance game of $\mathcal{A}$ on some Σ-tree $\mathcal{T}$ starting in node $v \in \mathrm{dom}(\mathcal{T})$; the actual transition function for the automaton can be extracted from this.

Initially, in state q_0 at node v, Eve guesses some state $q \in Q_{\mathcal{A}'}$ and some $(d_1, r_1) \wedge (d_2, r_2)$ in $\delta_{\mathcal{A}'}(q, \mathcal{T}'(v))$, where $\mathcal{T}'$ is the tree obtained from $\mathcal{T}$ by taking $P_1^{\mathcal{T}'} = \dots = P_j^{\mathcal{T}'} = \{v\}$. If v is the root, then q must be $q^0_{\mathcal{A}'}$. Adam can either challenge some (d_i, r_i) and move in direction d_i to downward state r_i, or (if v is not the root) he can move to the parent and switch to upward mode state (d, q), where v is the d-child of its parent.

In downwards mode state r at node w, the original automaton $\mathcal{A}'$ is simulated, i.e. Eve guesses some $(d_1, r_1) \wedge (d_2, r_2)$ in $\delta_{\mathcal{A}'}(r, \mathcal{T}(w))$, and then Adam chooses some i and moves in direction d_i to state r_i.

In upwards mode state (d, r) at node w, Eve guesses some state q and some $(d, r) \wedge (d', r')$ or $(d', r') \wedge (d, r)$ that is a disjunct in $\delta(q, \mathcal{T}(w))$. If w is the root, then q must be $q^0_{\mathcal{A}'}$. Adam can either move in direction d' to downward state r', or (if w is not the root) he can move to the parent of w in upward mode state (d'', q), where w is the d''-child of its parent. Note that Adam is not allowed to move in direction d, since this where the automaton came from.

The priority assignment is given by $\Omega_{\mathcal{A}}(q_0) := 1$, $\Omega_{\mathcal{A}}(r) := \Omega_{\mathcal{A}'}(r)$, and $\Omega_{\mathcal{A}}((d, r)) := \Omega_{\mathcal{A}'}(r)$.

If there is an accepting run ρ' of $\mathcal{A}'$ on $\mathcal{T}'$, then it is not hard to see that this induces a winning strategy ρ for Eve in the game $\mathcal{G}(\mathcal{A}, \mathcal{T}, v)$: Eve guesses in a backwards fashion the part of the run ρ' on the path from v to the root, and then processes the rest

of the tree in a normal downwards fashion, using ρ' to drive her choices. Using this strategy, any play is infinite and a suffix of this play (namely, once the play has switched to downward mode) corresponds directly to a suffix of a play in ρ'. Since the priorities in these suffixes are identical, the parity condition must be satisfied, so ρ is winning, and $\mathcal{A}$ accepts $\mathcal{T}$ starting from v.

Now suppose Eve has a winning strategy ρ in the acceptance game $\mathcal{G}(\mathcal{A}, \mathcal{T}, v)$. Using ρ, we stitch together an accepting run ρ' of $\mathcal{A}'$ on $\mathcal{T}'$. Recall that such an accepting run can be viewed as a labelling of $\mathcal{T}'$ with states consistent with $\delta_{\mathcal{A}'}$ and $\mathcal{T}'$ such that the maximum priority occurring infinitely often on each branch is even. We construct this labelling starting at v, based on Eve's guess of the state in the initial state. The labelling of the subtree rooted at v is then induced by the plays in ρ that switch immediately to downward mode at v. We can then proceed to label the parent u of v, by considering Eve's choice when Adam stays in upward mode and moves to the parent u of v. If v is the d-child of u, then the subtree in direction d from u is already labelled; the subtree in the other direction can be labelled by considering the plays when Adam switches to downward mode at u. Continuing in this fashion, we obtain a labelling of the entire tree with states such that $q^0_{\mathcal{A}'}$ is the label at the root and the other labels are consistent with $\delta_{\mathcal{A}'}$ on $\mathcal{T}'$ (it is consistent with $\mathcal{T}'$ and not $\mathcal{T}$, because in the initial state, Eve's choice of state and disjunct is under the assumption of the extra predicates present at v). Every branch in this run tree satisfies the parity condition, since a suffix of the branch corresponds to a suffix of a play in ρ that satisfies the parity condition. Hence, $\mathcal{A}'$ accepts $\mathcal{T}'$ from the root. □

D. Missing proofs from Section 6

D.1 Proof of Proposition 15 (Automaton checking consistency)

Recall the statement:

> There is a 2-way alternating parity tree automaton C that checks whether or not a $\tilde{\sigma}_k$-tree (possibly extended with additional free variable markers for $\boldsymbol{z}$ and $\boldsymbol{Z}$) is consistent. The size of C is at most exponential in $(|\sigma| + |\boldsymbol{z}| + |\boldsymbol{Z}|) \cdot |U_k|^k$.

Proof. We can construct a 2-way alternating parity tree automaton for each consistency condition, and then take the product of these automata using Proposition 9.

The constructions are straightforward, but we give one example, for the automaton checking the condition

> for all constants $z \in \sigma$, there is exactly one node w and one $c \in \text{names}(w)$ such that $V_{c/z}\, v$.

The automaton can be launched from any position in the tree. First, Eve navigates to a node v where some $V_{c/z}$ holds. While she is doing this, the automaton is in a state with priority 1. If she finds a node like this, then the automaton switches to a state with priority 0. In this state, Adam tries to navigate to a different node $v' \neq v$ where $V_{d/z}$ holds. He does this by choosing at each stage a direction to move in, with the restriction that he cannot move in the direction he came from (to ensure that he does not return to v). Hence, the automaton must record in its state whether the previous node was in direction $\mathbf{l}$, $\mathbf{r}$, or $\uparrow$. If he finds some $V_{d/z}$ at $v' \neq v$, then the automaton enters a sink state with priority 1, and Adam wins. There are a constant number of states, and the overall size is linear in the size of the alphabet, which is exponential in $(|\sigma|+|\boldsymbol{z}|+|\boldsymbol{Z}|)\cdot|U_k|^k$. □

D.2 Proof of Lemma 17 (Automata for GNFP)

We prove the following result about automata for GNFP:

> Let $\psi(\boldsymbol{y}, \boldsymbol{Z}) \in \text{GNFP}^k[\sigma']$ in normal form. Then for every local assignment $\boldsymbol{b}/\boldsymbol{y}$, we can construct a 2-way alternating parity tree automaton $\mathcal{A}^{\boldsymbol{b}/\boldsymbol{y}}_\psi$ such that for all consistent $\tilde{\sigma}'_k$-trees $(\mathcal{T}, \boldsymbol{Z}^{\rightarrow})$ and for all nodes $w \in \text{dom}(\mathcal{T})$ with $\boldsymbol{b} \subseteq \text{names}(w)$,
>
> $$\mathfrak{D}(\mathcal{T}), [w, \boldsymbol{b}], \boldsymbol{Z} \models \psi \text{ iff } \mathcal{A}^{\boldsymbol{b}/\boldsymbol{y}}_\psi \text{ accepts } (\mathcal{T}, \boldsymbol{Z}^{\rightarrow}) \text{ from } w.$$
>
> There is a polynomial function f independent of ψ such that the number of states for all such localized automata is at most $N := f(m) \cdot 2^{f(klr)}$ where $m = |\psi|$, $l = |\text{const}(\sigma')|$, and $r = \text{rank}_{\text{CQ}}(\psi)$. The number of priorities in each automaton is linear in $|\psi|$. The overall size is exponential in $|\sigma'| \cdot N$.

A similar result was already known from [8]. We cannot directly use this previous result, since this earlier work used a slightly different encoding of the tree-like models: we use a binary encoding here rather than working with trees with arbitrary branching degree; and we mark a free variable or constant in one node only rather than the entire connected part of the tree containing this element.

We present a construction here designed for our encodings. It is similar to the construction in [8], but it utilizes the localization theorem (Theorem 13). and deals directly with equality (whereas the construction in [8] treats equality separately).

