The Complexity of Boundedness for Guarded Logics

Michael Benedikt
University of Oxford

Balder ten Cate
UC Santa Cruz and LogicBlox Inc

Thomas Colcombet
Université Paris Diderot

Michael Vanden Boom
University of Oxford

***Abstract*—Given a formula $\phi(\boldsymbol{x}, X)$ positive in X, the boundedness problem asks whether the fixpoint induced by ϕ is reached within some uniform bound independent of the structure (i.e. whether the fixpoint is spurious, and can in fact be captured by a finite unfolding of the formula). In this paper, we study the boundedness problem when ϕ is in the guarded fragment or guarded negation fragment of first-order logic, or the fixpoint extensions of these logics. It is known that guarded logics have many desirable computational and model theoretic properties, including in some cases decidable boundedness. We prove that boundedness for the guarded negation fragment is decidable in elementary time, and, making use of an unpublished result of Colcombet, even 2EXPTIME-complete. Our proof extends the connection between guarded logics and automata, reducing boundedness for guarded logics to a question about cost automata on trees, a type of automaton with counters that assigns a natural number to each input rather than just a boolean.**

I. Introduction

A standard technique for analyzing the behavior of logics is to translate from logics to *automata*. Such translations are fundamental in deciding the satisfiability and model-checking problems for modal and temporal logics and for the modal fixpoint logic, the μ-calculus [1].

In this work, we consider a family of first-order languages that have been the focus of much research in the past decades — guarded logics. The guarded fragment of first-order logic (GF) restricts quantification to occur in the context of a guard predicate that contains all free variables. In the guarded negation fragment of first-order logic (GNF), existential quantification is unrestricted, but uses of negation must be under the scope of a guard. The interest in these logics stems from their expressiveness — they can express the predicate logic encodings of many modal logics, as well as integrity constraint and mapping languages that come from databases — and their attractive properties for analysis. Satisfiability and validity are decidable for these logics, and they also have the finite model property — any implication that is valid over finite models is valid over all models ([2], [3]). The decidability results extend to their fixpoint extensions, guarded fixpoint logic (GFP) and guarded negation fixpoint logic (GNFP) ([4], [3]).

What is more, finer analysis problems are possible for the fixpoint logic extensions, including an analysis of *boundedness*. Informally, a fixpoint formula is bounded if the number of times that the fixpoint operators are unfolded can be bounded independently of the input. For certain logics boundedness is known to be equivalent to first-order definability.

The work of Blumensath et al. [5] studies the complexity of the boundedness problems of arbitrary monadic second-order logic (MSO) formulas over tree inputs, showing that the complexity is below a tower of exponentials that is polynomial in the formula. Their proof relies on results about *cost automata* — a variant of tree automata that produce a value from $\mathbb{N} \cup \{\infty\}$, rather than a boolean. They reduce the boundedness question for an MSO formula to the *limitedness problem* for a cost automaton, where the limitedness problem asks whether the value given by the automaton on an accepted tree has a bound independent of the input. Using this master result as well as MSO interpretations and model theoretic transfer results, they are able to show boundedness is decidable for a number of other logics. In particular, they show that boundedness is decidable for GFP, subsuming an earlier result of Otto [6] that μ-calculus boundedness is decidable.

The boundedness problem was studied originally for variants of the language Datalog (e.g. [7], [8], [9]), which are restrictions of fixpoint logic. Bárány et. al. [10] build on the results of [5] to show that boundedness is decidable for the language GN-Datalog, which restricts GNFP by allowing negation to occur only in a "stratified" manner. No complexity bound is given for the boundedness problem in [10], and the techniques provided there do not suffice to give an elementary bound, for two reasons. First, the reduction to problems on trees in [10] is non-elementary, making use of the machinery of MSO interpretations. Secondly, our understanding of quantitative problems for cost automata on trees is incomplete. Hence, a finer analysis is needed to show an elementary bound for guarded logics over arbitrary structures.

In this work we return to the issue of boundedness of guarded logics. We first revisit the logic-to-automata connection with boundedness in mind. We define a cost analog of GNFP, called cGNFP, which defines functions from structures to $\mathbb{N} \cup \{\infty\}$. The boundedness problem for the quantitative logic cGNFP is to decide whether the value given by a cGNFP formula is bounded independently of the input: it is easily seen to subsume the natural boundedness questions concerning unfolding of fixpoints in GNFP. We give a reduction from a cGNFP formula ϕ to a 2-way alternating cost automaton $\mathcal{A}_\phi$ over unranked trees such that ϕ is bounded iff the value of $\mathcal{A}_\phi$ is bounded over all inputs. We isolate a subclass of cGNFP formulas that suffices to capture the counting of the unfoldings of a single fixpoint, and show that the corresponding automata are of a much simpler form.

We then provide a finer complexity analysis of the boundedness problem for cost automata on trees, focusing on the class of automata that arise from the subclass of cGNFP mentioned above. We provide more details on the cost automata results that are being used implicitly in [5] and [10], and develop some new cost automata results tailored to this problem.

Combining these two components, we derive improved results about the complexity of boundedness for guarded logics. In particular, boundedness for GNF and GN-Datalog is decidable in elementary time, and even 2EXPTIME-complete using an unpublished result due to Colcombet [11]. As a by-product of our analysis, we get a self-contained proof of the 2EXPTIME bound on the satisfiability problem for GNFP via automata-theoretic methods.

Organization. In Sections II and III, we provide background information on guarded logics and cost automata. The main technical contributions follow: we translate from cGNFP to 2-way cost automata in Section IV, and then in Section V describe new technical results for efficiently deciding boundedness for a subclass of these cost automata. Finally, in Section VI we use this logic-automata connection to derive results on the complexity of boundedness for guarded logics.

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

II. PRELIMINARIES

Notation and conventions. We write $\mathbb{N}_\infty$ for $\mathbb{N} \cup \{\infty\}$, the set of natural numbers extended with a special ∞ symbol.

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

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

Basics of guarded logics. The *Guarded Negation Fragment* of FO (denoted GNF) is built up according to the grammar:

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

where R is either a relation symbol or the equality relation, α is an atomic relation (including equality), and $\boldsymbol{t}$ is a tuple over variables and constants. Notice that any use of negation must occur conjoined with an atomic relation that contains all the free variables of the negated formula. Such an atomic relation is a *guard* of the formula. We write GNF$[\sigma]$ to denote GNF formulas over some particular signature σ.

The purpose of allowing equalities as guards is to ensure that every formula with at most one free variable can always be guarded. Thus GNF includes the *Unary Negation Fragment* (UNF) [12], which is built up as above, but allowing negation only on formulas with at most one free variable.

GNF should be compared to the *Guarded Fragment* (GF):

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

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

We write $\mathrm{gdd}(\boldsymbol{x})$ for the *guardedness predicate* asserting that $\boldsymbol{x}$ is guarded by some atom. This can be understood as an abbreviation for the disjunction of existentially quantified relational atoms involving all of the variables in $\boldsymbol{x}$. We say that a formula $\phi(\boldsymbol{x})$ is *answer-guarded* if it is logically equivalent to $\mathrm{gdd}(\boldsymbol{x}) \wedge \phi(\boldsymbol{x})$.

It is easy to see that every *union of conjunctive queries* (UCQ) is expressible in GNF. It is only slightly more difficult to verify that every answer-guarded GF formula can be expressed in GNF [3]. Not only is GNF an expressive fragment of FO, but it was shown to be decidable and to have the finite model and tree-like model properties (see Theorem 2).

Like modal logic, the fixpoint extensions of these guarded logics are decidable and have nice model theoretic properties too. *Guarded Negation Fixpoint Logic* (denoted GNFP) and *Guarded Fixpoint Logic* (denoted GFP) can be defined as the extensions of GNF and GF, respectively, with formulas

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

where (i) X only appears positively in ϕ, (ii) second-order variables like X cannot be used as guards and (iii) $\mathrm{gdd}(\boldsymbol{x})$ is the guardedness predicate expressing that $\boldsymbol{x}$ is guarded by an atom from the original signature.[1] We briefly review the semantics of this fixpoint operator now, and introduce some notation that we will use to talk about boundedness later. Since $\phi(\boldsymbol{x}, X, \boldsymbol{Y})$ is monotone in X, it induces an operator $U \mapsto \mathcal{O}_\phi^{\mathfrak{A},\boldsymbol{V}}(U) := \{\boldsymbol{a} : \mathfrak{A}, U, \boldsymbol{V} \models \mathrm{gdd}(\boldsymbol{a}) \wedge \phi(\boldsymbol{a}, X, \boldsymbol{Y})\}$ on every structure $\mathfrak{A}$ with valuation $\boldsymbol{V}$ for $\boldsymbol{Y}$, and this operator has a least fixpoint. Given some ordinal β, the *fixpoint approximant* $\phi^\beta(\mathfrak{A}, \boldsymbol{V})$ of ϕ on $\mathfrak{A}, \boldsymbol{V}$ is defined such that

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

We let $\phi^\infty(\mathfrak{A}, \boldsymbol{V}) := \bigcup_\beta \phi^\beta(\mathfrak{A}, \boldsymbol{V})$ denote the least fixpoint based on this operation, and the least ordinal β such that $\phi^\beta(\mathfrak{A}, \boldsymbol{V}) = \phi^{\beta+1}(\mathfrak{A}, \boldsymbol{V}) = \phi^\infty(\mathfrak{A}, \boldsymbol{V})$ is called the *closure ordinal*. Thus, $[\mu X, \boldsymbol{x}.\,\mathrm{gdd}(\boldsymbol{x}) \wedge \phi(\boldsymbol{x}, X, \boldsymbol{Y})]$ defines

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

a new predicate named X of arity $|\boldsymbol{x}|$, and $\mathfrak{A}, \boldsymbol{V}, \boldsymbol{a} \models [\mu X, \boldsymbol{x}.\,\mathrm{gdd}(\boldsymbol{x}) \wedge \phi(\boldsymbol{x}, X, \boldsymbol{Y})](\boldsymbol{x})$ iff $\boldsymbol{a} \in \phi^{\infty}(\mathfrak{A}, \boldsymbol{V})$. If $\boldsymbol{V}$ is empty or understood in context, we just write $\phi^{\infty}(\mathfrak{A})$.

We also write ϕ^{β} for the fixpoint approximations obtained by unfolding the fixpoint β times. That is, $\phi^{0} := \bot$, $\phi^{\beta+1}(\boldsymbol{x}) := \mathrm{gdd}(\boldsymbol{x}) \wedge \phi[\phi^{\beta}(\boldsymbol{y})/X(\boldsymbol{y})]$, and if β is a limit ordinal, $\phi^{\beta} := \bigvee_{\beta' < \beta} \phi^{\beta'}$. This formula defines the β-approximation of the fixpoint process based on $[\mu X, \boldsymbol{x}.\,\mathrm{gdd}(\boldsymbol{x}) \wedge \phi(\boldsymbol{x}, X, \boldsymbol{Y})]$. In general these formulas are not in GNFP, but instead live in an infinitary version of GNFP that allows conjunctions and disjunctions over arbitrary sets of formulas. However, for finite β, it is straightforward to check that ϕ^{β} is in GNFP and is the same width as ϕ.

Consider the following example, due to the authors of [3].

Example 1. *Let φ be the* GNF *formula*

$$\exists y_1 y_2.(R_1 x_1 y_1 \wedge R_2 x_2 y_2 \wedge X y_1 y_2) \vee (R_1 x_1 x_1 \wedge R_2 x_2 x_2)$$

where R_1 and R_2 are two binary relations, and X is a fixpoint variable. Then $[\mu X, x_1 x_2.\,\mathrm{gdd}(x_1 x_2) \wedge \varphi](y_1 y_2)$ is in GNFP *and expresses the existence of a "ladder" consisting of R_1 and R_2 paths of the same length starting from y_1 and y_2, ending in self-loops, and such that the pair of elements on each rung are guarded.*

We will actually allow *simultaneous fixpoints*, or *vectorial fixpoints*, of the form $[\mu X_i, \boldsymbol{x}_i.S](\boldsymbol{t})$ where

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

is a system of formulas ϕ_i where $X_1, \ldots, X_j$ occur positively, and satisfy the same requirements for the body of the fixpoint formulas as before. Such a system defines a monotone operation on vectors of relations, and $[\mu X_i, \boldsymbol{x}_i.S](\boldsymbol{t})$ expresses that $\boldsymbol{t}$ is a tuple in the i-th component of the least fixpoint defined by this operation. Allowing simultaneous fixpoints does not change the expressivity of GFP or GNFP, since they can be eliminated in favor of traditional fixpoints [13], with a possible exponential blow-up in the size of the formula, but only a polynomial blow-up if a DAG-representation is used.

The following theorem summarizes the decidability and model theoretic results about these guarded logics that we will make use of. The tree-like model property is particularly important in this work, since it opens up the use of tree automata techniques to reason about these logics.

Theorem 2 ([3])**.** *Satisfiability (and even finite satisfiability) is* 2ExpTime*-complete for* GNF *and* GNFP.

GNF *has the finite-model property: if ϕ is satisfiable, then ϕ is satisfiable in a finite structure. This does not hold for* GNFP.

GNF *and* GNFP *have the tree-like model property: if ϕ is satisfiable, then ϕ is satisfiable over structures of tree-width* $(\mathrm{width}(\phi) - 1)$.

Datalog programs. The guarded logics considered in this paper are related to several variants of the query language Datalog. Boundedness of Datalog is an important optimization problem that has been heavily studied within the database community (e.g. [7], [8], [9]), and it is closely related to static analysis problems concerning querying with respect to constraints [14]. We briefly recall the definition of Datalog, and then define the types of Datalog programs that interest us.

A *Datalog program* is specified by $\Pi = \langle \mathrm{EDB}^{\Pi}, \mathrm{IDB}^{\Pi}, \mathrm{Rules}^{\Pi} \rangle$ where the extensional predicates EDB^{Π} and intensional predicates IDB^{Π} are disjoint sets, and Rules^{Π} consists of formulas of the form $R(x_1 \ldots x_n) \leftarrow \psi(\boldsymbol{xy})$ where R is an IDB predicate and ψ is a conjunction of atoms.