We are now ready to give the automaton construction for GNFP.

Proof of Lemma 17. We describe the construction of $\mathcal{A}^{\boldsymbol{b}/\boldsymbol{y}}_\psi$, which proceeds by induction on the structure of ψ. Instead of formally specifying the transition function of the automaton, we describe the operation of $\mathcal{A}^{\boldsymbol{b}/\boldsymbol{y}}_\psi$ in terms of the roles of Adam and Eve. Let $\boldsymbol{z}$ denote the constants in σ'.

- Assume ψ is an atomic formula $R\,\boldsymbol{t}$ for $R \in \sigma'$ and $\boldsymbol{t} = t_1 \dots t_j$. We first construct a 2-way alternating automaton running on trees with markers for free(ψ). Eve navigates to a node w' with some $R_{\boldsymbol{a}}$. If she is able to do this, Adam can then challenge Eve to show that $\boldsymbol{t}$ corresponds to $\boldsymbol{a}$. Say he challenges her on $a_i \in \boldsymbol{a}$. Then Eve must navigate from w' to the node w'' carrying the marker for t_i and check that this marker is a_i. However, she must do this by passing through a series of biological neighbors that also contain a_i (the intermediate nodes in between biological neighbors might not contain a_i), to ensure that the a_i in w' corresponds to the same element in w''. If she is able to do this, she wins (the automaton enters a sink state with priority 0). The other states are assigned priority 1.
  The number of states of the automaton is linear in kl (indeed, linear in $|U_k|$), since the automaton must store the name a_i that Adam is challenging. We can convert this into an equivalent nondeterministic parity tree automaton with a state set of size exponential kl. We can localize this nondeterministic automaton to get localized 2-way alternating parity tree automata with a combined state set exponential in kl.
  The construction is similar for ψ of the form $Z\,\boldsymbol{t}$ for $Z \in \boldsymbol{Z}$, or for the guardedness predicate gdd($\boldsymbol{y}$).
- Assume ψ is an equality of the form $t_1 = t_2$. Again, we first construct an automaton running on trees with the free variable markers. Eve navigates to the node carrying the marker for t_1, with some name a. Then she must navigate to the node carrying t_2, by navigating along biological neighbors that also carry a. If she is able to do this, she wins (the automaton enters a sink state with priority 0). The other states are assigned priority 1.
  The number of states of the automaton is linear in kl, since the automaton must store the name a for t_1. We can convert this into an equivalent nondeterministic parity tree automaton with a state set of size exponential kl. We can localize this nondeterministic automaton to get localized 2-way alternating parity tree automata with a combined state set exponential in kl.

- Assume ψ is a CQ

$$\exists \boldsymbol{x}.(\alpha_1(\boldsymbol{xy}) \wedge \cdots \wedge \alpha_j(\boldsymbol{xy})).$$

This case is helpful for handling the general CQ-shaped formulas below. Again, we start by defining an automaton that runs on trees with the free variable markers. Take the intersection of the automata for each α_i (running on trees with markers for the free variables), together with an automaton checking that the free variable markers for $\boldsymbol{x}$ are unique. This results in a 2-way alternating automaton C of size linear in $klj \leq klr$. Using Theorem 12, we can then construct an equivalent nondeterministic tree automaton $\mathcal{B}$ with an exponential blow up in the number of states. The desired automaton is the projection of $\mathcal{B}$ on the free variable markers: it simulates $\mathcal{B}$ while guessing the markers for the free variables $\boldsymbol{x}$. Even after localization, the number of states is still at most exponential in klr.

- Assume ψ is a CQ-shaped formula

$$\exists \boldsymbol{x}.(\psi_1(\boldsymbol{xy}) \wedge \cdots \wedge \psi_j(\boldsymbol{xy})).$$

We know that each ψ_i is either an atom, or is a guarded subformula (i.e., the free variables in ψ_i are guarded).
We first construct an automaton $\mathcal{N}$ for the CQ δ obtained from ψ by replacing each non-atomic guarded subformula ψ_i with a new predicate Y_i. Examining this case above, we see that this automaton is of size at most exponential in klr. Note that this automaton runs on trees with a valuation for these guarded subformula predicates marked on the tree. The desired automaton starts by simulating $\mathcal{N}$, but with Eve guessing valuations for the guarded subformula predicates Y_i. Adam can either accept Eve's guesses of the valuation and continue the simulation of $\mathcal{N}$, or can challenge one of Eve's assertions for some Y_i by launching the localized version of ψ_i.
The number of states is linear in the sum of the number of states of $\mathcal{N}$ and the automata for the ψ_i, which is at most exponential in klr but polynomial in $|\psi|$.
- Assume ψ is of the from $\bigvee_i \psi_i$. Then Eve chooses some i and launches $\mathcal{A}^{\boldsymbol{b}_i/\boldsymbol{y}_i}_{\psi_i}$ from w.
- Assume ψ is of the form $\alpha \wedge \neg\psi'$. Let $\mathcal{A}^{\boldsymbol{b}/\boldsymbol{y}}_{\psi}$ be the automaton where Adam chooses whether to simulate $\mathcal{A}^{\boldsymbol{b}/\boldsymbol{y}}_{\alpha}$ or the dual of $\mathcal{A}^{\boldsymbol{b}/\boldsymbol{y}}_{\psi'}$.
- Assume ψ is of the form $\mathrm{gdd}(\boldsymbol{y}) \wedge \psi'$. Let $\mathcal{A}^{\boldsymbol{b}/\boldsymbol{y}}_{\psi}$ be the automaton where Adam chooses whether to simulate $\mathcal{A}^{\boldsymbol{b}/\boldsymbol{y}}_{\mathrm{gdd}(\boldsymbol{y})}$ or $\mathcal{A}^{\boldsymbol{b}/\boldsymbol{y}}_{\psi'}$.
- Assume ψ is a fixpoint $[\mathbf{lfp}_{X,\boldsymbol{x}}.\psi'(\boldsymbol{x}, X, \boldsymbol{Z})](\boldsymbol{t})$.
Let $\mathcal{A}^{\boldsymbol{b}/\boldsymbol{y}}_{\psi}$ be the automaton in which Eve navigates to a node w' with $\boldsymbol{t}' = \boldsymbol{t}[\boldsymbol{b}/\boldsymbol{y}, \boldsymbol{c}/\boldsymbol{z}]$ such that $[w', \boldsymbol{b}] = [w, \boldsymbol{b}]$ and $[w', c_i] = [w_i, c_i]$ where w_i is the node with the marking V_{c_i/z_i} (she immediately loses if she does not navigate to such a node).
Eve then simulates $\mathcal{A}^{\boldsymbol{t}'/\boldsymbol{x}}_{\psi'}$ while guessing a valuation for $X^{\rightarrow}$. During this simulation, if Eve guesses that $X_{\boldsymbol{a}}$ holds at u, then Adam can challenge this choice by launching another copy of $\mathcal{A}^{\boldsymbol{a}/\boldsymbol{x}}_{\psi'}$ from u; the priority in this case is some odd priority larger than the priorities in $\mathcal{A}^{\boldsymbol{t}'/\boldsymbol{x}}_{\psi'}$. The simulation continues in a similar fashion (with Eve guessing a valuation and Adam allowed to challenge these guesses).
The number of states is linear in the the sum of the sizes of the state sets for the localized versions of ψ', which meets the desired bound.
- Assume ψ is a simultaneous fixpoint of the form $[\mathbf{lfp}_{X_i,\boldsymbol{x_i}}.S](\boldsymbol{t})$.
We proceed as in the previous case, starting with automata for the formulas in the equations in S, and allowing Eve to guess a valuation for $X_j^{\rightarrow}$ for each second-order variable X_j in the system. Adam can challenge these guesses by launching the appropriate automata for formulas in S as described before.

□

D.3 Proof of Lemma 18 (Automata for GNFP-UP)

We prove the following result about automata for GNFP-UP:

> Let $\phi(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{Z})$ be a subformula of $\theta \in (\text{GNFP-UP})^k[\sigma]$ with $\mathrm{params}(\phi) \subseteq \boldsymbol{z}$. For each local assignment $\boldsymbol{b}/\boldsymbol{y}$, we can construct a 2-way alternating parity tree automaton $\mathcal{B}^{\boldsymbol{b}/\boldsymbol{y}}_{\phi}$ such that for all consistent $\tilde{\sigma}_k$-trees $(\mathcal{T}, \boldsymbol{z}^{\rightarrow}, \boldsymbol{Z}^{\rightarrow})$ and for all nodes $w \in \mathrm{dom}(\mathcal{T})$ with $\boldsymbol{b} \subseteq \mathrm{names}(w)$,
>
> $\mathfrak{D}(\mathcal{T}), [w, \boldsymbol{b}], \boldsymbol{z}, \boldsymbol{Z} \models \phi$ iff $\mathcal{B}^{\boldsymbol{b}/\boldsymbol{y}}_{\phi}$ accepts $(\mathcal{T}, \boldsymbol{z}^{\rightarrow}, \boldsymbol{Z}^{\rightarrow})$ from w.
>
> For $\mathrm{pdepth}_{\boldsymbol{z}}(\phi) \geq 1$, there is a polynomial function f independent of ϕ such that the size of all such localized automata is at most $\exp_f^{\mathrm{pdepth}_{\boldsymbol{z}}(\phi)}(f(mn) \cdot 2^{f(klr)})$ where $m = |\phi|$, $n = |\sigma|$, $l = |\mathrm{const}(\sigma)|$, and $r = \mathrm{rank}_{\mathrm{CQ}}(\phi)$. The number of priorities is linear in $|\phi|$. For $\mathrm{pdepth}_{\boldsymbol{z}}(\phi) = 0$, the bounds match Lemma 17.

Proof. We proceed by induction on $|\phi|$, with $\mathrm{params}(\phi) \subseteq \boldsymbol{z}$. Let $m = |\phi|$, $n = |\sigma|$, $l = |\mathrm{const}(\sigma)|$, and $r = \mathrm{rank}_{\mathrm{CQ}}(\phi)$.

Assume $\phi' := \mathrm{transform}_{\boldsymbol{z}}(\phi)$ is helpful. Note that this always holds for the smallest (atomic) formulas, so this covers the base of the induction. We construct $\mathcal{B}^{\boldsymbol{b}/\boldsymbol{y}}_{\phi}$ to simulate the automaton $\mathcal{A}^{\boldsymbol{b}/\boldsymbol{y}}_{\phi'}$ from Lemma 17, while allowing Eve to guess valuations for the F_η relations from ϕ'. Constants and free variables are encoded in the same way, so it makes no difference that we are viewing $\boldsymbol{z}$ as free variables in the automaton for ϕ and as constants in the automaton for ϕ'. Since ϕ' is helpful, we know that every F_η relation in ϕ' is for some formula η that is strictly smaller than ϕ, and hence the inductive hypothesis ensures there is a corresponding automaton for every suitable local assignment. During the simulation of $\mathcal{A}^{\boldsymbol{b}/\boldsymbol{y}}_{\phi'}$, if Eve asserts $F_{\eta(\boldsymbol{x})}\,\boldsymbol{a}$ at w for some $\boldsymbol{a} \subseteq \mathrm{names}(w)$, then Adam can challenge this by launching the automaton for the intersection of $\mathcal{B}^{\boldsymbol{a}/\boldsymbol{x}}_{\eta(\boldsymbol{x})}$ and $\mathcal{B}^{\boldsymbol{a}/\boldsymbol{x}}_{\mathrm{gdd}(\boldsymbol{x})}$ from w; likewise, if Eve does not assert $F_{\eta(\boldsymbol{x})}\,\boldsymbol{a}$ at w for some $\boldsymbol{a} \subseteq \mathrm{names}(w)$, then Adam can challenge this by launching the automaton for the dual of the intersection of $\mathcal{B}^{\boldsymbol{a}/\boldsymbol{x}}_{\eta(\boldsymbol{x})}$ and $\mathcal{B}^{\boldsymbol{a}/\boldsymbol{x}}_{\mathrm{gdd}(\boldsymbol{x})}$ from w. Correctness follows from Lemma 17, Lemma 30 (see below), and the fact that the inductive hypothesis ensures that the subautomata for F_η in ϕ' are correct. Observe that the size of $\mathcal{A}^{\boldsymbol{b}/\boldsymbol{y}}_{\phi'}$ is at most $\exp(g(mn) \cdot 2^{g(klr)})$ for some polynomial function g, since $|\sigma_{\boldsymbol{z},\phi}| \leq m+n$. The size of the subautomata for each F_η relation appearing in ϕ' is at most $\exp_f^{\mathrm{pdepth}_{\boldsymbol{z}}(\eta)}(f(|\eta|n) \cdot 2^{f(klr)})$. Moreover, the sum of the sizes of these subformulas η is strictly less than the size of ϕ. Assuming f sufficiently dominates g, this ensures that the size of the automaton for ϕ is at most $\exp_f^{\mathrm{pdepth}_{\boldsymbol{z}}(\phi)}(f(|\phi|n) \cdot 2^{f(klr)})$.

Next, assume that $\phi' := \mathrm{transform}_{\boldsymbol{z}}(\phi)$ is unhelpful. Then we need to consider two possible cases. Throughout these arguments we make use of the closure properties described in Section 5.

- The first case is that ϕ is a UCQ-shaped subformula

$$\bigvee_i \exists \boldsymbol{x}_i. \bigwedge_j \psi_{ij}$$

where variables from some $\boldsymbol{x}_i$ are used as parameters in $\bigwedge_j \psi_{ij}$. To start, we consider each CQ-shaped formula separately, so fix some $\exists \boldsymbol{x}_i. \bigwedge_j \psi_{ij}$. We use the inductive hypothesis to obtain 2-way alternating automata for each conjunct ψ_{ij}, using the empty local assignment. This is possible since the size of these conjuncts must be strictly less than the size of ϕ. Because we are using the empty local assignment, these automata operate on

trees with markers for all of the free variables: $\boldsymbol{x}_i \cup \boldsymbol{y} \cup \boldsymbol{z}$. Take the intersection of these automata using Proposition 9. Then take the intersection with the automaton from Proposition 15 checking consistency. This yields a 2-way alternating automaton corresponding to $\bigwedge_j \psi_{ij}$. We then convert this automaton to a nondeterministic version using Theorem 12 and project away the information about $\boldsymbol{x}_i$ using Proposition 10. This yields an equivalent (1-way) nondeterministic parity automaton for the CQ. Next, we localize this automaton to the desired variables $\boldsymbol{y}$ using Theorem 13. Finally, to construct the automaton $\mathcal{B}_\phi^{\boldsymbol{b}/\boldsymbol{y}}$, we take the union of the individual CQ automata using Proposition 9.

The conversion from a 2-way alternating automaton to a nondeterministic automaton is the costly step in this process, resulting in an exponential blow-up. Observe that $\mathrm{pdepth}(\psi_{ij}) < \mathrm{pdepth}_{\boldsymbol{z}}(\phi)$ for each conjunct ψ_{ij} in the CQs. If $\mathrm{pdepth}_{\boldsymbol{z}}(\phi) \geq 2$, then the size of each $\mathcal{B}_{\psi_{ij}}^{\emptyset/\emptyset}$ is at most $\exp_f^{\mathrm{pdepth}_{\boldsymbol{z}}(\phi)-1}(f(|\psi_{ij}|n) \cdot 2^{f(klr)})$, so we can absorb the exponential blow-up. Likewise, if $\mathrm{pdepth}_{\boldsymbol{z}}(\phi) = 1$, then the size of each $\mathcal{B}_{\psi_{ij}}^{\emptyset/\emptyset}$ is at most $\exp(g(|\psi_{ij}|n) \cdot 2^{g(klr)})$ but the number of states is at most $g(|\psi_{ij}|) \cdot 2^{g(klr)}$ for some polynomial function g. Because the conversion to a 1-way nondeterministic automaton is actually exponential only in the size of the state set, this means that we can absorb the blow-up here too. In either case, it can be checked that the size of $\mathcal{B}_\phi^{\boldsymbol{b}/\boldsymbol{y}}$ is at most $\exp_f^{\mathrm{pdepth}_{\boldsymbol{z}}(\phi)}(f(mn) \cdot 2^{f(klr)})$.