A program defines a transformation taking a structure $\mathfrak{A}$ interpreting each relation $R \in \mathrm{EDB}^{\Pi} \cup \mathrm{IDB}^{\Pi}$ by a set of tuples $\mathfrak{A}(R)$, and outputting a structure $\mathfrak{A}'$ interpreting relation R by $\mathfrak{A}(R) \cup \bigcup \{t_1 \ldots t_n \mid \mathfrak{A}, \boldsymbol{t} \models \exists \boldsymbol{y}.\psi(\boldsymbol{xy})\}$ where the union is over all rules with R the relation on the left. If Π is a program and $\mathfrak{A}$ is a structure over the signature EDB^{Π}, then we write $\Pi^{k}(\mathfrak{A})$ for the structure obtained by iterating this transformation k times starting from the $\mathrm{EDB}^{\Pi} \cup \mathrm{IDB}^{\Pi}$ structure extending $\mathfrak{A}$ by letting all intensional predicates be empty. We let $\Pi^{\infty}(\mathfrak{A})$ denote the structure obtained by running the transformation on the structure above until a fixpoint is reached; that is $\bigcup_j \Pi^{j}(\mathfrak{A})$.

A *Datalog program with stratified negation* is a tuple $\langle \Pi_1, \ldots, \Pi_j \rangle$ of Datalog programs $\Pi_i := \langle \mathrm{EDB}^{\Pi_i}, \mathrm{IDB}^{\Pi_i}, \mathrm{Rules}_i \rangle$ such that for all $1 < i \leq j$, $\mathrm{EDB}^{\Pi_i} := \mathrm{EDB}^{\Pi_{i-1}} \cup \mathrm{IDB}^{\Pi_{i-1}}$, and rules in Rules_i may include conjuncts that are negated atomic formulas as long as the predicates in these negated atoms are from EDB^{Π_i} and the variables appear positively somewhere in the rule. The semantics of Datalog extends by defining $\Pi(\mathfrak{A})$ as $\Pi_j^{\infty}(\Pi_{j-1}^{\infty}(\ldots(\Pi_1^{\infty}(\mathfrak{A}))\ldots))$.

We are interested in a subclass of Datalog programs with stratified negation where negation is guarded. A GN-*Datalog program* is a Datalog program with stratified negation $\langle \Pi_1, \ldots, \Pi_j \rangle$ where each rule $R(x_1 \ldots x_n) \leftarrow \psi(\boldsymbol{xy})$ in Π_i is *negation guarded*: there is a positive EDB^{Π_i} atom in ψ that guards $x_1 \ldots x_n$, and for every atomic formula $\alpha(\boldsymbol{z})$ that appears negated in ψ, there is a positive EDB^{Π_i} atom in ψ that guards $\boldsymbol{z}$. This ensures each rule can be expressed in GNF.

Note that Datalog programs can easily be expressed using simultaneous fixpoints, and GN-Datalog programs can be expressed using GNFP. We remark also that *Monadic Datalog* (Datalog where all IDB predicates are unary) is contained in GN-Datalog, since Monadic Datalog does not allow any use of negation in the body of the rule, and the unary IDB predicates are trivially guarded.

Finally, a GNF-*program* $\Pi = \langle \mathrm{EDB}^{\Pi}, \mathrm{IDB}^{\Pi}, \mathrm{Rules}^{\Pi} \rangle$ is a program where EDB^{Π} and IDB^{Π} are disjoint sets of relation names, and Rules consists of formulas of the form $R(x_1 \ldots x_n) \leftarrow \psi(\boldsymbol{xy})$ where ψ is an answer-guarded GNF formula that uses IDB predicates only positively.

Boundedness. Given a formula $\phi(\boldsymbol{x}, X)$ that is positive in X, we say that ϕ is *bounded* over a class $\mathcal{C}$ of structures if there exists some ordinal $\beta < \omega$ such that $\phi^{\beta}(\mathfrak{A}) = \phi^{\beta+1}(\mathfrak{A}) =$

$\phi^\infty(\mathfrak{A})$ for all $\mathfrak{A} \in \mathcal{C}$. In other words, across all structures in $\mathcal{C}$, the fixpoint for ϕ is reached after at most β unfoldings. For instance, φ in Example 1 is unbounded: consider the family of structures corresponding to "ladders" of unbounded size. The *boundedness problem* for a logic L over a class $\mathcal{C}$, asks: given some $\phi \in L$, is ϕ bounded over $\mathcal{C}$?

Similarly, we say that a Datalog program $\Pi = \langle \Pi_1, \ldots, \Pi_j \rangle$ is *bounded* over a class $\mathcal{C}$ of structures if there is some $n \in \mathbb{N}$, such that $\Pi^n(\mathfrak{A}) = \Pi^{n+1}(\mathfrak{A})$ for all $\mathfrak{A} \in \mathcal{C}$. A Datalog program with stratified negation Π is *fully bounded* over $\mathcal{C}$ if each stratum is bounded over $\mathcal{C}$ by evaluating all IDB predicates from lower strata according to the program. Equivalently, Π is fully bounded over $\mathcal{C}$ if there are $n_1, \ldots, n_j \in \mathbb{N}$ such that $\Pi^\infty(\mathfrak{A}) = \Pi_j^{n_j}(\Pi_{j-1}^{n_{j-1}}(\ldots(\Pi_1^{n_1}(\mathfrak{A}))\ldots))$ for all $\mathfrak{A} \in \mathcal{C}$.

In general, boundedness for Datalog programs is undecidable [7]. A significant exception is boundedness for monadic Datalog programs, which was shown to be decidable by Cosmadakis, Gaifman, Kanellakis, and Vardi in [8] We refer the interested reader to Blumensath et al. [5], which includes a survey of other positive results. In particular, [5] proved that boundedness is decidable for GF and GFP. These results follow from a master theorem about the decidability of boundedness for *guarded second-order logic* over structures of bounded tree-width, which in turn relies on results about cost automata over infinite trees by Colcombet and Löding.

Using the master result in [5], Bárány et al. [10] prove that boundedness is decidable for GNF-programs and answer-guarded GNF, and full boundedness is decidable for GN-Datalog. Moreover, for both of these cases, boundedness over all structures coincides with (i) boundedness over finite structures, (ii) boundedness over structures of bounded tree-width, (iii) rewriteability in GNF, and (iv) rewriteability in FO.

Unfortunately, going through the master theorem in [5] yields only non-elementary complexity bounds. Our contribution in this paper is to provide direct automata constructions that yield elementary bounds for the cases above and other related boundedness problems. In some cases, this even enables us to provide tight bounds on the complexity of boundedness.

III. Cost automata

Overview. This section provides background information on the types of automata used in this paper. These automata are extensions of traditional tree automata with some exotic features including counters, 2-way movement, and the ability to work on trees with arbitrary branching.

Definition. A cost automaton is a traditional finite state automaton (on words, trees, etc.) that has been enriched with a finite set of counters that can be manipulated on each transition, and that are used to assign a value from $\mathbb{N}_\infty$ to each input structure. We will view the operation of an automaton on an input as a game between players Eve and Adam.

In place of a traditional acceptance or winning condition, we use an *objective* $\mathrm{Obj} = \langle \mathrm{Act}, f, \mathrm{goal} \rangle$. The objective describes the possible actions Act on each transition, the function f that reads a sequence of these actions and assigns a value from $\mathbb{N}_\infty$, and a $\mathrm{goal} \in \{\min, \max\}$ that says whether Eve is trying to minimize or maximize this value from f. We write $\overline{\mathrm{Obj}}$ for the *dual objective* that results from exchanging $\min$ for $\max$ in Obj; this describes the goal for Adam.

For instance, the well-known parity acceptance condition using some finite set of priorities Pri could be captured in a *parity objective* of the form $\langle \mathrm{Pri}, \mathrm{cost}_{\text{parity}}, \mathrm{goal} \rangle$ where $\mathrm{goal} = \min$ (respectively, $\mathrm{goal} = \max$) and $\mathrm{cost}_{\text{parity}}$ maps a sequence of priorities from Pri to 0 (respectively, ∞) if the maximum priority that occurs infinitely often is even, and to ∞ (respectively, 0) otherwise.

In this work, we need to make use of a number of different objectives based on counters and the parity condition, so we define a generic cost automaton model. For now, just think of the objective as a way to evaluate a run of the automaton, in the same way an acceptance condition is usually used.

An *(alternating) cost automaton* $\mathcal{A}$ on Σ-labelled trees is a tuple $\langle \Sigma, Q, q_0, \mathrm{Dir}, \mathrm{Obj}, \delta \rangle$ where Q is a finite set of states partitioned into states owned by Eve and states owned by Adam, Σ is a finite alphabet of node labels, $q_0 \in Q$ is the initial state, Dir describes the possible directions for a move, and $\mathrm{Obj} = \langle \mathrm{Act}, f, \mathrm{goal} \rangle$ describes the objective. The transition function has the form $\delta : Q \times \Sigma \to \mathcal{P}^+(\mathrm{Dir} \times \mathrm{Act} \times Q)$ where $\mathcal{P}^+(S)$ denotes the non-empty subsets of S. The notation $\delta(p, a) = \bigvee_{i \in I}(d_i, c_i, q_i)$ (resp. $\bigwedge_{i \in I}(d_i, c_i, q_i)$) is a concise notation for the fact that the state p is owned by Eve (resp. Adam), and $\delta(p, a) = \{(d_i, c_i, q_i) \ : \ i \in I\}$. We sometimes use more complex positive Boolean combinations as shorthand for describing several transitions and states in one step.

Notice that we make explicit the set Dir of *directions* that the automaton uses as it traverses an input tree. A direction $d \in \mathrm{Dir}$ is a function that maps a position in the tree to the set of positions that are accessible in that direction. We will represent these directions in the usual way. For instance, a traditional top-down tree automaton working on m-ary trees with ordered children uses directions $\{1, \ldots, m\}$. Sometimes we will be working with 2-way automata on m-ary trees that can move up or remain in the same position in the tree, similar to the automata in [15]; for these automata, we use directions $\{-1, 0, 1, \ldots, m\}$, where 0 denotes staying in the same position, and -1 denotes moving up to the parent of the current node. In the next section, we will actually be working with automata on infinite, unranked, unordered trees of arbitrary—possibly infinite—degree that are similar to the automata in [4]; for these trees, we will use only directions 0 (stay in the same position) and $\updownarrow$ (move to some neighbor, in any direction). Note that in such a case, there are several possible successor nodes in the tree, and the owner of the origin state decides which one to choose.

In general, with this sort of transition function $\mathcal{A}$ is an *alternating automaton*. We view $\mathcal{A}$ running on a tree t starting at node $v_0 \in \mathrm{dom}(t)$ as a game $\mathcal{G}(\mathcal{A}, t, v_0)$. The arena is $Q \times \mathrm{dom}(t)$, with initial position (q_0, v_0). From position (q, v),

- the player that owns state q selects $\chi \in \delta(q, t(v))$,
- the other player selects some (d, c, r) in χ,

- the player that owns state q selects a neighbor w of v in direction d.

The output from this move is c, and the game continues from position (r, w). The corresponding move is written $(c, (r, w))$. A *play* in $\mathcal{G}(\mathcal{A}, t, v_0)$ is a sequence $(q_0, v_0), (c_1, (q_1, v_1)), (c_2, (q_2, v_2)), \ldots$ of moves in the game.

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

If the objective $\mathrm{Obj} = \langle \mathrm{Act}, f, \mathrm{goal} \rangle$ uses $\mathrm{goal} = \min$ (respectively, $\mathrm{goal} = \max$), then an *n-winning strategy* for Eve is a strategy such that the value from f on the output of any play consistent with the strategy is at most n (respectively, at least n). We define $[\![\mathcal{A}]\!]_{v_0}(t)$ to be

$$\mathrm{op}\,\{n : \text{Eve has an } n\text{-winning strategy in } \mathcal{G}(\mathcal{A}, t, v_0)\}\,.$$

where $\mathrm{op} = \inf$ (respectively, $\sup$) if $\mathrm{goal} = \min$ (respectively, $\mathrm{goal} = \max$). If the starting node is the root ϵ, we often write $\mathcal{G}(\mathcal{A}, t)$ and $[\![\mathcal{A}]\!](t)$ instead of $\mathcal{G}(\mathcal{A}, t, \epsilon)$ and $[\![\mathcal{A}]\!]_\epsilon(t)$. We say that $[\![\mathcal{A}]\!]$ is the function defined by $\mathcal{A}$.

We will mainly be interested in automata with objectives related to counters. The elementary actions on counters that we will use are *increment* $\mathtt{i}$, *reset* $\mathtt{r}$, *check* $\mathtt{c}$, and *no change* ε. Consider a single counter γ. It initially has value 0, and then takes values from $\mathbb{N}$ based on some sequence u of actions from $\{\mathtt{i}, \mathtt{r}, \mathtt{c}, \varepsilon\}$. The meaning of $\mathtt{i}$, $\mathtt{r}$, and ε is as expected. The check operation $\mathtt{c}$ does not change the counter value, but instead indicates when we are interested in the value of γ. We let $C_\gamma(u)$ denote the set of values at the moment(s) in the sequence u when the counter γ is checked. For instance, $C_\gamma(\mathtt{iiiri}\varepsilon\mathtt{ic}) = \{2\}$ and $C_\gamma(\mathtt{icri}^2\mathtt{cri}^3\mathtt{cr}\ldots) = \{1, 2, 3, \ldots\}$. Likewise, for a sequence u of actions from $\{\mathtt{i}, \mathtt{r}, \mathtt{c}, \varepsilon\}^\Gamma$ describing actions on a set Γ of counters we write $C_\Gamma(u)$ for $\bigcup_{\gamma \in \Gamma} C_\gamma(u)$.

The B-*objective* over some set Γ of counters is $\langle \{\mathtt{ic}, \mathtt{r}, \varepsilon\}^\Gamma, \mathrm{cost_B}, \min \rangle$ where $\mathrm{cost_B}$ maps a sequence u of actions to $\sup C_\Gamma(u)$. Notice that $\mathtt{ic}$ is an atomic action (so the counter is checked every time it is incremented). With the B-objective, Eve is trying to minimize the counter values. In the special case when there is only a single counter with actions $\mathtt{ic}$ and ε (no $\mathtt{r}$), we call this the *distance objective*, and denote it simply as 'dist'.