- The second case is when ϕ is a fixpoint formula where fixpoint variables are used as parameters in the body of the fixpoint. Suppose ϕ is of the form $[\mathbf{lfp}^{\boldsymbol{z}'}_{X,\boldsymbol{x}} . \mathrm{gdd}(\boldsymbol{x}) \wedge \chi(\boldsymbol{x}\boldsymbol{z}', X\boldsymbol{Z})](\boldsymbol{t})$ where $\mathrm{params}(\chi) \cap \boldsymbol{x} \neq \emptyset$, so $\mathrm{params}(\chi) \not\subseteq \boldsymbol{z}$. The formula χ in the body of the fixpoint is strictly smaller, so we can apply the inductive hypothesis to get an automaton for this part. We want this automaton to be localized to $\boldsymbol{x}$ so we can test for tuples in the fixpoint by launching copies of the automaton for some local assignment. However, the inductive hypothesis does not directly yield this, since some variables from $\boldsymbol{x}$ are used as parameters. Hence, we must use the inductive hypothesis to get a 2-way automaton for $\mathcal{B}_{\chi'}^{\emptyset/\emptyset}$, apply Theorem 12 to get an equivalent 1-way nondeterministic automaton, and then localize to some assignment $\boldsymbol{a}/\boldsymbol{x}$ using Theorem 13.

  Once we have these localized automata $\mathcal{B}_\chi^{\boldsymbol{a}/\boldsymbol{x}}$ for the body of the fixpoint, we can proceed as we did for the fixpoint case in the GNFP automaton construction.

  The conversion between 2-way and 1-way nondeterministic automata results in an exponential blow-up, but $\mathrm{pdepth}_{\boldsymbol{z}}(\phi) = \mathrm{pdepth}_{\boldsymbol{z}}(\chi) + 1$, so the size of the fully localized automata for χ are at most $\exp_f^{\mathrm{pdepth}_{\boldsymbol{z}}(\phi)}(f(|\chi|n) \cdot 2^{f(klr)})$. Hence, the overall size of the automaton is at most $\exp_f^{\mathrm{pdepth}_{\boldsymbol{z}}(\phi)}(f(mn) \cdot 2^{f(klr)})$.

  The argument is similar if ϕ is a simultaneous fixpoint.

□

In the previous proof, we made use of the following straightforward lemma, which just says that the GNFP formula obtained using the transformation operation is "equivalent" to the GNFP-UP formula, under the assumption that the additional predicates in the augmented signature used by the GNFP formula are interpreted in the expected way.

Lemma 30. *Let $\psi(\boldsymbol{y}\boldsymbol{z}, \boldsymbol{Z}) \in (\text{GNFP-UP})^k[\sigma]$ in normal form with $\mathrm{params}(\psi) \subseteq \boldsymbol{z}$. If $\psi'(\boldsymbol{y}) := \mathit{transform}_{\boldsymbol{z}}(\psi) \in \mathrm{GNFP}^k[\sigma_{z,\psi}]$ then for all σ-structures $\mathfrak{M}$, for all σ-guarded relations $\boldsymbol{J}$ for $\boldsymbol{Z}$, for all σ-guarded $\boldsymbol{b}$ for $\boldsymbol{y}$ and all $\boldsymbol{c}$ for $\boldsymbol{z}$,*

$$\mathfrak{M}, \boldsymbol{b}\boldsymbol{c}, \boldsymbol{J} \models \psi(\boldsymbol{y}\boldsymbol{z}, \boldsymbol{Z}) \quad \textit{iff} \quad \mathfrak{M}', \boldsymbol{b}, \boldsymbol{J} \models \psi'(\boldsymbol{y}, \boldsymbol{Z})$$

where $\mathfrak{M}'$ is the $\sigma_{z,\psi}$-structure obtained from $\mathfrak{M}$ by taking $\boldsymbol{z}^{\mathfrak{M}'} := \boldsymbol{c}$, and

$$F^{\mathfrak{M}'}_{\eta(\boldsymbol{x})} := \{\boldsymbol{a} : \boldsymbol{a} \textit{ is } \sigma\textit{-guarded in } \mathfrak{M} \textit{ and } \mathfrak{M} \models \eta[\boldsymbol{a}/\boldsymbol{x}, \boldsymbol{c}/\boldsymbol{z}]\} .$$

Proof. We proceed by induction on the structure of ψ. For notational simplicity, we omit the second-order variables $\boldsymbol{Z}$ in the parts of the argument where these do not play a role.

- If ψ is atomic, then the result is immediate since $\boldsymbol{z}^{\mathfrak{M}'} := \boldsymbol{c}$.
- Assume ψ is of the form $\alpha \wedge \neg\phi$. Then $\psi' := \alpha \wedge \neg \mathrm{transform}_{\boldsymbol{z} \cap \mathrm{free}(\phi)}(\phi)$. Since $\boldsymbol{z}^{\mathfrak{M}'} := \boldsymbol{c}$, $\mathfrak{M} \models \alpha[\boldsymbol{b}/\boldsymbol{y}, \boldsymbol{c}/\boldsymbol{z}]$ iff $\mathfrak{M}' \models \alpha[\boldsymbol{b}/\boldsymbol{y}]$. By the inductive hypothesis, $\mathfrak{M} \models \phi[\boldsymbol{b}/\boldsymbol{y}, \boldsymbol{c}/\boldsymbol{z}]$ iff $\mathfrak{M}' \models \phi[\boldsymbol{b}/\boldsymbol{y}]$. This implies the desired result.
- Assume ψ is $[\mathbf{lfp}^{\boldsymbol{z}'}_{X,\boldsymbol{x}} . \mathrm{gdd}(\boldsymbol{x}) \wedge \phi(\boldsymbol{x}\boldsymbol{z}', X\boldsymbol{Z})](\boldsymbol{t})$.
  Consider the case when $\psi' = F_\psi \boldsymbol{t}$. If $\mathfrak{M}' \models (F_\psi \boldsymbol{t})[\boldsymbol{b}/\boldsymbol{y}]$, then $\mathfrak{M} \models \psi[\boldsymbol{b}/\boldsymbol{y}, \boldsymbol{c}/\boldsymbol{z}]$ by definition of $F^{\mathfrak{M}'}_\psi$. Likewise, if $\mathfrak{M} \models \psi[\boldsymbol{b}/\boldsymbol{y}, \boldsymbol{c}/\boldsymbol{z}]$, $\boldsymbol{t}[\boldsymbol{b}/\boldsymbol{y}, \boldsymbol{c}/\boldsymbol{z}]$ must be σ-guarded since any tuple in the fixpoint is guarded. Hence, $\boldsymbol{t}[\boldsymbol{b}/\boldsymbol{y}, \boldsymbol{c}/\boldsymbol{z}] \in F^{\mathfrak{M}'}_\psi$ and $\mathfrak{M}' \models (F_\psi \boldsymbol{t})[\boldsymbol{b}/\boldsymbol{y}]$.
  Otherwise, consider the case when $\psi' = [\mathbf{lfp}_{X,\boldsymbol{x}} . \mathrm{gdd}(\boldsymbol{x}) \wedge \mathrm{transform}_{\boldsymbol{z}}(\phi(\boldsymbol{x}, X\boldsymbol{Z}))](\boldsymbol{t})$. We can prove by a straightforward induction on the fixpoint approximant β (an ordinal) that for all σ-guarded tuples $\boldsymbol{a}$, $\boldsymbol{a} \in \phi^\beta_{\mathfrak{M}}$ iff $\boldsymbol{a} \in (\mathrm{transform}_{\boldsymbol{z}}(\phi))^\beta_{\mathfrak{M}'}$. The result follows by noting that the least fixpoint is the union over these approximants, and $\boldsymbol{t}[\boldsymbol{b}/\boldsymbol{y}, \boldsymbol{c}/\boldsymbol{z}]$ must be σ-guarded to be in either of the fixpoints defined by ψ and ψ'.
- Assume ψ is $\bigvee_i \exists \boldsymbol{x}_i . \bigwedge_j \psi_{ij}$.
  Consider the case when ψ' is $F_\eta \boldsymbol{y}$. Since $\boldsymbol{b}$ is σ-guarded by assumption, $\mathfrak{M} \models \psi[\boldsymbol{b}/\boldsymbol{y}, \boldsymbol{c}/\boldsymbol{z}]$ iff $\mathfrak{M}' \models (F_\eta \boldsymbol{y})[\boldsymbol{b}/\boldsymbol{y}]$ by definition of $S^{\mathfrak{M}'}_\psi$.
  On the other hand, if ψ' is $\bigvee_i \exists \boldsymbol{x}_i . \bigwedge_j \mathrm{transform}_{\boldsymbol{z} \cap \mathrm{free}(\psi_{ij})}(\psi_{ij})$, then the result follows by a straightforward application of the inductive hypothesis.