The S-*objective* over some set Γ of counters is $\langle \{\mathtt{i}, \mathtt{r}, \mathtt{cr}\}^\Gamma, \mathrm{cost_S}, \max \rangle$ where $\mathrm{cost_S}$ maps a sequence u of actions to $\inf C_\Gamma(u)$. In this case, Eve is trying to maximize the checked counter values.

We will primarily be interested in objectives that combine these counter conditions with a parity condition. Given objectives $O_1 = \langle \mathrm{Act}_1, f_1, \mathrm{goal} \rangle$ and $O_2 = \langle \mathrm{Act}_2, f_2, \mathrm{goal} \rangle$, we define $O_1 \wedge O_2$ as the objective $\langle \mathrm{Act}_1 \times \mathrm{Act}_2, \max(f_1, f_2), \min \rangle$ if $\mathrm{goal} = \min$, and $\langle \mathrm{Act}_1 \times \mathrm{Act}_2, \min(f_1, f_2), \max \rangle$ if $\mathrm{goal} = \max$. For example, with the B $\wedge$ parity objective, the goal of Eve is both to satisfy the parity condition and to ensure the counter values remain as low as possible. Likewise, with the S $\wedge$ parity objective, the goal of Eve is to satisfy the parity condition and to maximize the checked counter values. We say that the automaton is, e.g., a B $\wedge$ *parity automaton* if it is a cost automaton with a B $\wedge$ parity objective.

There are a few special types of automata that are worth highlighting now. A cost automaton on m-ary trees is *nondeterministic* if it is a 1-way automaton using directions $\mathrm{Dir} = \{1, \ldots, m\}$ and for all $q \in Q$ and $a \in \Sigma$, $\delta(q, a)$ is of the form $\bigvee_{i \in I}(1, c_1^i, q_1^i) \wedge \cdots \wedge (m, c_m^i, q_m^i)$ (with a slight abuse of notations). A B $\wedge$ parity automaton is *weak* if there is no cycle that visits both an even and an odd priority (this is equivalent to the classical notion of weakness [16]). It is called *counter-weak* or *quasi-weak* if for any cycle in the automaton that visits both an even and an odd priority, there is a counter that is incremented but not reset. These weak variants of the automata have the nice property that they are equivalent to automata using only two priorities, and it is equivalent for the priorities to be taken from either $\{1, 2\}$ or $\{0, 1\}$.

Cost functions. We have seen that cost automata define functions with range $\mathbb{N}_\infty$. The main question we will ask about the function $[\![\mathcal{B}]\!]$ defined by a cost automaton $\mathcal{B}$ is whether or not it is bounded: we say that a function f is *bounded* over some domain $\mathcal{D}$ (usually trees over some finite alphabet) if there is $n \in \mathbb{N}$ such that $f(t) \leq n$ for all $t \in \mathcal{D}$. The *boundedness question* asks whether the function $[\![\mathcal{B}]\!]$ defined by some cost automaton is bounded over all structures in its domain. This is related to the *limitedness question* which asks whether the function defined by some cost automaton $\mathcal{B}$ is bounded over the domain of accepted structures. We choose to present our results in terms of the boundedness question rather than limitedness, but this is a presentation decision.

One important idea in the work on cost automata is that we do not care about exact values of the functions. Indeed, we are interested only in boundedness questions, so when we convert between different types of automata and logic, we are not interested in preserving the value exactly; instead, we are interested in preserving the boundedness properties.

For this purpose, we write $[\![\mathcal{B}]\!] \preccurlyeq [\![\mathcal{B}']\!]$ if for all sets U (over some domain $\mathcal{D}$ that will usually be implicit in the context), if $[\![\mathcal{B}']\!]$ is bounded on U, then $[\![\mathcal{B}]\!]$ is bounded on U. We write $[\![\mathcal{B}]\!] \approx [\![\mathcal{B}']\!]$ iff $[\![\mathcal{B}]\!] \preccurlyeq [\![\mathcal{B}']\!]$ and $[\![\mathcal{B}']\!] \preccurlyeq [\![\mathcal{B}]\!]$.

A *regular cost function* is the equivalence class for $\approx$ of a function computed by the above automata. Regular cost functions yield a theory that parallels and subsumes the one of regular languages ([17], [18]). In particular, regular cost functions, like regular languages, have many nice closure properties and computational properties.

For cost automata over finite words and finite trees, it was shown $[\![\mathcal{A}]\!] \preccurlyeq [\![\mathcal{B}]\!]$ is decidable when $\mathcal{A}$ and $\mathcal{B}$ are defined by B-automata or S-automata ([17], [19]). This implies decidability of boundedness and limitedness, since boundedness is equivalent to $[\![\mathcal{A}]\!] \preccurlyeq 0$, and limitedness is equivalent to $[\![\mathcal{A}]\!] \preccurlyeq [\![\mathcal{A}']\!]$, where $\mathcal{A}'$ is $\mathcal{A}$ stripped of counter actions.

For cost automata over infinite trees much less is known. One positive decidability result follows.

Theorem 3 ([20], [21])**.** *Let $\mathcal{S}_{\text{nd}}$ and $\mathcal{B}_{\text{nd}}$ be nondeterministic* S $\wedge$ *parity and* B $\wedge$ *parity automata, respectively. Then $[\![\mathcal{S}_{\text{nd}}]\!] \preccurlyeq [\![\mathcal{B}_{\text{nd}}]\!]$ is decidable in elementary time. In the special case when testing boundedness for $[\![\mathcal{S}_{\text{nd}}]\!]$ (i.e. testing $\mathcal{S}_{\text{nd}} \preccurlyeq 0$), the complexity is polynomial in the number of states and exponential in the number of priorities and counters of $\mathcal{S}_{\text{nd}}$.*

The complexity is not stated in [21, Theorem 4.32], but the proof yields the elementary time bound. In this work, we only need the special case when testing boundedness for $\mathcal{S}_{\text{nd}}$. We provide some more details for this case in the full version.

We remark that most of the work on cost functions so far has focused only on decidability issues. This is one reason why the complexity has not been examined closely yet. In this paper we utilize and extend some of the recent advancements in the theory of regular cost functions over trees in order to answer boundedness questions about guarded logics. Indeed, this work provides additional motivation to analyze more closely the complexity of various constructions for cost automata.

Building-block results about cost automata on words. We conclude this section by summarizing some additional results about cost automata on words. As usual, we can view these as a special case of automata on trees where the branching degree is at most one. Automata on words play an important role in simulation and complementation constructions for automata on trees. This is true in the cost setting as well, so we describe some results for cost automata on words that we use as building blocks later. Some of these results are also implicitly used in [5] and [10].

One important difference between cost automata and traditional automata is that cost automata on words cannot in general be made deterministic. However, they can be made *history-deterministic*, a weakening of normal determinism [17]. We do not focus on the exact definition here, but the crucial property of history-deterministic automata is that they can be used safely in constructions on trees that require running an automaton on words over every branch (see [19]).

The following theorems are the crucial "black-box" results that we are using from the theory of regular cost functions that has been developed over words. These theorems can be viewed as non-trivial generalizations of the determinization and complementation results for traditional automata. We refer to these as *dualization results* since they convert between the dual B- and S-forms of cost automata.

Theorem 4 (**Dual-Finite**, [17])**.** *Let $\mathcal{B}$ be a* B*-automaton on finite words. Then there is a history-deterministic* S*-automaton $\mathcal{S}$ on finite words such that $[\![\mathcal{B}]\!] \approx [\![\mathcal{S}]\!]$, and $\mathcal{S}$ can be constructed from $\mathcal{B}$ in elementary time.*

This elementary complexity bound is given in the published paper [17, Theorem 1]. This involved proof relies on an algebraic characterization of these regular cost functions over words in terms of *stabilization monoids* (see [17] and [18]). Converting from B to S relies on converting to a stabilization monoid as an intermediate step. An improved single exponential bound is shown in the unpublished paper [11, Theorem 1], where a construction is given that mimics Safra's determinization construction ([22], [23]) and that outputs a history-deterministic S-automaton of exponential size.

Theorem 5 (**EXP-Dual-Finite**, [11])**.** *Let $\mathcal{B}$ be a* B*-automaton on finite words. Then there is a history-deterministic* S*-automaton $\mathcal{S}$ on finite words such that $[\![\mathcal{B}]\!] \approx [\![\mathcal{S}]\!]$, and $\mathcal{S}$ can be constructed from $\mathcal{B}$ in* EXPTIME *such that it has $m!(k(m+1))^{m+1}$ states and km counters, where m is the number of states and k is the number of counters in $\mathcal{B}$.*[2]

In order to make clear where we use this result, we mark uses of the previous theorem with (EXP-Dual-Finite).

Finally, it is mentioned at the end of [17] that the results about cost automata over finite words can also be generalized to the infinite word case, using suitable algebraic notions.

Theorem 6 (**Dual-Infinite**, [17])**.** *Let $\mathcal{B}$ be a* B $\wedge$ *parity automaton on words. Then there is a history-deterministic* S $\wedge$ *parity automaton $\mathcal{S}$ such that $[\![\mathcal{B}]\!] \approx [\![\mathcal{S}]\!]$, and $\mathcal{S}$ can be constructed from $\mathcal{B}$ in elementary time.*

Again, because the details for this infinite case have not been published, we mark applications of this theorem with (Dual-Infinite). We remark that the results from [5] and [10] stated earlier rely on (Dual-Infinite). In [10], this assumption (in fact, a more general assumption) was marked (ILT).

IV. REDUCTION TO COST AUTOMATA BOUNDEDNESS

Our approach differs from previous work in that rather than reducing problems to boundedness questions for cost automata, we bring the cost capabilities to guarded logics.

cGNFP. We are now ready to define a variant of GNFP called *cost* GNFP (written cGNFP). Sentences in this logic define functions from structures to $\mathbb{N}_\infty$, and the value assigned to a structure captures quantitative information about some least fixpoint subformulas. The idea is that cGNFP is a language for expressing boundedness problems for guarded logics. Indeed, we will see in Section VI that we can reduce a variety of boundedness problems for guarded logics to testing whether the function defined by a cGNFP formula is bounded.

Formally, cGNFP is an extension of GNFP with a *bounded expansion operator* μ^N that allows the formation of predicates $[\mu^N X, \boldsymbol{x}.\, \mathrm{gdd}(\boldsymbol{x}) \wedge \psi(\boldsymbol{x}, X, \boldsymbol{Y})](\boldsymbol{x})$ in addition to the μ-fixpoint predicates from standard GNFP. We also allow simultaneous μ^N fixpoints. We require that all μ^N-subformulas appear positively in the formula. The idea is that μ^N-operators mark the least fixpoint subformulas where we want to count (and ultimately bound) the number of unfoldings required to witness membership in the fixpoint.

Given ψ in cGNFP and $n \in \mathbb{N}$, we write $\mathfrak{A}, n \models \psi$ if $\mathfrak{A} \models \psi'$ where ψ' is the GNFP formula obtained from ψ

[2]Technically, these counters in $\mathcal{B}$ must be *hierarchical*, obeying certain stack-like properties. The cost automata used in this paper have this property.

by replacing each μ^N subformula by its n-unfolding. The function $[\![\psi]\!]$ defined by ψ is

$$[\![\psi]\!](\mathfrak{A}) := \inf\{n : \mathfrak{A}, n \models \psi\}.$$

For example, in the special case when ψ does not use any μ^N operators, $[\![\psi]\!](\mathfrak{A}) = 0$ if $\mathfrak{A} \models \psi$ and $[\![\psi]\!](\mathfrak{A}) = \infty$ if $\mathfrak{A} \not\models \psi$. Thus the boundedness problem for cGNFP — namely, whether or not the function computed by the formula has a uniform bound — generalizes the validity problem for GNFP formulas. As another example, take φ to be the GNFP formula defined in Example 1, and take $\psi := [\mu^N X, \boldsymbol{x}.\, \mathrm{gdd}(\boldsymbol{x}) \wedge \varphi](y_1 y_2)$. If $\mathfrak{A}_n$ denotes the structure consisting of a "ladder" starting from some distinguished elements a_1 and a_2 and having n-rungs before reaching self-loops, then $[\![\psi]\!](\mathfrak{A}_n, a_1 a_2) = n+1$. For a structure $\mathfrak{A}$ and elements a_1, a_2 such that $\mathfrak{A}, a_1 a_2 \not\models [\mu X, \boldsymbol{x}.\, \mathrm{gdd}(\boldsymbol{x}) \wedge \varphi](y_1 y_2)$, we have $[\![\psi]\!](\mathfrak{A}) = \infty$.

We can add these μ^N-operators to GFP formulas in a similar fashion. We write cGFP for the cost variant of GFP.

We introduce some additional conventions and terminology related to the fixpoint variables in order to describe the nesting of fixpoint operators, and a special subclass of cGNFP. We assume no fixpoint variable is bound by more than one fixpoint operator, so each X identifies a unique subformula $D_\phi(X)$ of ϕ in which X only appears positively. If $D_\phi(X)$ corresponds to a μ^N-predicate, then X is called a *μ^N-variable*. Otherwise, if X occurs positively (respectively, negatively) in ϕ then it is a *μ-variable* (respectively, *ν-variable*). We say that X *depends* on Y if Y occurs free in $D_\phi(X)$. The *dependency order* $\sqsubset_\phi$ is the transitive closure of this relation. The *alternation level* $\mathrm{al}_\phi(X)$ of X is the maximum number of alternations between μ and ν variables on the $\sqsubset_\phi$-paths descending from X, ignoring μ^N-variables. The *counting alternation level* $\mathrm{al}^N_\phi(X)$ of X is the maximum number of alternations between μ^N-variables and other variables on the $\sqsubset_\phi$-paths descending from X. The *alternation depth* $\mathrm{ad}(\phi)$ (respectively, *counting alternation depth* $\mathrm{ad}^N(\phi)$) is the maximum alternation level (respectively, counting alternation level) of any of its fixpoint variables. We say that a formula ϕ is *alternation-free* if $\mathrm{ad}(\phi) = 0$.

We will be interested in a fragment of cGNFP with restricted use of μ^N-variables. *Distance* cGNFP is defined as the fragment of cGNFP consisting of the formulas ϕ such that $\mathrm{ad}^N(\phi) \leq 1$ and every non-μ^N fixpoint variable X in ϕ satisfies $\mathrm{al}^N_\phi(X) = 0$. Roughly speaking, this corresponds to the fragment of cGNFP where we only need to count one fixpoint at a time, and these μ^N-variables do not depend on other μ- and ν-variables. For example, if ψ is in GNF and positive in X then $[\mu^N X, \boldsymbol{x}.\, \mathrm{gdd}(\boldsymbol{x}) \wedge \psi](\boldsymbol{x})$ is in distance cGNFP.

Normal form. For the automata constructions in this section, it is easier to work with cGNFP formulas φ in a normal form. We call this *weak* GN-*normal form*, because it is a slight weakening of the GN-normal form introduced in [3]. We assume that the formulas do not use equality or constants, but our approach can be adapted to handle these additional features, as explained in the full version of this paper.

Formulas φ in cGNFP in weak GN-normal form can be generated using the following grammar:

$$\varphi ::= \textstyle\bigvee_i \exists \boldsymbol{x}. \bigwedge_j \psi_{ij} \;\mid\; [\lambda Y, \boldsymbol{y}.\, \mathrm{gdd}(\boldsymbol{y}) \wedge \varphi(\boldsymbol{y}, \boldsymbol{X}, Y)](\boldsymbol{x}) \quad \text{where } \lambda \in \{\mu, \mu^N\}$$
$$\psi ::= \alpha(\boldsymbol{x}) \mid Y(\boldsymbol{x}) \mid \alpha(\boldsymbol{x}) \wedge \varphi(\boldsymbol{x}, \boldsymbol{X}) \mid \alpha(\boldsymbol{x}) \wedge \neg\varphi(\boldsymbol{x}, \boldsymbol{X})$$

where $\alpha(\boldsymbol{x})$ is an atomic formula or a guardedness predicate $\mathrm{gdd}(\boldsymbol{x})$, Y only appears positively in the fixpoint predicate $[\lambda Y, \boldsymbol{y}.\, \mathrm{gdd}(\boldsymbol{y}) \wedge \varphi(\boldsymbol{y}, \boldsymbol{X}, Y)](\boldsymbol{x})$, and μ^N-predicates only appear positively in φ. As before, the fixpoint predicate variables cannot be used as guards. The idea is that the grammar generates UCQ-shaped formulas where each conjunct is an answer-guarded subquery. Note that we do not allow simultaneous fixpoints in this normal form. We can convert arbitrary cGNFP formulas to this normal form (see Proposition 7).

Closure set. Given some φ in weak GN-normal form, we want to define the relevant formulas that need to be considered by the automaton in order to determine whether φ holds in some structure. It turns out that we need more information than just the usual subformula closure, namely (i) additional guardedness predicates, and (ii) "specializations" of CQ-shaped subformulas. Consider a CQ-shaped $\phi(\boldsymbol{y})$ of the form $\exists \boldsymbol{x}. \bigwedge_i \psi_i(\boldsymbol{xy})$. A *specialization* of ϕ is a formula ϕ' obtained from ϕ by the following operations:

- select a subset $\boldsymbol{z}$ of $\boldsymbol{x}$ (call variables from $\boldsymbol{yz}$ the *inside variables* and variables from $\boldsymbol{x} \setminus \boldsymbol{z}$ the *outside variables*);
- select a partition $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ of the outside variables with the property that for every ψ_j, all free variables of ψ_j are contained in some $\boldsymbol{x}_i$;
- let χ_0 be the conjunction of the ψ_i using only inside variables, and let χ_j be the conjunction of the ψ_i using outside variables and satisfying $\mathrm{free}(\psi_i) \subseteq \boldsymbol{x}_j \boldsymbol{yz}$;
- set $\phi'(\boldsymbol{yz})$ to be $\chi_0(\boldsymbol{yz}) \wedge \bigwedge_{j \in \{1,\ldots,k\}} \exists \boldsymbol{x}_j.\chi_j(\boldsymbol{x}_j \boldsymbol{yz})$.

A specialization describes a way in which the original CQ-shaped subformula could be satisfied in a tree-like model.

Now let $\mathrm{cl}_+(\varphi)$ be the smallest set C of formulas containing φ and satisfying the following closure conditions:

- if $\alpha \wedge \psi \in C$ or $\alpha \wedge \neg\psi \in C$, then $\alpha, \psi \in C$;
- if $\bigvee_i \psi_i \in C$ or $\bigwedge_i \psi_i \in C$, then $\psi_i \in C$ for all i;
- if $\exists \boldsymbol{x}.\psi(\boldsymbol{xy}) \in C$, then for every specialization $\chi_0(\boldsymbol{xy}) \wedge \bigwedge_j \exists \boldsymbol{z}_j.\chi_j(\boldsymbol{xyz}_j)$ of $\exists \boldsymbol{x}.\psi(\boldsymbol{xy})$, $\chi_0 \in C$ and $\exists \boldsymbol{z}_j.\chi_j \in C$;
- if $[\lambda Y, \boldsymbol{y}.\, \mathrm{gdd}(\boldsymbol{y}) \wedge \psi](\boldsymbol{x}) \in C$, then $\mathrm{gdd}(\boldsymbol{y}), \psi \in C$;
- if $\psi(\boldsymbol{x}) \in C$, then $\mathrm{gdd}(\boldsymbol{x}) \in C$.

We are interested in the size of this closure set, since we will see that this parameter controls the size of the automata. The following proposition describes the blow-up in the size of the formula and the size of the closure set when starting from formulas not necessarily in weak GN-normal form.

Proposition 7. *Let ψ be a formula in* cGNFP *(respectively, answer-guarded* cGFP*) with $m = |\psi|$ and $k = \mathrm{width}(\psi)$. We can construct a DAG representation of a sentence φ in weak* GN-*normal form such that $[\![\varphi]\!] \approx [\![\psi]\!]$ and*

- *the size of the DAG representation for φ is at most $2^{f(m)}$ (respectively, $f(m)$),*

- $\mathrm{width}(\varphi) \leq m$ *(respectively,* $\mathrm{width}(\varphi) = k$*),*
- $|\mathrm{cl}_+(\varphi)| \leq 2^{f(m)}$ *(respectively,* $|\mathrm{cl}_+(\varphi)| \leq f(m) \cdot 2^{f(k)}$*),*

where f is a polynomial function independent of ψ. There is no change in alternation-depth and counting alternation-depth. These bounds apply even when ψ uses simultaneous fixpoints.

For brevity, in this proposition and throughout 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.

Note also that the equivalence between the formulas is only up to $\approx$. This means the functions defined by the formulas may not agree on exact values, but do agree on boundedness properties, which is sufficient for our purposes.

Reduction to tree-like structures. In order to answer boundedness questions about cGNFP over all structures, we can actually restrict attention to structures of bounded tree-width. This is a straightforward consequence of the tree-like model property of GNFP. Similar transfer results to tree-like models were utilized in [5] and [10].

Proposition 8. *Let φ be a sentence in* cGNFP$[\sigma]$ *in weak* GN*-normal form such that* $\mathrm{width}(\varphi) = k$*. Then $[\![\varphi]\!]$ is bounded by n over all σ-structures iff $[\![\varphi]\!]$ is bounded by n over all σ-structures of tree-width $k-1$.*

Encodings of tree-like structures. We can encode σ-structures with tree decompositions of some fixed width $k-1$ as trees over a finite alphabet.

Each node in the tree encoding will describe at most k elements of the structure. The names of these elements are taken from $U_k := \{1, \ldots, 2k\}$. Neighboring nodes may describe overlapping pieces of the structure. This will be implicitly coded based on repeated use of names: if some name appears in two neighboring nodes, then the same element is being described in both nodes. This is why U_k has $2k$ names, even though at most k names are used in a single node.

More formally, we view these encodings as structures with unary predicates D_a for all $a \in U_k$ (where $D_a(v)$ indicates that a is a name in the node v in the tree decomposition) and unary predicates $R_{\boldsymbol{a}}$ for all $R \in \sigma$ of arity j and all $\boldsymbol{a} \in U_k^j$ (where $R_{\boldsymbol{a}}(v)$ indicates that R holds for the elements represented by $\boldsymbol{a}$ at v). We write $\mathbb{K}_{\sigma,k}$ for the finite alphabet capturing this information.

A $\mathbb{K}_{\sigma,k}$*-tree* is an infinite, unranked, unordered tree, with arbitrary (possibly infinite) branching and with labels from the finite alphabet $\mathbb{K}_{\sigma,k}$. In particular, this means that the label at a node does not dictate the number of children it has, and each node may have an arbitrary (possibly infinite) number of unordered children. As a result, in this section, we will use B$\wedge$parity automata with directions $\{0, \updownarrow\}$ on these $\mathbb{K}_{\sigma,k}$-trees, since we cannot refer to specific children.

Given a $\mathbb{K}_{\sigma,k}$-tree t and some node $v \in \mathrm{dom}(t)$, we write $\mathrm{names}(v)$ for the set $\{a \in U_k : D_a(v)\}$. We say that t is *consistent* if every node $v \in \mathrm{dom}(t)$ satisfies (i) $|\mathrm{names}(v)| \leq k$, (ii) if $R_{\boldsymbol{a}}(v)$ then $\boldsymbol{a} \subseteq \mathrm{names}(v)$, and (iii) if $R_{\boldsymbol{a}}(v)$ then $R_{\boldsymbol{a}}(w)$ for all neighbors w of v with $\boldsymbol{a} \subseteq \mathrm{names}(w)$. It is straightforward to construct a weak automaton without counters that runs on $\mathbb{K}_{\sigma,k}$-trees t and checks that t is consistent.

Any σ-structure of tree-width $k-1$ can be encoded in a consistent $\mathbb{K}_{\sigma,k}$-tree, and every consistent $\mathbb{K}_{\sigma,k}$-tree corresponds to an actual σ-structure. Given a consistent tree t, we say that two nodes u and v are *a-connected* if a is an element name in every node on the minimal path between u and v. We write $[v, a]$ for the equivalence class of a-connected nodes of v. Using this, we can define the *decoding* of t to be the σ-structure $\mathfrak{D}(t)$ with universe $\{[v, a] : v \in \mathrm{dom}(t) \text{ and } a \in \mathrm{names}(v)\}$, and $R^{\mathfrak{D}(t)}([v_1, a_1], \ldots, [v_j, a_j])$ iff there is some node $w \in \mathrm{dom}(t)$ such that $R_{\boldsymbol{a}}(w)$ holds, and $[w, a_i] = [v_i, a_i]$ for all i.

Finally, we write $\mathrm{cl}_+(\varphi, U_k)$ for the set of formulas obtained by substituting names from U_k for the free first-order variables in formulas from $\mathrm{cl}_+(\varphi)$. This will allow us to talk about formulas ψ that hold in $\mathfrak{D}(t)$, when the free variables in ψ are interpreted by the elements represented at a node in t. Note that the size bounds given in Proposition 7 for $\mathrm{cl}_+(\varphi)$ still hold for $\mathrm{cl}_+(\varphi, U_k)$.

Reduction from cGNFP boundedness to cost automata boundedness. We have introduced a logic cGNFP for specifying boundedness problems, and we have seen that we can reduce to studying these problems on tree-like structures. We can now construct for each cGNFP sentence φ an automaton that runs on an encoding of a tree-like structure and assigns the same value as $[\![\varphi]\!]$ on this structure. We state this result and some of its consequences for boundedness and satisfiability testing, and then describe the main ideas of the construction.

Theorem 9. *Let φ be a sentence in* cGNFP$[\sigma]$ *in weak* GN*-normal form such that* $\mathrm{width}(\varphi) = k$*. Then there is a* B $\wedge$ *parity tree automaton $\mathcal{A}_\varphi$ using directions $\{0, \updownarrow\}$ such that for all consistent $\mathbb{K}_{\sigma,k}$-trees t,*

$$[\![\varphi]\!](\mathfrak{D}(t)) = n \quad \textit{iff} \quad [\![\mathcal{A}_\varphi]\!](t) = n.$$

The size of $\mathcal{A}_\varphi$ is exponential in $|\mathrm{cl}_+(\varphi, U_k)|$, but the number of states is polynomial in $|\mathrm{cl}_+(\varphi, U_k)|$. Moreover, $\mathcal{A}_\varphi$ uses priorities from $\{0, \ldots, 2 \cdot \mathrm{ad}(\varphi) + 1\}$, and counters from $\{\gamma_X : X \text{ is a } \mu^N\text{-variable in } \varphi\}$.

By combining $\mathcal{A}_\varphi$ with a parity automaton that checks whether a given $\mathbb{K}_{\sigma,k}$-tree is consistent it is straightforward to prove the following reduction.

Corollary 10 (Automata for Boundedness)**.** *Let φ be a sentence in* cGNFP$[\sigma]$ *in weak* GN*-normal form such that* $\mathrm{width}(\varphi) = k$*. Then there exists a* B $\wedge$ *parity tree automaton $\mathcal{B}_\varphi$ such that $[\![\varphi]\!]$ is bounded over all σ-structures iff $[\![\mathcal{B}_\varphi]\!]$ is bounded over all $\mathbb{K}_{\sigma,k}$-trees.*

The size of $\mathcal{B}_\varphi$ is exponential in $|\mathrm{cl}_+(\varphi, U_k)|$, but the number of states is polynomial in $|\mathrm{cl}_+(\varphi, U_k)|$. The number of priorities and counters is linear in $|\varphi|$.

Moreover, $\mathcal{B}_\varphi$ is a dist$\wedge$parity automaton if φ is in distance cGNFP*, and is counter-weak if φ is alternation-free.*

Application to satisfiability. In the special case when φ is in GNFP or GFP, Theorem 9 provides parity tree automata (without counters) that can be used to test satisfiability.