□

This completes the proof of Lemma 18.

E. Missing proofs from Section 7

E.1 Proof of Theorem 20 (Satisfiability for GNFP-UP)

We prove:

> Satisfiability for $\theta \in$ GNFP-UP is decidable in (pdepth(θ) + 2)-ExpTime.

Proof. Let k be the width of θ. We first consider the case when $\mathrm{pdepth}(\theta) \geq 1$. Given $\theta \in$ GNFP-UP with signature σ, we can construct a 2-way alternating parity tree automaton $\mathcal{B}_\theta^{\emptyset/\emptyset}$ using Theorem 16 or Corollary 19 of (pdepth(θ) + 1)-exponential size, since the width, CQ-rank, and number of constants are at most $|\theta|$. Take the product of $\mathcal{B}_\theta^{\emptyset/\emptyset}$ with the automaton C from Proposition 15 that checks that a tree is a consistent $\tilde{\sigma}_k$-tree. Call the resulting automaton $\mathcal{A}_\theta$. The size of $\mathcal{A}_\theta$ is still of (pdepth(θ)+1)-exponential size. By the tree-like model property in Proposition 6, we know that there is a model for θ iff there is a tree-like model of tree-width $k + |\mathrm{const}(\theta)| - 1$. Hence, θ is satisfiable iff $L(\mathcal{A}_\theta)$ is empty. We can decide emptiness for a 2-way alternating parity tree automaton in ExpTime hence satisfiability θ can be decided in (pdepth(θ) + 2)-ExpTime.

For $\mathrm{pdepth}(\theta) = 0$, we can view θ as a GNFP formula (with any free variables viewed as constants) and apply Lemma 17 instead of Theorem 16 to start. Although the overall size of this automaton is doubly exponential in $|\theta|$, the number of states is only single exponential and the number of priorities is linear. Moreover, emptiness

for 2-way alternating parity tree automata is exponential only in the number of states and number of priorities, so the satisfiability test can be done in 2-ExpTime as desired. □

E.2 Proof of Corollary 23 (Satisfiability for CQPDL)

We prove:

> Satisfiability of CQPDLs (or boolean combinations of CQPDL sentences and GNFP sentences) is decidable in 2-ExpTime.

We first prove the following lemma about the translation of PDL programs and formulas into GNFP-UP:

Lemma 31. *For each PDL program P we can construct in* PTime *an equi-satisfiable* GNFP-UP *formula* $\mathrm{translate}(P)(x,y) \wedge \Gamma_P$ *in normal form such that* $\mathrm{translate}(P)$ *has pdepth 0 and CQ-rank at most 3 and* Γ_P *has pdepth 1 and CQ-rank at most 3.*

Likewise, for each PDL test t, we construct in PTime *an equi-satisfiable* GNFP-UP *formula in normal form* $\mathrm{translate}(t)(x) \wedge \Gamma_t$ *such that* $\mathrm{translate}(t)$ *has pdepth 0 and CQ-rank at most 3 and* Γ_t *has pdepth 1 and CQ-rank at most 3.*

Proof. The proof is by induction.

An atomic program P is a binary relation R (or its inverse R^-); in this case, we set $\mathrm{translate}(P)(x,y) := Rxy$ (respectively, $\mathrm{translate}(P)(x,y) := Ryx$). We set $\Gamma_P := \top$ in either case. Atomic tests are similar.

Consider an arbitrary program P. We start by flattening P into some P' such that every $t?$ at the outer level in P is replaced with a new unary relation $U_{t?}$. The resulting P' is a traditional regular expression over the original binary relations and the special unary relations $U_{t?}$. We can then define $\mathrm{translate}(P)$ to be the normal form GNFP-UP formula that describes the operation of the finite state automaton $\mathcal{A}$ for P'. This is similar to Example 4, but for technical reasons, we describe the backwards operation of the automaton. In more detail: we start with a finite state automaton $\mathcal{A}$ for P' (obtained from P' in PTime), with states Q, initial state q_0, and accepting states F. We write a GNFP-UP formula with simultaneous fixpoints $[\mathbf{lfp}^{x}_{Y,y} \ . \ S](y)$ which has a second-order variable Y_i for each state $q_i \in Q$, together with an additional variable Y. The equation for the j-th component Y_j, y_j in S captures the possible backwards transitions from state q_j:

$$\bigvee_{(q_i,T,q_j)\in\Delta} \exists z.(\chi_T(z,y_j) \wedge Y_i z) \ \vee \ \begin{cases} y_j = x & \text{if } j = 0 \\ \bot & \text{if } j \neq 0 \end{cases}$$

where $\chi_T(z,y_j)$ is Rzy_j if $T = R$, Ry_jz if $T = R^-$, and $y_j = z \wedge U_{t?}(z)$ if $T = U_{t?}$. The equation for the goal predicate Y, y captures the acceptance condition: $\bigvee_{q_i\in F} Y_i y$. This simultaneous fixpoint has parameters x, pdepth 0, and CQ-rank at most 3. We set Γ_P to be the conjunction of Γ_t for each $U_{t?}$ in P', together with $\forall x.(U_{t?}(x) \leftrightarrow \mathrm{translate}(t)(x))$ that describes the meaning of $U_{t?}$; this can be expressed in GNFP-UP since it uses only unary negation.

Now consider an arbitrary test t. We can translate atomic tests using the inductive hypothesis. We replace any test of the form $\langle P\rangle$ in t with the GNFP-UP formula $\exists y.\mathrm{translate}(P)(x,y)$, which still has pdepth 0 since $\mathrm{translate}(P)(x,y)$ has pdepth 0 and the quantified variable y is not used as a parameter in $\mathrm{translate}(P)(x,y)$. Since only unary negation is used, all boolean operations in t can also be expressed in GNFP-UP. Moreover, this boolean combination can always be written in normal form with only a polynomial blow-up in size, and no change to the CQ-rank (since it is composed of only unary formulas, and the quantification comes only from the translation of $\langle P\rangle$). Finally, Γ_t is obtained by taking the conjunction of Γ_P for all $\langle P\rangle$ appearing in t. □

Now given some input CQPDL formula

$$\exists \boldsymbol{x}.(P_1(\boldsymbol{x}_1) \wedge \cdots \wedge P_n(\boldsymbol{x}_n))$$

we can construct in PTime an equi-satisfiable GNFP-UP formula

$$\exists \boldsymbol{x}.(\mathrm{translate}(P_1)(\boldsymbol{x}_1) \wedge \cdots \wedge \mathrm{translate}(P_n)(\boldsymbol{x}_n)) \wedge \Gamma_{P_1} \wedge \cdots \wedge \Gamma_{P_n}$$

of pdepth 1 using Lemma 31. Using Theorem 20, we can then get a 3-ExpTime bound on satisfiability for CQPDL.