Corollary 11 (Automata for Satisfiability)**.** *Let φ be a sentence in* GNFP *in weak* GN*-normal form such that $\mathrm{width}(\varphi) = k$. Then there exists a 2-way alternating parity tree automaton $\mathcal{B}_\varphi$ such that φ is satisfiable iff $L(\mathcal{B}_\varphi) \neq \emptyset$, where the language $L(\mathcal{B}_\varphi) := \{t : [\![\mathcal{B}_\varphi]\!](t) = 0\}$ is the set of accepted $\mathbb{K}_{\sigma,k}$-trees.*

The size of $\mathcal{B}_\varphi$ is exponential in $|\mathrm{cl}_+(\varphi, U_k)|$, but the number of states is polynomial in $|\mathrm{cl}_+(\varphi, U_k)|$. The number of priorities and counters is linear in $|\varphi|$.

There is a procedure due to Vardi [15] that tests non-emptiness for 2-way alternating parity tree automata in time exponential in the number of states and number of priorities. Hence, Corollary 11 provides an alternative and direct proof of the 2EXPTIME bound for satisfiability of GNFP [3]. Moreover, when applied to fixed-width GFP sentences, the same construction demonstrates the optimal EXPTIME bound from [4].

Thus, even ignoring the additional cost features that will allow us to decide boundedness problems for these guarded logics, we believe the constructions in this section are useful because they provide a uniform approach to answering satisfiability questions for these guarded logics.

Construction of cost automaton for Theorem 9. We now seek to describe some of the ideas behind the construction of the cost automaton $\mathcal{A}_\varphi$ in Theorem 9.

Our construction for cGNFP can be seen as a generalization of the construction for GFP presented by Grädel and Walukiewicz [4], so we briefly describe their construction, and how we extend it. Given some GFP sentence φ, [4] defines an automaton $\mathcal{A}_\varphi$ that recognizes consistent trees that encode structures satisfying φ, and uses this to show optimal bounds on the complexity of satisfiability for GFP. The state set for their constructed automaton is just $\mathrm{cl}(\varphi, U_k)$ (the subformula closure of φ, with names from U_k substituted for free variables), and if the automaton is in state ψ at some position v in the input tree t, then it represents an assertion by Eve that ψ holds in $\mathfrak{D}(t)$.

In their construction, it is very clear when a fixpoint is being unfolded: it is precisely a transition from some state $X(\boldsymbol{b})$ for a fixpoint variable X to a state corresponding to the body $D_\varphi(X)$ of the fixpoint. The automaton uses the parity acceptance condition to help enforce the correct fixpoint semantics. If we were only interested in boundedness questions related to GFP and the cost version of GFP, then we could take directly the construction from [4] and hook in a counter for each μ^N-variable X that is incremented each time that fixpoint is unfolded and reset when any outer fixpoint is unfolded.

However, we are interested in an automaton $\mathcal{A}_\varphi$ for cGNFP φ. When run on a consistent $\mathbb{K}_{\sigma,k}$-tree t, $\mathcal{A}_\varphi$ computes the least n such that $\mathfrak{D}(t)$ is a model of φ when all μ^N-subformulas are evaluated using their n-th approximants. We want to emphasize that we are not using the counters in a complicated way: the counters are just counting the number of times a μ^N-operator is unfolded. Hence, in the special case, when there are no μ^N operators in φ, $\mathcal{A}_\varphi$ is a parity automaton with no counters that accepts a consistent $\mathbb{K}_{\sigma,k}$-tree t iff t encodes a structure satisfying the GNFP sentence φ.

We construct the automaton for φ by induction on the the structure of φ. The idea is to allow Eve to guess an annotation of t with information about which answer-guarded subformulas of φ hold, and then run an automaton that checks φ with the help of these annotations. In order to prevent Eve from cheating with her guesses about the subformulas, Adam is allowed to launch inductively-defined automata in order to check Eve's claims about these subformulas.

Likewise, testing $\mathfrak{D}(t), n \models [\lambda Y, \boldsymbol{y}.\, \mathrm{gdd}(\boldsymbol{y}) \wedge \psi(\boldsymbol{y}, Y)](\boldsymbol{a})$ for $\lambda \in \{\mu, \mu^N\}$ can be viewed as a game between Adam and Eve which starts with $\boldsymbol{y} = \boldsymbol{a}$ and proceeds as follows:

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

If $\lambda = \mu^N$ (respectively, $\lambda = \mu$) and the game exceeds n turns (respectively, never terminates), then Adam is declared the winner. We can implement this game as an automaton running on t, where Eve guesses an annotation of t with the valuation of Y in the current round, and then simulates the inductively-defined automaton checking $\psi(\boldsymbol{y}, Y)$. Adam can challenge any $\boldsymbol{b}$ in the set Y chosen by Eve by launching another copy of the automaton checking ψ starting from the node carrying $\boldsymbol{b}$. Correctness is enforced by using the parity condition and counters: an odd priority is used if Adam challenges a μ fixpoint, and a counter is incremented if Adam challenges a μ^N fixpoint. The proof uses ideas familiar from work on the μ-calculus and GFP (see, e.g., [24], [4], [25]).

V. DECIDING BOUNDEDNESS FOR DIST $\wedge$ PARITY AUTOMATA

Conversions required for deciding boundedness. In the previous section, we showed how to reduce cGNFP boundedness to cost automata boundedness. Unfortunately, it is not known in general whether boundedness for arbitrary B$\wedge$parity automata on infinite trees is decidable, even when we restrict to 1-way automata on infinite binary trees.

However, if we can convert a 2-way B $\wedge$ parity automaton to an equivalent 1-way nondeterministic S$\wedge$parity automaton, then we can apply the decidability result in Theorem 3. Why a nondeterministic S $\wedge$ parity automaton? In order to show that $[\![\mathcal{A}]\!]$ is unbounded for some cost automaton $\mathcal{A}$, we need to check that there is a family of trees $(t_n)_{n\in\mathbb{N}}$ such that $[\![\mathcal{A}]\!](t_n) > n$. We can cast this as a game where Eve is trying to guess a family of strategies $(\tau_n)_{n\in\mathbb{N}}$ on input trees $(t_n)_{n\in\mathbb{N}}$ that witness this unboundedness. Hence, we want $\mathcal{A}$ to be a cost automaton where Eve is the maximizing player and a *single* strategy can witness that $[\![\mathcal{A}]\!](t_n) > n$. Automata with an S $\wedge$ parity objective meet this requirement. Moreover, the automaton must be nondeterministic to ensure that Eve's guessed strategy τ_n actually induces a corresponding tree t_n.

In this section, we show that this conversion to a nondeterministic S $\wedge$ parity automata is possible if we start with dist$\wedge$parity automata (which have one counter that can only be incremented or left unchanged). Given a 2-way dist$\wedge$parity tree automaton $\mathcal{A}$ using directions $\{0, \updownarrow\}$, we proceed as follows.

1) Convert $\mathcal{A}$ to an equivalent 2-way $\overline{\text{dist} \wedge \text{parity}}$ tree automaton, i.e. a cost automaton using the dual of the dist$\wedge$parity objective (denoted $\overline{\text{dist} \wedge \text{parity}}$, as defined in Section III) where Eve is the maximizing player. Because we are working with alternating automata, this is easy: we exchange the states owned by each player, and $\min$ for $\max$ in the objective.
2) Reduce to trees with some finite branching m, where m is the number of states of $\mathcal{A}$. This is a straightforward consequence of the positional determinacy of $\overline{\text{dist} \wedge \text{parity}}$ games described in Proposition 13. We can then convert to a 2-way $\overline{\text{dist} \wedge \text{parity}}$ tree automaton with directions $\{-1, 0, 1 \ldots, m\}$ rather than directions $\{0, \updownarrow\}$.
3) Convert from a 2-way $\overline{\text{dist} \wedge \text{parity}}$ tree automaton with directions $\{-1, 0, 1, \ldots, m\}$ to a 1-way nondeterministic S $\wedge$ parity tree automaton (Theorem 14). This generalizes Vardi's construction in [15], and is the most interesting part of the conversion process.
4) Test boundedness of the resulting nondeterministic S $\wedge$ parity automaton using Theorem 3.

Using this conversion process yields the following results.

Theorem 12. *Let $\mathcal{A}$ be a dist $\wedge$ parity tree automaton using directions $\{0, \updownarrow\}$.*

- *If $\mathcal{A}$ uses priorities $\{0, 1\}$, then boundedness for $[\![\mathcal{A}]\!]$ is decidable in elementary time. Using* (EXP-Dual-Finite)*, this can be improved to time $|\mathcal{A}|^{f(m)}$ where m is the number of states of $\mathcal{A}$, and f is a polynomial independent of $\mathcal{A}$.*
- *If $\mathcal{A}$ uses arbitrary priorities, then using* (Dual-Infinite)*, boundedness for $[\![\mathcal{A}]\!]$ is decidable in elementary time.*

We remark that one reason dist $\wedge$ parity automata are easier to work with — and one explanation for why we are able to prove boundedness is decidable in this special case — is that the underlying games have positional strategies.

Proposition 13 (due to Colcombet and Löding)**.** *If Eve has an n-winning strategy in dist $\wedge$ parity or $\overline{\text{dist} \wedge \text{parity}}$ game, then she has an n-winning positional strategy.*

The proof is straightforward: the idea is to break the problem down into solving various parity games (without counters). These parity games have positional strategies [26], and we use these positional strategies in subgames to construct a positional strategy in the original cost game. The corresponding result for general B $\wedge$ parity and $\overline{\text{B} \wedge \text{parity}}$ games is not known.

Conversion to nondeterministic automaton. We now describe in more detail item (3), the conversion from 2-way $\overline{\text{dist} \wedge \text{parity}}$ automata to 1-way nondeterministic S $\wedge$ parity automata, operating on trees with finite branching.

Theorem 14. *Let $\mathcal{A}$ be a 2-way $\overline{\text{dist} \wedge \text{parity}}$ tree automaton using directions $\{-1, 0, 1, \ldots, m\}$.*

- *If $\mathcal{A}$ uses only priorities $\{0, 1\}$, then there is a 1-way nondeterministic S $\wedge$ parity tree automaton $\mathcal{S}_{\text{nd}}$ of elementary size such that $[\![\mathcal{S}_{\text{nd}}]\!] \approx [\![\mathcal{A}]\!]$. Moreover, using* (EXP-Dual-Finite)*, $\mathcal{S}_{\text{nd}}$ is of size at most $|\mathcal{A}|^{f(m)}$ with at most $|\mathcal{A}|^{f(m)}$ states, $f(m)$ counters, and priorities $\{1, 2\}$ where m is the number of states of $\mathcal{A}$ and f is a polynomial function independent of $\mathcal{A}$.*
- *If $\mathcal{A}$ uses arbitrary priorities, then using* (Dual-Infinite) *there is a 1-way nondeterministic S $\wedge$ parity tree automaton $\mathcal{S}_{\text{nd}}$ of elementary size such that $[\![\mathcal{S}_{\text{nd}}]\!] \approx [\![\mathcal{A}]\!]$.*

The proof proceeds in stages. We sketch some key ideas.

Fix some 2-way $\overline{\text{dist} \wedge \text{parity}}$ automaton $\mathcal{A}$ using directions $\{-1, 0, \ldots, m\}$. Consider a tree t that has been annotated with a positional strategy ζ in $\mathcal{G}(\mathcal{A}, t)$. Formally, a *strategy annotation* is a labelling of t such that each node x is annotated with a function ζ_x mapping a state q to the set of tuples $(d, (p, c), r)$ such that it is possible to move from (q, x) to (r, xd) with output (p, c) when using the strategy ζ in the game in $\mathcal{G}(\mathcal{A}, t)$. We write (t, ζ) for this annotated tree. It is easy to construct a 2-way $\overline{\text{dist} \wedge \text{parity}}$ automaton $\mathcal{U}$ where all moves are controlled by Adam such that when run on (t, ζ) it computes the value of ζ in $\mathcal{G}(\mathcal{A}, t)$. Note that $\mathcal{U}$ is of size at most $|\mathcal{A}|^{f(m')}$ where m' is the number of states and priorities of $\mathcal{A}$, but it uses the same number of state and priorities as $\mathcal{A}$.

This automaton $\mathcal{U}$ operates in a 2-way fashion, but we can convert it into a 1-way automaton. The price we pay to eliminate the upward movement is (i) we give some control back to Eve so the automaton is no longer universal, and (ii) we increase the size by a polynomial factor.

Lemma 15. *We can construct a 1-way alternating $\overline{\text{dist} \wedge \text{parity}}$ tree automaton $\mathcal{B}$ such that $[\![\mathcal{U}]\!] \approx [\![\mathcal{B}]\!]$. The size of $\mathcal{B}$ is at most $|\mathcal{A}|^{f(m')}$ where m' is the number of states and priorities of $\mathcal{A}$, and f is some polynomial function independent of $\mathcal{A}$. Moreover, the number of states in $\mathcal{B}$ is polynomial in m', and the number of priorities and counters is the same as $\mathcal{A}$.*

Proof sketch: The idea is that the automaton runs on (t, ζ) and simulates $\mathcal{U}$, but Adam is never allowed to move upwards. Instead, at any time, Adam can issue a request to stay in the same position, change state, and output some action, which represents an assertion that there is some play consistent with ζ that would have looped back to the current position.

When Adam makes a request like this, Eve can either grant him the request, or challenge the request. If she grants the request, then she just uses the move described by Adam. If she challenges the request, then the automaton enters a different mode where Adam must actually witness the requested loop with some finite play. Note that in order to witness it, he may need to make additional requests, which Eve can also decide to challenge. The parity condition is used to enforce that any challenged request is eventually witnessed.

A final transformation is used to eliminate the stationary moves, to ensure the resulting automaton is truly 1-way. ■

This 1-way $\overline{\text{dist} \wedge \text{parity}}$ automaton $\mathcal{B}$ can be *simulated* by a nondeterministic S $\wedge$ parity automaton.

Lemma 16. *Let $\mathcal{B}$ be a 1-way alternating $\overline{\text{dist} \wedge \text{parity}}$ tree automaton.*

- *If $\mathcal{B}$ uses priorities $\{0,1\}$, then there is a 1-way nondeterministic* S $\wedge$ *parity tree automaton $\mathcal{S}$ of elementary size such that $[\![\mathcal{S}]\!] \approx [\![\mathcal{B}]\!]$. Moreover, using* (Exp-Dual-Finite)*, $\mathcal{S}$ is of size $|\mathcal{B}|^{f(m)}$, with at most $|\mathcal{B}|^{f(m)}$ states, m counters, and priorities $\{1,2\}$ where m is the number of states of $\mathcal{B}$ and f is a polynomial independent of $\mathcal{B}$.*
- *If $\mathcal{B}$ uses arbitrary priorities, then using* (Dual-Infinite) *there is a 1-way nondeterministic* S $\wedge$ *parity tree automaton $\mathcal{S}$ of elementary size such that $[\![\mathcal{S}]\!] \approx [\![\mathcal{B}]\!]$.*

Proof sketch: We first consider the case for arbitrary priorities. When running on t, the nondeterministic S $\wedge$ parity automaton guesses a positional strategy ζ in $\mathcal{G}(\mathcal{B}, t)$. Let w be a word describing the strategy annotation labels along a given branch in (t, ζ). There is a nondeterministic dist $\wedge$ parity automaton on infinite words that guesses a play described by w and outputs the value of this play. The number of states is polynomial in $|Q_{\mathcal{A}}|$ (since the automaton only needs to remember the currently guessed state on this play). By (Dual-Infinite), there is an equivalent history-deterministic S $\wedge$ parity automaton $\mathcal{H}$ of elementary size. There is a deterministic tree automaton $\mathcal{D}$ (with no counters and using only priorities $\{0,1\}$) of size exponential in the number of states and priorities of $\mathcal{A}$ that checks that a given annotation of a tree t actually corresponds to a positional strategy in $\mathcal{G}(\mathcal{B}, t)$. The desired automaton $\mathcal{S}$ is obtained by guessing an annotation ζ, simulating $\mathcal{D}$ on (t, ζ), and simultaneously running $\mathcal{H}$ on each branch. Since positional strategies suffice in $\overline{\text{dist} \wedge \text{parity}}$ games (Proposition 13), this automaton is $\approx$-equivalent to $\mathcal{B}$.

In the special case when $\mathcal{B}$ uses only priorities $\{0,1\}$, then we can optimize the construction to get the improved complexity bounds stated above. In addition to the positional strategy, the automaton guesses infinitely many "breakpoints" (based on occurrences of priority 1 and increments). The value of the strategy can then be defined as the limit of the values of the strategy up to these breakpoints. This means the value of plays along a single branch can be computed by a cost automaton on *finite words*. This allows us to make use of (Dual-Finite) or (Exp-Dual-Finite) instead of (Dual-Infinite), which yields the improved complexity bounds. ∎

Finally, the desired nondeterministic 1-way automaton $\mathcal{S}_{\text{nd}}$ for Theorem 14 is obtained by guessing a strategy annotation ζ in $\mathcal{G}(\mathcal{A}, t)$, running a deterministic tree automaton (without counters) that checks that this is a valid annotation, and simulating $\mathcal{S}$ on (t, ζ). The correctness of this construction relies on Proposition 13, Lemma 15, and Lemma 16.

This concludes the proof sketch of Theorem 14.

VI. Complexity of boundedness problems for guarded logics

Upper bounds. The following corollary, obtained by combining Theorem 9 and Theorem 3, summarizes our work so far.

Corollary 17. *Let φ be a* cGNFP *sentence. Boundedness for $[\![\varphi]\!]$ is in elementary time for*

- *alternation-free distance* cGNFP*;*
- *arbitrary distance* cGNFP*, using* (Dual-Infinite)*.*

Using (Exp-Dual-Finite)*, this can be improved to* 2ExpTime *for alternation-free distance* cGNFP *and* ExpTime *for alternation-free fixed-width distance* cGFP*.*

We can use this to derive upper bounds on the complexity of boundedness for some guarded logics and query languages.

Theorem 18. *Boundedness is decidable in elementary time for the answer-guarded fragments of the following logics:*

- GNF *and* GF*;*
- GNFP *and* GFP*, using* (Dual-Infinite)*.*

Using (Exp-Dual-Finite)*, this can be improved to* 2ExpTime *for answer-guarded* GNF *and* GF*, and* ExpTime *for fixed-width answer-guarded* GF*.*

Proof: Consider a formula $\varphi(\boldsymbol{x}, X)$ in answer-guarded GNFP[σ] or answer-guarded GFP[σ]. The goal is to write a sentence in cGNFP that expresses what it means for φ to be bounded. Informally, this means that it should express

$$\forall \boldsymbol{x}. \left([\mu X, \boldsymbol{x}. \varphi(\boldsymbol{x}, X)](\boldsymbol{x}) \rightarrow [\mu^N X, \boldsymbol{x}. \varphi(\boldsymbol{x}, X)](\boldsymbol{x}) \right).$$

We must be more careful to ensure this actually meets the syntactic conditions for cGNFP, so we actually consider the following distance cGNFP sentence φ':

$$\neg \exists \boldsymbol{x}. \begin{pmatrix} \text{gdd}(\boldsymbol{x}) \wedge [\mu X, \boldsymbol{x}. \text{gdd}(\boldsymbol{x}) \wedge \varphi(\boldsymbol{x}, X)](\boldsymbol{x}) \\ \wedge \neg [\mu^N X, \boldsymbol{x}. \text{gdd}(\boldsymbol{x}) \wedge \varphi(\boldsymbol{x}, X)](\boldsymbol{x}) \end{pmatrix}.$$

It is straightforward to check that $[\![\varphi']\!]$ is bounded iff φ is bounded.

The complexity bounds follow from Corollary 17, observing that φ' is in alternation-free cGNFP when φ is in answer-guarded GNF or answer-guarded GF. ∎

We can also give better bounds on the complexity for GN-Datalog boundedness.

Theorem 19. *Full boundedness for* GN*-Datalog programs is decidable in elementary time. Using* (Exp-Dual-Finite)*, this can be improved to* 2ExpTime*.*

Proof: Fix some GN-Datalog program $\Pi = \langle \Pi_1, \ldots, \Pi_j \rangle$. For each level i, we can view Π_i as a system S_i of GNFP formulas over the signature $\text{EDB}^{\Pi_i} \cup \text{IDB}^{\Pi_i}$. For each IDB^{Π_i} predicate X, there is a GNFP formula $\varphi_i^X(\boldsymbol{x}) = [\mu X, \boldsymbol{x}. S_i](\boldsymbol{x})$ asserting that $\boldsymbol{x}$ is in the X-component of the least fixpoint. This formula is answer-guarded, because all tuples generated by a GN-Datalog program must be guarded.

With this in mind, we can construct a DAG-representation of a system of answer-guarded GNFP formulas for the program up to level i (starting at the lowest stratum, and working upwards). This transformation can be done in polynomial time, with the DAG-representation polynomial in the size of $\langle \Pi_1, \ldots, \Pi_i \rangle$. Each formula in the system is an alternation-free GNFP formula in weak GN-normal form with simultaneous fixpoints, but these simultaneous fixpoints can be eliminated with only an additional polynomial blow-up.

The GN-Datalog program is fully bounded if for each level i and for each $X \in \mathrm{IDB}^{\Pi_i}$, the alternation-free GNFP formula φ_i^X is bounded. Using a similar argument as Theorem 18, we can decide boundedness for each φ_i^X. The complexity bounds follow from Corollary 17 and the fact that the formulas we are testing are alternation-free. ∎

Extensions. Boundedness for non answer-guarded GF and GFP formulas can be reduced to the answer-guarded case: indeed, given ϕ in GF or GFP, $\phi(\boldsymbol{x})$ is bounded iff $\mathrm{gdd}(\boldsymbol{x}) \wedge \phi$ is bounded [5]. Hence the boundedness results for answer-guarded GF and GFP extend to the non answer-guarded case.

However, for non answer-guarded GNF or GNFP, boundedness is undecidable. This can be seen by observing that the body of a single Datalog rule can be trivially written as a non answer-guarded GNF formula ϕ, and boundedness of this single-rule Datalog program (undecidable by [9]) is equivalent to boundedness of ϕ.

Lower bounds. For these guarded logics, lower bounds for boundedness follow from lower bounds on the complexity of the satisfiability problem and from the containment problem. The latter bound relies on a recently-proven lower bound for containment of Monadic Datalog in UCQs [27].

Theorem 20. *Boundedness is* 2ExpTime*-hard for answer-guarded* GF *and also for Monadic Datalog (hence for* GN*-Datalog). It is* ExpTime*-hard for fixed-width answer-guarded* GF *without equality.*

VII. Conclusions

In this paper we have re-visited the translation from guarded logics to automata, extending it to the setting of costs. This allows us to isolate the complexity of the boundedness problem. We also believe our translations give some insight into automata for guarded logics, which may serve as a basis for other results — e.g. complexity bounds for richer guarded logics, or better bounds for fragments.

Boundedness coincides with first-order definability for answer-guarded GNF and GN-Datalog by the Barwise-Moschovakis theorem [28]. This is not the case for boundedness of answer-guarded GNFP, due to the presence of additional fixpoints. Thus our work leaves open the question of whether one can decide if a GNFP formula is equivalent to a first-order formula.

Note that combining the results here with those of [14] gives the first elementary bounds on deciding the FO-rewritability of conjunctive queries over guarded and frontier-guarded tuple-generating dependencies (see Section 5 of [14] for the relevant definitions). We believe that the results here should also be applicable to many decision problems involving reasoning with incomplete information (e.g. chase termination).

Acknowledgment

Benedikt and Vanden Boom are supported by EPSRC grants EP/H017690/1 and EP/L012138/1. ten Cate is supported by NSF grant IIS-1217869. Colcombet received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n^{o}259454.

References

[1] J. Bradfield and C. Stirling, "Modal mu-calculi," in *Handbook of Modal Logic*. Elsevier, 2007, pp. 721–756.
[2] H. Andréka, I. Németi, and J. van Benthem, "Modal languages and bounded fragments of predicate logic," *Journal of Philosophical Logic*, vol. 27, no. 3, pp. 217–274, 1998.
[3] V. Bárány, B. ten Cate, and L. Segoufin, "Guarded negation," in *ICALP*, 2011.
[4] E. Grädel and I. Walukiewicz, "Guarded fixed point logic," in *LICS*, 1999.
[5] A. Blumensath, M. Otto, and M. Weyer, "Decidability results for the boundedness problem," *Logical Methods in Computer Science*, vol. 10, no. 3, 2014.
[6] M. Otto, "Eliminating recursion in the μ-calculus," in *STACS*, 1999.
[7] G. G. Hillebrand, P. C. Kanellakis, H. G. Mairson, and M. Y. Vardi, "Undecidable boundedness problems for datalog programs," *J. Log. Program.*, vol. 25, no. 2, pp. 163–190, 1995.
[8] S. S. Cosmadakis, H. Gaifman, P. C. Kanellakis, and M. Y. Vardi, "Decidable optimization problems for database logic programs," in *STOC*, 1988.
[9] S. Abiteboul, "Boundedness is undecidable for datalog programs with a single recursive rule," *IPL*, vol. 32, no. 6, pp. 281–287, 1989.
[10] V. Bárány, B. ten Cate, and M. Otto, "Queries with guarded negation," *PVLDB*, vol. 5, no. 11, pp. 1328–1339, 2012.
[11] T. Colcombet, "A Safra-like construction for regular cost functions over finite words," available at http://www.liafa.univ-paris-diderot.fr/~colcombe/.
[12] B. ten Cate and L. Segoufin, "Unary negation," in *STACS*, 2011.
[13] A. Arnold and D. Niwiński, *Rudiments of μ-Calculus*. Elsevier, 2001.
[14] V. Bárány, M. Benedikt, and B. ten Cate, "Rewriting guarded negation queries," in *MFCS*, 2013.
[15] M. Y. Vardi, "Reasoning about the past with two-way automata," in *ICALP*, 1998.
[16] D. E. Muller, A. Saoudi, and P. E. Schnupp, "Alternating automata. the weak monadic theory of the tree, and its complexity," in *ICALP*, 1986.
[17] T. Colcombet, "The theory of stabilisation monoids and regular cost functions," in *ICALP*, 2009.
[18] ——, "Regular Cost Functions, Part I: Logic and Algebra over Words," *Logical Methods in Computer Science*, vol. 9, no. 3, 2013.
[19] T. Colcombet and C. Löding, "Regular cost functions over finite trees," in *LICS*, 2010.
[20] M. Vanden Boom, "Weak cost monadic logic over infinite trees," in *MFCS*, 2011.
[21] ——, "Weak cost automata over infinite trees," Ph.D. dissertation, University of Oxford, 2012.
[22] S. Safra, "On the complexity of omega-automata," in *FOCS*, 1988.
[23] S. Schewe, "Tighter bounds for the determinisation of Büchi automata," in *FOSSACS*, 2009.
[24] R. S. Streett and E. A. Emerson, "An automata theoretic decision procedure for the propositional mu-calculus," *Inf. Comput.*, vol. 81, no. 3, pp. 249–264, 1989.
[25] D. Berwanger and E. Grädel, "Games and model checking for guarded logics," in *LPAR*, 2001.
[26] E. A. Emerson and C. S. Jutla, "The complexity of tree automata and logics of programs (extended abstract)," in *FOCS*, 1988.
[27] M. Benedikt, P. Bourhis, and P. Senellart, "Monadic datalog containment," in *ICALP*, 2012.
[28] J. Barwise and Y. Moschovakis, "Global inductive definability," *The Journal of Symbolic Logic*, vol. 43, no. 3, pp. 521–534, 1978.
[29] W. Thomas, "Languages, Automata, and Logic," in *Handbook of Formal Languages*, G. Rozenberg and A. Salomaa, Eds., 1997.
[30] E. Grädel, "On the restraining power of guards," *JSL*, vol. 64, no. 4, pp. 1719–1742, 1999.