We can do better by noting that the formulas produced by Lemma 31 have a fixed width, fixed CQ-rank, and no constants. Hence, the size of the 2-way alternating automaton for $\Gamma_{P_1} \wedge \cdots \wedge \Gamma_{P_n}$ is only exponential in the size of the input by Lemma 18. Moreover, the automaton for each $\mathrm{translate}(P_i)(\boldsymbol{x}_i)$ is only exponential in the size of the input, with state set of size only polynomial in the size of the input by Lemma 17. Thus, by taking the intersection of the 2-way alternating automata for the $\mathrm{translate}(P_i)(\boldsymbol{x}_i)$, converting to a nondeterministic version, and then projecting, we have an automaton for $\exists \boldsymbol{x}.(\mathrm{translate}(P_1)(\boldsymbol{x}_1) \wedge \cdots \wedge \mathrm{translate}(P_n)(\boldsymbol{x}_n))$ that is only exponential in the size of the input.

Using Lemma 17, the state set and priorities of automata for GNFP sentences is also only at most exponential in the size of the input using (although the overall size may be doubly exponential).

Hence, an automaton for a boolean combination of CQPDL sentences and GNFP sentences can be constructed in at most doubly exponential time, and has at most exponentially many states and priorities. Using Theorem 14, this means we can test satisfiability in doubly exponential time as desired in Corollary 23.

F. Missing proofs from Section 8

F.1 Proof sketch of Theorem 24 (Boundedness for GNFP-UP)

In this section, we sketch the proof of Theorem 24:

> Assuming ILT, the boundedness problem is decidable for $\boldsymbol{x}$-guarded $\phi(\boldsymbol{x}, X) \in$ GNFP-UP$[\sigma]$ in non-elementary time (elementary time for fixed pdepth).
>
> In the special case of $\boldsymbol{x}$-guarded $\phi(\boldsymbol{x}, \boldsymbol{z}, X) \in$ GNF$[\sigma]$, boundedness is decidable in elementary time (without assuming ILT).

We start with the first part of the theorem, about answer-guarded $\phi(\boldsymbol{x}, X) \in$ GNFP-UP. Like we did for satisfiability, we first observe that we can restrict to considering structures of bounded tree-width.

Proposition 32. *Let $\phi(\boldsymbol{x}, X) \in$ (GNFP-UP)$^k[\sigma]$ be positive in X (of arity $|\boldsymbol{x}|$) and answer-guarded. Then $\phi(\boldsymbol{x}, X)$ is bounded over all σ-structures iff it is bounded over tree-like σ-structures of tree-width at most $k + |\mathrm{const}(\sigma)| - 1$.*

This is a straightforward consequence of the tree-like model property of (GNFP-UP)^k (Proposition 6), and the fact that the fixpoint approximants for natural numbers n are expressible in (GNFP-UP)^k. It opens up the ability to use automata for boundedness, like we did earlier for the question of satisfiability. However, we must shift to using a type of automaton that can carry some quantitative information about the number of fixpoint unfoldings that are used.

In particular, we reduce the question of boundedness for $\phi(\boldsymbol{x}, X) \in$ (GNFP-UP)$^k[\sigma]$ to a question of boundedness for a *cost automaton*, a traditional automaton enriched with a finite set of counters that are initially zero and that can be incremented, reset, or left unchanged on each transition. A cost automaton $\mathcal{B}$ naturally defines a function $[\![\mathcal{B}]\!]$ that maps an input to the least $n \in \mathbb{N} \cup \{\infty\}$ such that Eve has a winning strategy in the acceptance game that keeps the counter values below the threshold n. The *boundedness problem for cost automata* is:

Given a cost automaton $[\![\mathcal{A}]\!]$ over σ-trees, is there some n such that on all σ-trees $\mathcal{T}$, $[\![\mathcal{A}]\!](\mathcal{T}) \leq n$?

This is the target problem for our reduction.

The following theorem describes the automata that we construct. The idea is to adapt the automata we constructed in Theorem 16 to count the outermost fixpoint unfolding. This construction actually produces a special type of cost automaton called a *distance ∧ parity automaton*: it has only a single counter that is incremented or left unchanged on each transition (never reset). We only need a single counter like this, since we only have to count the unfolding of the single outermost fixpoint.

Theorem 33. *Let $\phi(\boldsymbol{x}, X) \in (\text{GNFP-UP})^k[\sigma]$ such that it is positive in X of arity $|\boldsymbol{x}|$ and answer-guarded, and $\text{pdepth}(\phi) \geq 1$. There is a 2-way alternating cost automaton $\mathcal{B}_\phi$ on $\tilde{\sigma}_k$-trees such that $\phi(\boldsymbol{x}, X)$ is bounded over all σ-structures iff $[\![\mathcal{B}_\phi]\!]$ is bounded over all $\tilde{\sigma}_k$-trees. Moreover, $\mathcal{B}_\phi$ is a distance ∧ parity automaton of $(\text{pdepth}(\phi) + 1)$-size. The number of priorities is linear in $|\phi|$.*

Proof sketch. For a fixed structure $\mathfrak{A}$, what we want to measure is the threshold n such that

for all guarded tuples of elements $\boldsymbol{a}$ in $\text{dom}(\mathfrak{A})$, if $\boldsymbol{a}$ is in the least fixpoint induced by ϕ, then $\boldsymbol{a}$ is in the n-unfolding of ϕ.

There is a cost automaton that computes this threshold for encodings of tree-like structures. The idea is to take the automata we constructed in Section 6 and add a single counter which is incremented exactly when the fixpoint is unfolded.

Claim 34. *There is a 2-way alternating cost parity tree automaton $\mathcal{B}$ such that for all consistent $\tilde{\sigma}_k$-trees $\mathcal{T}$, $[\![\mathcal{B}]\!](\mathcal{T}) \leq n$ iff for all tuples $\boldsymbol{a} \in \text{dom}(\mathfrak{D}(\mathcal{T}))$, if $\boldsymbol{a}$ is in in the least fixpoint induced by ϕ in $\mathfrak{D}(\mathcal{T})$, then $\boldsymbol{a}$ is in the n-unfolding of ϕ in $\mathfrak{D}(\mathcal{T})$[3] Moreover, $\mathcal{B}$ is a distance ∧ parity automaton of $(\text{pdepth}(\phi) + 1)$-size, and the number of priorities is linear in $|\phi|$.*

Proof of claim. Using Lemma 18, we can construct for each localization $\boldsymbol{b}/\boldsymbol{y}$ a localized 2-way alternating parity tree automaton $\mathcal{A}^{\boldsymbol{b}/\boldsymbol{y}}$ such that when launched from some node v in a consistent $\tilde{\sigma}_k$-tree $\mathcal{T}$, the automaton accepts just if $\mathfrak{D}(\mathcal{T}), [v, \boldsymbol{b}] \models [\mathbf{lfp}_{X,\boldsymbol{x}} . \text{gdd}(\boldsymbol{x}) \wedge \phi(\boldsymbol{x}, X)](\boldsymbol{y})$. We can create a cost automaton based on this that actually counts the number of fixpoint unfoldings that are needed. Let $\mathcal{B}^{\boldsymbol{b}/\boldsymbol{y}}$ be the result of taking $\mathcal{A}^{\boldsymbol{b}/\boldsymbol{y}}$ and adding a single counter. This counter is never touched, except for the transitions when a fixpoint is unfolded: namely, the transitions where Adam challenges Eve's choice of valuation for $X^{\rightarrow}$. The value of $\mathcal{B}^{\boldsymbol{b}/\boldsymbol{y}}$ when it starts from v in a consistent $\tilde{\sigma}_k$-tree $\mathcal{T}$ is $n \in \mathbb{N}$ iff $\mathfrak{D}(\mathcal{T}), [v, \boldsymbol{b}]$ satisfies the n-unfolding of the fixpoint (but does not satisfy the n'-unfolding for any $n' < n$).