APPENDIX A
MISSING PROOFS FROM SECTION III

A. Proof sketch for special case of Theorem 3 (Complexity of deciding cost automata boundedness)

We sketch the proof of the complexity bound for deciding boundedness of $[\![\mathcal{A}]\!]$, when $\mathcal{A}$ is a nondeterministic S $\wedge$ parity automaton with m states, n counters, and l priorities. Roughly speaking, the complexity is exponential in n and l and only polynomial in m; we state more precise bounds at the end of this section.

We refer the interested reader to [21, Theorem 4.2] for the full proof of a more general decidability result for the problem of testing $[\![\mathcal{A}]\!] \preccurlyeq [\![\mathcal{B}]\!]$. But complexity bounds were not stated there.

We proceed in stages, eventually reducing the boundedness problem to solving a certain Rabin game. We assume familiarity with traditional winning conditions like this (see, e.g., [29]).

Stage 1: S $\wedge$ parity game $\mathcal{G}$. Let $\mathcal{G}$ be the S $\wedge$ parity game where Eve guesses a tree and simulates $\mathcal{A}$ on it. Positions in the game are of the form $q \in Q$. The initial position is q_0. At position q, the game proceeds as follows:

- Eve selects a letter $a \in \Sigma$, and a disjunct φ of $\delta(q, a)$.
- Adam selects a conjunct $(d, (c, p), r)$ in φ.
- The counter action and priority (c, p) is output, and play proceeds from state r.

The objective is taken directly from $\mathcal{A}$.

The idea is that Eve is trying to choose a tree and a run of $\mathcal{A}$ on this tree with a large value.

Note that the game graph has m nodes, n counters, and l priorities.

Stage 2: $\overline{\text{B} \wedge \text{parity}}$ game $\widetilde{\mathcal{G}}$. We now transform the S$\wedge$parity game into a $\overline{\text{B} \wedge \text{parity}}$ game.

There is a history deterministic B $\wedge$ parity automaton $\mathcal{H}$ on infinite words that reads S $\wedge$ parity actions and converts to an equivalent (up to $\approx$) sequence of B$\wedge$parity actions [21, Lemma 3.3]. The number of states of this automaton is exponential in n, and it uses n counters and l priorities.[3]

Positions are of the form $(q, q') \in Q_{\mathcal{A}} \times Q_{\mathcal{H}}$.

- Eve selects a letter $a \in \Sigma$, and a disjunct φ of $\delta(q, a)$.
- Adam selects a conjunct $(d, (c, p), r)$ in φ and a transition $(r', (c', p')) \in \delta_{\mathcal{H}}(q', (c, p))$.
- The counter action and priority (c', p') is output, and play proceeds from state (r, r').

The objective is now $\overline{\text{B} \wedge \text{parity}}$, so Eve is trying to maximize the value.

Note that the game graph is of size at most $|\mathcal{A}|^{f(nl)}$ for some polynomial function f independent of $\mathcal{A}$, and it has n counters and l priorities.

Stage 3: Rabin game $\widetilde{\mathcal{G}}'$. Finally, we construct a Rabin game $\widetilde{\mathcal{G}}'$ based on $\widetilde{\mathcal{G}}$. Recall that a Rabin condition is given by a set of pairs $\{(E_1, F_1), \ldots, (E_s, F_s)\}$, and a play is winning with respect to this condition if there is some i such that E_i is visited only finitely often, but F_i is visited infinitely often.

We define the game graph for $\widetilde{\mathcal{G}}'$ to be the same as in $\widetilde{\mathcal{G}}$. However, we now view it as an ω-regular game where the winning condition expresses

- the parity condition is not satisfied or
- at least one counter is incremented infinitely often and reset finitely often.

We emphasize that the counter actions are now evaluated according to this ω-regular condition, so the resulting game is no longer a cost game.

It is straightforward to convert a parity condition (or the negation of a parity condition) with l priorities into a corresponding Rabin condition with l pairs. The second part of the winning condition above fits exactly the structure of a Rabin condition with n pairs (since we are expressing that there is some i, such that counter i is reset finitely often but incremented infinitely often). The disjunction of Rabin conditions is again a Rabin condition, so overall this is a Rabin condition where the number of pairs is at most $n + l$.

Traditionally, Rabin games like this do not have edge labels, so we can convert to a more traditional game by shifting the output to nodes (this results in increasing the size of the game graph by the size of the edge alphabet, in this case $2(n + l)$).

Hence, the overall size is at most $|\mathcal{A}|^{f'(nl)}$ for some polynomial function f' independent of $\mathcal{A}$, and the number of pairs is at most $n + l$.

We can then use a pumping lemma to prove that a winning strategy in $\widetilde{\mathcal{G}}'$ can be used to generate a family of trees and runs of $\mathcal{A}$ that witness the unboundedness of $[\![\mathcal{A}]\!]$.

Lemma 21. *$[\![\mathcal{A}]\!]$ is unbounded iff Eve has a winning strategy in the Rabin game $\widetilde{\mathcal{G}}'$.*

This means that we can decide boundedness by solving the Rabin game $\widetilde{\mathcal{G}}'$.

Rabin games can be solved in time $O((MN)^{3N})$ [26] where M is the size of the game graph and N is the number of pairs. Hence, solving $\widetilde{\mathcal{G}}'$ can be done in time $O((MN)^{3N})$ where $M = |\mathcal{A}|^{f'(nl)}$ and $N = n + l$.

Because the transformations from $\mathcal{A}$ to $\widetilde{\mathcal{G}}'$ in each stage were also bounded by this time, the overall complexity of deciding boundedness for $[\![\mathcal{A}]\!]$ is also $|\mathcal{A}|^{f''(nl)}$ for some polynomial function f'' independent of $\mathcal{A}$.

[3] No complexity bound is stated in [21, Lemma 3.5], but it is the result of taking the product of n copies of the 3-state automaton in [21, Figure 3.3].

APPENDIX B
MISSING PROOFS FROM SECTION IV

A. Proof of Proposition 7 (Normal form sizes)

We seek to prove Proposition 7:

> Let ψ be a formula in cGNFP (respectively, answer-guarded cGFP) with $m = |\psi|$ and $k = \mathrm{width}(\psi)$. We can construct a DAG representation of a sentence φ in weak GN-normal form such that $[\![\varphi]\!] \approx [\![\psi]\!]$ and
> - the size of the DAG representation for φ is at most $2^{f(m)}$ (respectively, $f(m)$),
> - $\mathrm{width}(\varphi) \leq m$ (respectively, $\mathrm{width}(\varphi) = k$),
> - $|\mathrm{cl}_+(\varphi)| \leq 2^{f(m)}$ (respectively, $|\mathrm{cl}_+(\varphi)| \leq f(m) \cdot 2^{f(k)}$),
>
> where f is a polynomial function independent of ψ. There is no change in alternation-depth and counting alternation-depth. These bounds apply even when ψ uses simultaneous fixpoints.

We present the argument first for formulas ψ without simultaneous fixpoints, and then argue how to extend the result to formulas with simultaneous fixpoints.

We start by proving some lemmas about the conversion of formulas into weak GN-normal form. We track the change in size, width, and *CQ-rank* during this conversion. For ϕ' in weak GN-normal form, we define $\mathrm{rank}_{\mathrm{CQ}}(\phi')$ to be the maximum number of conjuncts ψ_i in any CQ-shaped subformula $\exists \boldsymbol{x}. \bigwedge_i \psi_i$ of ϕ' for non-empty $\boldsymbol{x}$. For the purposes of CQ-rank, $\alpha(\boldsymbol{x}) \wedge \phi(\boldsymbol{x})$ and $\alpha(\boldsymbol{x}) \wedge \neg\phi(\boldsymbol{x})$ are treated as single conjuncts in a CQ-shaped subformula. The size, width, and CQ-rank will be parameters when calculating the size of the closure set.

Lemma 22. *Let ϕ be a* cGNFP *formula without simultaneous fixpoints. We can construct an equivalent ϕ' in weak* GN*-normal form (in fact, a stronger* GN*-normal form) such that $[\![\phi]\!] = [\![\phi']\!]$ and*

- *$|\phi'|$ is exponential in $|\phi|$;*
- *$\mathrm{width}(\phi') \leq |\phi|$;*
- *$\mathrm{ad}(\phi') = \mathrm{ad}(\phi)$;*
- *$\mathrm{rank}_{\mathrm{CQ}}(\phi') \leq |\phi|$.*

Proof: Use the transformations described in [3] to convert to GN-normal form (although it is stated for GNF and GNFP, the same transformation works for cGNFP and satisfies $[\![\phi]\!] = [\![\phi']\!]$). ∎

Likewise, formulas in answer-guarded cGFP can be converted to weak GN-normal form with only a polynomial blow-up. Note that this result only holds for answer-guarded cGFP formulas (not arbitrary cGFP formulas).

Lemma 23. *Let ϕ be a answer-guarded* cGFP *formula without simultaneous fixpoints. We can construct an equivalent ϕ' in weak* GN*-normal form such that $[\![\phi]\!] = [\![\phi']\!]$ and*

- *$|\phi'|$ is polynomial in $|\phi|$;*
- *$\mathrm{width}(\phi') = \mathrm{width}(\phi)$;*
- *$\mathrm{ad}(\phi') = \mathrm{ad}(\phi)$;*
- *$\mathrm{rank}_{\mathrm{CQ}}(\phi') = 1$.*

Proof: We proceed by structural induction on ϕ.

- Assume ϕ is atomic. Then set $\phi' := \phi$.
- Assume ϕ is $\neg\eta(\boldsymbol{y})$. Then set $\phi' := \mathrm{gdd}(\boldsymbol{y}) \wedge \neg\eta'(\boldsymbol{y})$.
- Assume ϕ is $\phi_1 \wedge \phi_2$. Inductively, construct ϕ'_i and set $\phi' := \phi'_1 \wedge \phi'_2$. Although $\phi'_1 \wedge \phi'_2$ is a CQ-shaped subformula with two conjuncts, it does not contribute to the CQ-rank, since there is no outer quantification. Similarly for ϕ of the form $\phi_1 \vee \phi_2$.
- Assume ϕ is $\exists \boldsymbol{x}.\beta(\boldsymbol{xy}) \wedge \eta(\boldsymbol{xy})$. Inductively, construct η' for $\beta(\boldsymbol{xy}) \wedge \eta(\boldsymbol{xy})$ and set $\phi' := \exists \boldsymbol{x}.\beta(\boldsymbol{xy}) \wedge \eta'(\boldsymbol{xy})$. Note this CQ-shaped formula has CQ-rank 1 because β guards all of the variables in η', and the width of the formula is unchanged.
- Assume ϕ is $[\lambda X, \boldsymbol{x}.\eta](\boldsymbol{y})$. Inductively, construct η' and set $\phi' := [\lambda X, \boldsymbol{x}.\, \mathrm{gdd}(\boldsymbol{x}) \wedge \eta'](\boldsymbol{y})$. Correctness can be seen by observing that (i) $\boldsymbol{y}$ must be guarded by assumption that ϕ is answer-guarded (ii) any occurrence of $X\boldsymbol{z}$ inside η either satisfies $\boldsymbol{z} \subseteq \boldsymbol{y}$ so $\boldsymbol{z}$ is guarded, or $\boldsymbol{z} \not\subseteq \boldsymbol{y}$ so $X\boldsymbol{z}$ must appear inside some guarded quantification in η and hence $\boldsymbol{z}$ is guarded.

This transformation satisfies $[\![\phi]\!] = [\![\phi']\!]$. The other properties are straightforward to check. ∎

We now compute the size of the closure set for a formula in weak GN-normal form.

Lemma 24. *Given a* cGNFP *formula ϕ' in weak* GN*-normal form $|\mathrm{cl}_+(\phi')| \leq f(m) \cdot 2^{f(kr)}$ where $m = |\phi'|$, $k = \mathrm{width}(\phi')$, $r = \mathrm{rank}_{\mathrm{CQ}}(\phi')$ and f is a polynomial function independent of ϕ'. These bounds apply even when ϕ' is given as a DAG and m is the size of the DAG-representation.*

Proof: We have $\mathrm{gdd}(\boldsymbol{x}) \in \mathrm{cl}_+(\phi')$ for every vector of k variable names Hence, there are k^k guardedness predicates in $\mathrm{cl}_+(\phi')$.

There are at most m subformulas of ϕ', all of which appear in $\mathrm{cl}_+(\phi')$. However, each CQ-shaped subformula ψ can contribute additional formulas to the closure set due to specializations.

Fix some CQ-shaped subformula ψ of ϕ'. There are at most 2^r additional CQ-shaped subformulas ψ' obtained by choosing some subset of the conjuncts in ψ. In each ψ', there are at most 2^k choices of the inside variables, and k^k ways to partition the outside variables resulting in specializations ψ'' of ψ'. Notice that these specializations ψ'' only have CQ-shaped subformulas resulting from taking some subset of the conjuncts in the original CQ-shaped formula ψ, so these formulas have already been accounted for. Hence, ψ contributes at most $2^r 2^k k^k$ new CQ-shaped subformulas to $\mathrm{cl}_+(\phi')$.

Overall, this means $|\mathrm{cl}_+(\phi')| \leq m \cdot 2^r 2^k k^k + k^k$. ∎

Putting the conversion lemmas and the previous lemma together, we can get the stated bounds on the size of closure set for Proposition 7. The cGFP bounds follow because the CQ-rank of an answer-guarded cGFP formula is 1.

Simultaneous fixpoints. As mentioned in Proposition 7, these bounds apply even when the original formula uses simultane-

ous fixpoints. The idea is that we would first adapt the transformations in Lemmas 22 or 23 to handle formulas with simultaneous fixpoints, and then use a standard procedure known as the Bekič principle to eliminate any simultaneous fixpoints (see, e.g., [13]). This procedure preserves alternation-depth and counting alternation-depth. It can distort the values of the function defined by the formula, but it preserves the function up to $\approx$. The elimination can also result in an exponential blow-up in the size of the formula due to the duplication of subformulas, but by using a DAG-representation, this can be done with only a polynomial increase in the size. Finally, we can observe that the bounds on the size of the closure set given in Lemma 24 apply even when the formula is presented as a DAG. Hence, by using a DAG-representation for the intermediate weak GN-normal form formula, Proposition 7 applies even to formulas that have simultaneous fixpoints.