The automaton $\mathcal{B}$ operates as follows. First, Adam is allowed to navigate to some node in the tree, and pick out some guarded tuple (i.e. some node v and localization $\boldsymbol{b}/\boldsymbol{y}$). Eve can then choose whether to launch the dual of $\mathcal{A}^{\boldsymbol{b}/\boldsymbol{y}}$ or the cost automaton $\mathcal{B}^{\boldsymbol{b}/\boldsymbol{y}}$. Note that Adam cannot force a high value by choosing tuples outside of the fixpoint, since in that case Eve can simply run the dual of $\mathcal{A}^{\boldsymbol{b}/\boldsymbol{y}}$, which will yield value 0. For actual guarded tuples in the fixpoint, however, Eve must simulate $\mathcal{B}^{\boldsymbol{b}/\boldsymbol{y}}$ in order to win, so the value in this case will be the number of unfoldings required to witness that this tuple is in the fixpoint.

Overall, this ensures that $[\![\mathcal{B}]\!](\mathcal{T}) \leq n$ iff for all tuples $\boldsymbol{a} \in \text{dom}(\mathfrak{D}(\mathcal{T}))$, if $\boldsymbol{a}$ is in in the least fixpoint induced by ϕ, then $\boldsymbol{a}$ is in the n-unfolding of ϕ.

[3] If $n = \infty$, then by the n-unfolding we mean the fixpoint itself, so the statement is vacuous.

Due to the size bounds in Lemma 18, the size of $\mathcal{B}$ is at most $(\text{pdepth}(\phi) + 1)$-exponential in the size of ϕ. □

The automaton guaranteed by this claim has the property that it is bounded over all consistent $\tilde{\sigma}_k$-trees iff ϕ is bounded over all tree-like σ-structures of width $k + |\text{const}(\sigma)| - 1$. However, the boundedness problem for cost automata talks about the values over all $\tilde{\sigma}_k$-trees, not just consistent $\tilde{\sigma}_k$-trees. Hence, the desired automaton $\mathcal{B}_\phi$ is just the union of $\mathcal{B}$ and the dual of the automaton C from Proposition 15 that checks that the input $\tilde{\sigma}_k$-tree is consistent. This automaton $\mathcal{B}_\phi$ is bounded over all $\tilde{\sigma}_k$-trees iff ϕ is bounded over all tree-like σ-structures of width $k + |\text{const}(\sigma)| - 1$ iff (by Proposition 32) ϕ is bounded over all σ-structures.

This concludes the proof sketch of Theorem 33. □

It is not known in general whether boundedness is decidable for cost automata over infinite trees. However, boundedness for 2-way distance ∧ parity automata was shown to be decidable in [8, Theorem 12], subject to the unpublished result of Colcombet (called ILT or Dual-Inf)

Theorem 35 ([8, Theorem 12])**.** *Boundedness for 2-way distance ∧ parity automata is decidable in elementary time, assuming ILT.*

If only priorities $\{0, 1\}$ are used, then boundedness is decidable in elementary time.

Together, Proposition 32, Theorem 33, and Theorem 35 imply the first part of Theorem 24. The complexity is non-elementary in general (because of the size of the automata), but elementary for fixed pdepth.

For the second part of the theorem, we are given $\boldsymbol{x}$-guarded $\phi(\boldsymbol{x}, \boldsymbol{z}, X)$ positive in X of arity $|\boldsymbol{x}|$ but with extra parameters $\boldsymbol{z}$, is there a natural number n such that for all σ-structures $\mathfrak{A}$ and for all valuations $\boldsymbol{z} \mapsto \boldsymbol{c}$, does $\phi^n(\mathfrak{A}, \boldsymbol{c}) = \phi^{n+1}(\mathfrak{A}, \boldsymbol{c})$? This situation can be handled by the argument above by treating $\boldsymbol{z}$ as constants, i.e. by viewing ϕ as a formula over the signature σ' extending σ with constants for each $z \in \boldsymbol{z}$. However, it can be shown that the automaton $\mathcal{B}_\phi$ constructed in this case uses only priorities $\{0, 1\}$, so we can use an optimized result from [8, Theorem 12] which says that boundedness can be decided in elementary time (this is not subject to any unpublished results). This means that boundedness for $\boldsymbol{x}$-guarded $\phi(\boldsymbol{x}, \boldsymbol{z}, X) \in \text{GNF}[\sigma]$ is decidable in elementary time.

F.2 Proof sketch of Theorem 26 (FO definability of negation-free CQPDL)

The goal is to prove:

It is decidable whether or not a negation-free CQPDL sentence relative to a GNF sentence can be expressed in FO.

We start by considering a C2RPQ ψ relative to GNF sentence θ. The argument consists of two main steps:

1. show that expressibility of $\psi \wedge \theta$ in FO can be reduced to a boundedness question;
2. show that this boundedness question is decidable.

Step 1. Reduce to boundedness question The Barwise-Moschovakis theorem [7] states that an FO formula $\phi(\boldsymbol{x}, X)$ is bounded iff $[\mathbf{lfp}_{X,\boldsymbol{x}} .\phi]$ can be expressed in FO. It is easy to see that this result extends to simultaneous fixpoints, but for completeness (and because we could not find an explicit statement of this in the literature), we formalize the argument here.

If we have a system S of equations $A_1, \boldsymbol{x}^1 := \phi_1, \ldots, A_n, \boldsymbol{x}^n := \phi_n$ for a simultaneous fixpoint, we say that $\phi(\boldsymbol{x}^1)$ *defines the projection of the fixpoint on* A_1 if in any structure $\mathfrak{A}$, the set $\phi_{A_1}(\mathfrak{A}) := \{\boldsymbol{a}^1 : \mathfrak{A}, \boldsymbol{a}^1 \models \phi\}$ agrees with $\{\boldsymbol{a}^1 : \exists \boldsymbol{a}^2 \ldots \boldsymbol{a}^n$ such that $\boldsymbol{a}^1 \boldsymbol{a}^2 \ldots \boldsymbol{a}^n$

is in the least fixpoint of the equations on $\mathfrak{A}$}. If the simultaneous fixpoint is bounded by some j with respect to A_1 we will say, S, A_1 *is bounded*. If S, A_1 is bounded (by j) then $[\mathbf{lfp}_{A_1,\boldsymbol{x}^1}.S]$ is clearly FO definable, since we can express the projection of the fixpoint on A_1 by the FO formula that describes the j-th approximant of the fixpoint. For the other direction:

Lemma 36. *If we have a system S of equations $A_1, \boldsymbol{x}^1 := \phi_1$, ..., $A_n, \boldsymbol{x}^n := \phi_n$ and there is a first-order $\phi(\boldsymbol{x}^1)$ that defines the projection of the fixpoint on A_1, then there is $j \in \mathbb{N}$ such that in any structure $\mathfrak{A}$, $\phi_{A_1}(\mathfrak{A})$ is $\{\boldsymbol{a}^1 : \exists \boldsymbol{a}^1 \ldots \boldsymbol{a}^n$ such that $\boldsymbol{a}^1 \boldsymbol{a}^2 \ldots \boldsymbol{a}^n$ is in the j^{th} iteration of the equations on $\mathfrak{A}\}$. In this case, we say that S, A_1* is bounded.

Proof. This is proven via a slight variation of the proof of the Barwise-Moschovakis theorem (see the outline in [25]). First it is shown that in a saturated model, every tuple that is in the projection of the fixpoint is in the projection of some finite iteration of the equations. This follows from the fact that the closure ordinal of a first-order operator is at most ω in a saturated model. Second, we note that if the projection of fixpoint does not collapse to the projection of some fixed number of iterations, we can get a tuple that does not enter the projection of the fixpoint in any finite stage, contradicting the result above. □

Thus, it suffices to show that $\psi \wedge \theta$ can be written as a single least fixpoint of a system S of equations using FO formulas and some goal predicate goal, since this will imply that $\psi \wedge \theta$ is FO definable iff S, goal is bounded.

Each $E(x, y)$ in the body of ψ can be translated into an LFP formula using a simultaneous fixpoint and no parameters. This is similar to Example 4, but we include both variables explicitly in the fixpoint as fixpoint variables, rather than as parameters. We write S_E for the system of equations coming from E, and write X_E for its designated goal predicate.

We now rewrite $\psi \wedge \theta$ as a simultaneous least fixpoint of the form $[\mathbf{lfp}_{\text{goal},\emptyset}.S]()$ (that is, the distinguished goal predicate is a nullary predicate called goal) where S is a system of the following equations:

- the equations from S_E for each $E(x, y)$ in the body of ψ;
- $Q, \emptyset := \psi'$, where Q is a fresh 0-ary variable and ψ' is the result of replacing each $E(x, y)$ in ψ with $X_E(x, y)$;
- goal, $\emptyset := Q \wedge \theta$.

(Note that it would not have been possible to combine the definitions of the $E(x, y)$ and the CQ ψ' in a single fixpoint like this if we were using x or y as parameters in S_E and these variables were quantified in ψ.)

Since $[\mathbf{lfp}_{\text{goal},\emptyset}.S]()$ is equivalent to $\psi \wedge \theta$, by (the extension of) Barwise-Moschovakis, definability in FO corresponds to this fixpoint being bounded.

Step 2. Show that this boundedness question is decidable
Unfortunately, we cannot decide boundedness directly for S, goal since $[\mathbf{lfp}_{\text{goal},\emptyset}.S]()$ is not in GNFP-UP.

However, we know that there is a GNFP-UP formula ψ' equivalent to ψ that we can relate this to.

Lemma 37. *S, goal is bounded iff there is some natural number n such that $(\psi')^n \wedge \theta$ is equivalent to $\psi' \wedge \theta$, where $(\psi')^n$ denotes the result of replacing every fixpoint in ψ' with its n-th approximant.*

Proof. Suppose that S, goal is unbounded. Then for all $n \in \mathbb{N}$, there is a structure $\mathfrak{A}_n$ satisfying the fixpoint, but requiring at least n iterations to reach the fixpoint. Consider the fixpoint game for $[\mathbf{lfp}_{\text{goal},\emptyset}.S]()$ on $\mathfrak{A}_n$ (the fixpoint game is similar to the game described in Section 6, but is adapted to simultaneous fixpoints). Since $\mathfrak{A}_n$ requires at least n iterations to reach the fixpoint, Adam must have a strategy to enforce at least n rounds in the fixpoint game. This means that regardless of the S_E valuations Eve chooses to witness ψ', Adam has a strategy to challenge some $(a_1, a_2) \in S_E$, to enforce a further $n-1$ rounds. But once Adam challenges her on some $(a_1, a_2) \in S_E$, then we are in the fixpoint game for S_E. Hence, this implies that the fixpoint related to the conjunct $E(x, y)$ in ψ requires at least $n-1$ unfoldings: $\mathfrak{A}_n \not\models (E')^{n-2}(a_1, a_2)$ where $(E')^{n-2}$ is the $n-2$-approximant of the GNFP-UP formula E' equivalent to E. That means that $(\psi')^{n-2} \wedge \theta$ is not equivalent to $\psi' \wedge \theta$ on $\mathfrak{A}_n$. Since this is true for all n, this means that the family of structures $\mathfrak{A}_n$ witnesses the fact that there is no N such that for all structures, $(\psi')^N \wedge \theta$ is equivalent to $\psi' \wedge \theta$.

Similar reasoning shows that if S, goal is bounded, then the uniform bound N for this fixpoint implies that $(\psi')^N \wedge \theta$ is equivalent to $\psi' \wedge \theta$. □

Hence, we have a new (but equivalent) boundedness question that we can try to answer: is there some natural number n such that $(\psi')^n \wedge \theta$ is equivalent to $\psi' \wedge \theta$. Because this involves GNFP-UP formulas, we can tackle these with the automata machinery developed in this paper, and some previous cost automata results. As before, we can restrict to tree-like models, and construct cost automata related to this question.

The cost automaton that we construct has a nice property: it is *counter-weak* (or *quasi-weak*), which means that in any cycle in the automaton that visits states with both odd and even priorities, there is a counter which is incremented but not reset. This is a generalization of *weak* automata which satisfy the stronger condition that there is no cycle that visits states with both odd and even priorities.

Lemma 38. *There is a 2-way alternating counter-weak cost parity tree automaton $\mathcal{B}$ such that $[\![\mathcal{B}]\!]$ is bounded iff there is some n such that $(\psi')^n \wedge \theta$ is equivalent to $\psi' \wedge \theta$ over all structures.*

Proof sketch. There are counter-weak cost automata for each conjunct in ψ': we take the automaton from Lemma 18, and just add a counter that is incremented each time the fixpoint is unfolded. They are weak (and hence counter-weak) because these automata start in states with odd priorities and once they switch to an even priority, they remain there forever (since the only states with even priorities are sink states).

Counter-weak cost automata are closed under maximum (respectively, minimum) simply by giving Adam (respectively, Eve) the choice of which conjunct automaton to simulate. These are the closure operations needed to capture conjunction and disjunction in the logic.

The operation corresponding to projection is more challenging. Without counters, we would convert to a nondeterministic automaton, and then allow Eve to guess the free variable markers. In general, it is not known how to convert alternating cost automata to an equivalent nondeterministic version and do this projection. However, counter-weak automata are closed under a special form of *weak projection* where the set being projected is finite [10] (it is called weak inf-projection there). This is all we need, since we only need to project the finite number of free variable markers. So there is a counter-weak automaton $\mathcal{B}_{\psi'}$ that (roughly speaking) computes the minimum number of unfoldings required for the fixpoints in ψ'.

On input $\mathcal{T}$, the overall automaton $\mathcal{B}$ works by giving Eve a choice to (a) show $\mathcal{T}$ is inconsistent, or (b) show that $\mathfrak{D}(\mathcal{T})$ does not satisfy $\psi' \wedge \theta$, or (c) run $\mathcal{B}_{\psi'}$. This automaton is bounded over all trees iff there is some n such that $(\psi')^n \wedge \theta$ is equivalent to $\psi' \wedge \theta$ over all structures. Because the automata checking (a) and (b) are also

weak (and hence counter-weak), the overall automaton is counter-weak as desired. □

Unlike general cost automata over infinite trees, these counter-weak automata are another special class of cost automata for which boundedness is known to be decidable.

Theorem 39 ([10, Corollary 3])**.** *Boundedness for 2-way counter-weak cost automata is decidable.*

Overall, putting these lemmas together gives us a method to construct a 2-way counter-weak cost automaton that is bounded iff S, goal is bounded, which we can decide using the previous theorem.

***Extension to negation-free* CQPDL** In Step 1, we proceed by induction on the number of nested programs/tests in the CQPDL. The base case is exactly the C2RPQ case from before. For the inductive case, the idea is to flatten the CQPDL in a similar way as in Corollary 23. For each $E(x, y)$ in the CQPDL, we translate to a C2RPQ translate$(E)(x, y)$ as in Corollary 23. This C2RPQ has new predicates $U_{t?}$ for $t?$ occurring in E. We add equations in the simultaneous fixpoint based on the automaton for translate$(E)(x, y)$, and add equations describing the meaning of each new predicate $U_{t?}$ based on the inductively defined formulas. The rest of step 1 is similar.

In Step 2, we again construct a 2-way counter-weak automaton for which boundedness is decidable. The construction of the automaton relies on the closure properties of 2-way counter-weak automata described in [10], including closure under (weak) projection, min/max, localization, etc.

It is important for both of these steps that we are in negation-free CQPDL: the flattening to a single least fixpoint in step 1 only works because all of the $U_{t?}$ occur positively, and the reduction to cost automata boundedness in step 2 only works because we are trying to minimize the unfoldings of *least* fixpoints.