*B. Proofs of Theorem 9 and Corollary 10 (*cGNFP *to Automata)*

First, recall the statement of Theorem 9:

> Let φ be a sentence in cGNFP$[\sigma]$ in weak GN-normal form such that $\mathrm{width}(\varphi) = k$. Then there is a B $\wedge$ parity tree automaton $\mathcal{A}_\varphi$ using directions $\{0, \updownarrow\}$ such that for all consistent $\mathbb{K}_{\sigma,k}$-trees t,
>
> $$[\![\varphi]\!](\mathfrak{D}(t)) = n \quad \text{iff} \quad [\![\mathcal{A}_\varphi]\!](t) = n.$$
>
> The size of $\mathcal{A}_\varphi$ is exponential in $|\mathrm{cl}_+(\varphi, U_k)|$, but the number of states is polynomial in $|\mathrm{cl}_+(\varphi, U_k)|$. Moreover, $\mathcal{A}_\varphi$ uses priorities from $\{0, \dots, 2 \cdot \mathrm{ad}(\varphi) + 1\}$, and counters from $\{\gamma_X : X \text{ is a } \mu^N\text{-variable in } \varphi\}$.

We will give the construction that proves this, and later analyze it further to obtain the corollary about special classes mentioned in Corollary 10, which we will review below.

For the inductive proof we must deal with formulas with free-variables, so we introduce some more notation for this.

For free second-order variables $\boldsymbol{Z}$, each $Z \in \boldsymbol{Z}$ induces additional unary predicates $Z_{\boldsymbol{a}}$ for each free second-order variable $Z \in \boldsymbol{Z}$ with arity j and for each $\boldsymbol{a} \in U_k^j$. We write $\hat{\boldsymbol{Z}}$ for these new predicates, and write $(t, \hat{\boldsymbol{Z}})$ for the annotation of a tree encoding$t$ with information about these new predicates. In order for $(t, \hat{\boldsymbol{Z}})$ to be consistent, the consistency requirements described before must be satisfied not only for relations $R \in \sigma$ but also for $Z \in \boldsymbol{Z}$. For such a consistent tree, we write $\mathfrak{D}(\hat{Z})$ for $\{[v, \boldsymbol{a}] : Z_{\boldsymbol{a}}(v)\}$ in $\mathfrak{D}(t)$, the decoding of relation Z in $\mathfrak{D}(t)$.

Likewise, for free first-order variables, we need to indicate both a node and a name for which to evaluate these variables at. For notational convenience, given vectors $\boldsymbol{v} = v_1 \dots v_j$ and $\boldsymbol{a} = a_1 \dots a_j$, we write $[\boldsymbol{v}, \boldsymbol{a}]$ for $[v_1, a_1], \dots, [v_j, a_j]$. Likewise, given a single node v and a vector $\boldsymbol{a} = a_1 \dots a_j$, we write $[v, \boldsymbol{a}]$ for $[v, a_1], \dots, [v, a_j]$. Finally, we write $[\boldsymbol{v}, \boldsymbol{a}] = [w, \boldsymbol{a}]$ if $[v_i, a_i] = [w, a_i]$ for all i.

We now prove two technical lemmas leading to Theorem 9.

As a first step, it is straightforward to construct an automaton that checks a UCQ-shaped query when running on trees annotated with the answer-guarded subqueries. The idea is that the automaton guesses a specialization of each CQ, and uses the annotations to check that this specialization is actually realized.

Lemma 25. *Let $\psi(\boldsymbol{a}, \boldsymbol{Z}, \boldsymbol{S})$ be a UCQ-shaped formula from $\mathrm{cl}_+(\varphi, U_k)$ with every answer-guarded subquery χ replaced by a new predicate S_χ (we write $\boldsymbol{S}$ for this set of new predicates). There exists a parity tree automaton $\mathcal{C}_\psi$ such that for all consistent $\mathbb{K}_{\sigma,k}$-trees $(t, \hat{\boldsymbol{Z}}, \hat{\boldsymbol{S}})$ and for all nodes $v \in \mathrm{dom}(t)$,*

$$[\![\psi]\!](\mathfrak{D}(t), [v, \boldsymbol{a}], \mathfrak{D}(\hat{\boldsymbol{Z}}), \mathfrak{D}(\hat{\boldsymbol{S}})) = 0 \quad \textit{iff} \quad [\![\mathcal{C}_\psi]\!]_v(t, \hat{\boldsymbol{Z}}, \hat{\boldsymbol{S}}) = 0.$$

The size of $\mathcal{C}_\psi$ is exponential in $|\mathrm{cl}_+(\psi, U_k)|$, but the number of states in $\mathcal{C}_\psi$ is polynomial in $|\mathrm{cl}_+(\psi, U_k)|$. Moreover, $\mathcal{C}_\psi$ is weak and uses no counters.

Proof: Assume we have constructed automata for each CQ in ψ, of size exponential in k and r. Then it is easy to create an automaton for ψ of the desired size by taking the disjoint union of the automata for each CQ.

Hence, it suffices to show that for each CQ-shaped formula ψ, we can construct an automaton of size polynomial in $|\mathrm{cl}_+(\psi, U_k)|$.

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

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

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

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

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

The constructed automaton is weak since there is no cycle that visits both even and odd priorities. Correctness of the construction follows from the following results about specializations. We say that a specialization is *non-trivial* if either there are no outside variables (thus the specialization is only χ_0), or χ_0 is non-empty, or the partition of the outside variables is non-trivial ($l > 1$).

Claim 26. *Let $\phi(\boldsymbol{y}) \in$ GNFP be a CQ-shaped formula $\exists \boldsymbol{x} . \bigwedge_i \psi_i(\boldsymbol{xy})$. Given a structure $\mathfrak{M}$ and a tree decomposition T of $\mathfrak{M}$, if there exists a node v with $\boldsymbol{b} \subseteq T(v)$ and $\mathfrak{M}, \boldsymbol{b} \models \phi(\boldsymbol{y})$, then there is a non-trivial specialization $\phi'(\boldsymbol{yz})$*

of ϕ and a node w with a tuple $\boldsymbol{c}$ such that $\boldsymbol{bc} \in T(w)$ and $\mathfrak{M}, \boldsymbol{bc} \models \phi'(\boldsymbol{yz})$.

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

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

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

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

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

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

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

In the other direction, every specialization of ϕ logically implies ϕ.

Claim 27. *Let $\phi(\boldsymbol{y})$ be a formula of the form $\exists \boldsymbol{x}. \bigwedge_i \psi_i(\boldsymbol{xy})$. For all structures $\mathfrak{M}$ and for all specializations $\phi'(\boldsymbol{yz})$ of ϕ, if $\mathfrak{M}, \boldsymbol{bc} \models \phi'(\boldsymbol{yz})$, then $\mathfrak{M}, \boldsymbol{b} \models \phi(\boldsymbol{y})$.*

This concludes the proof of Lemma 25. ∎

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

Lemma 28. *Let $\psi(\boldsymbol{a}, \boldsymbol{Z}) \in \mathrm{cl}_+(\varphi, U_k)$. If ψ appears with polarity p in φ, then there exists a* B $\wedge$ *parity automaton $\mathcal{B}^p_\psi$ such that for all consistent $\mathbb{K}_{\sigma,k}$-trees $(t, \hat{\boldsymbol{Z}})$ and for all nodes $v \in \mathrm{dom}(t)$,*

$$[\![p\psi]\!](\mathfrak{D}(t), [v, \boldsymbol{a}], \mathfrak{D}(\hat{\boldsymbol{Z}})) = n \;\; \textit{iff} \;\; [\![\mathcal{B}^p_\psi]\!]_v(t, \hat{\boldsymbol{Z}}) = n.$$

The size of $\mathcal{B}^p_\psi$ is exponential in $|\mathrm{cl}_+(\psi, U_k)|$, but the number of states in $\mathcal{B}^p_\psi$ is polynomial in $|\mathrm{cl}_+(\psi, U_k)|$. Moreover, $\mathcal{B}^p_\psi$ uses priorities from $\{0, 1, \dots, 2 \cdot \mathrm{ad}(\psi) + 1\}$, and counters from $\{\gamma_X : X \text{ is a } \mu^N\text{-variable in } \psi\}$.

Proof: We describe the construction of $\mathcal{B}^p_\psi$, and then give the proof of correctness.

We proceed by induction on the structure of ψ.

- Assume ψ is a UCQ-shaped subformula. If $p = +$ (respectively, $p = -$), then let $\mathcal{B}^p_\psi$ be the automaton where Eve (respectively, Adam) chooses a valuation $\hat{\boldsymbol{S}}$, $\mathcal{C}_\psi$ (respectively, the dual of $\mathcal{C}_\psi$) from Lemma 25 is simulated, and Adam (respectively, Eve) can choose to launch $\mathcal{B}^p_{\psi''}$ from a node w if $S_{\psi''}(w)$.
- Assume ψ is of the form $\alpha(\boldsymbol{a}) \wedge p'\psi'$, where $p' \in \{+, -\}$ indicates whether ψ' appears positively or negatively in ψ. If $p = +$ (respectively, $p = -$), then Eve (respectively, Adam) chooses a conjunct. If $\alpha(\boldsymbol{a})$ is selected, then the automaton immediately accepts if $\alpha(\boldsymbol{a})$ appears (respectively, does not appear) in the tree encoding at v. Otherwise, simulate $\mathcal{B}^{p'}_{\psi'}$.
- Assume ψ is of the form $\mathrm{gdd}(\boldsymbol{a}) \wedge \psi'$. If $p = +$ (respectively, $p = -$), then Eve (respectively, Adam) chooses a conjunct. If $\mathrm{gdd}(\boldsymbol{a})$ is selected, then simulate $\mathcal{B}^p_\chi$ where χ is the UCQ equivalent to $\mathrm{gdd}(\boldsymbol{a})$. Otherwise, simulate $\mathcal{B}^p_{\psi'}$.
- Assume ψ is a fixpoint subformula $[\lambda X, \boldsymbol{x}. \mathrm{gdd}(\boldsymbol{x}) \wedge \psi'(\boldsymbol{x}, X, \boldsymbol{Z})](\boldsymbol{a})$ where $\lambda \in \{\mu, \mu^N\}$ and $\mathrm{al}_\varphi(X) = j$. If $p = +$ (respectively, $p = -$), then let $\mathcal{B}^p_\psi$ be the automaton where Eve (respectively, Adam) chooses valuation $\hat{X}$, $\mathcal{B}^p_{\mathrm{gdd}(\boldsymbol{a}) \wedge \psi'(\boldsymbol{a}, X, \boldsymbol{Z})}$ is simulated, and Adam (respectively, Eve) can choose to launch $\mathcal{B}^p_{\mathrm{gdd}(\boldsymbol{b}) \wedge \psi'(\boldsymbol{b}, X, \boldsymbol{Z})}$ from w if $X_{\boldsymbol{b}}(w)$. In case $X_{\boldsymbol{b}}(w)$ is challenged, the priority output is $2j - 1$ (respectively, $2j$) and counter $\gamma_{j'}$ is reset for any $X_{j'} \sqsubset_\varphi X_j$. In addition, if $\lambda = \mu^N$, then counter γ_j is incremented at the beginning of the simulation and any time some $X_{\boldsymbol{b}}(w)$ is challenged.

We focus on the interesting case in the proof when ψ is a fixpoint predicate $[\lambda X_i, \boldsymbol{x}.\eta](\boldsymbol{x})$. We assume that there are no free second-order variables in ψ, since these do not affect the arguments below. Let $\mathcal{G} := \mathcal{G}(\mathcal{B}^p_{\psi(\boldsymbol{a})}, t, v)$. In the arguments below, we also consider *approximant games* $\mathcal{G}(\mathcal{B}^p_{\eta(\boldsymbol{a})}, t, \hat{X}_i, v)$, which are intermediate games based on evaluating the body of the fixpoint with some valuation $\hat{X}_i$.

Case $p = +$ and $\lambda \in \{\mu, \mu^N\}$. We start by assuming $[\![p\psi]\!](\mathfrak{D}(t), [v, \boldsymbol{a}]) \leq n$. We show how to construct an n-winning strategy for Eve in $\mathcal{G}$.

There is some least ordinal α such that $\mathfrak{D}(t), n \models \eta^\alpha([v, \boldsymbol{a}])$. Moreover, α is a successor ordinal and $\mathfrak{D}(t), n \models \eta([v, \boldsymbol{a}])[\eta^{\alpha-1}(\boldsymbol{y})/X_i(\boldsymbol{y})]$. Consider the encoding of a valuation $\hat{X}_i$ for X_i such that $X_{i,\boldsymbol{b}}(w)$ holds iff $\mathfrak{D}(t), n \models \eta^{\alpha-1}([w, \boldsymbol{b}])$. By the inductive hypothesis, there is an n-winning strategy ζ' for Eve in the approximant game $\mathcal{G}^\alpha := \mathcal{G}(\mathcal{B}^p_{\eta(\boldsymbol{a})}, t, \hat{X}_i, v)$. We construct a strategy in $\mathcal{G}$ in which Eve

selects the valuation $\hat{X}_i$ defined above, and uses this strategy ζ' from $\mathcal{G}^\alpha$.

If Adam never challenges an annotation, then the play is n-winning by assumption on ζ'. Otherwise, if Adam challenges some $X_{i,\boldsymbol{b}}(w)$ in $\hat{X}_i$, then there is some least ordinal $\beta \leq \alpha - 1$ such that $\mathfrak{D}(t), n \models \eta^\beta([w, \boldsymbol{b}])$. From this position, we use the inductively-defined n-winning strategy in $\mathcal{G}^\beta := \mathcal{G}(\mathcal{B}^p_{\eta(\boldsymbol{b})}, t, \hat{X}_i, w)$ with the $\hat{X}_i$ valuation based on the $(\beta-1)$-approximant, and reason as before.

Continuing to reason in this fashion, Eve must eventually choose a valuation for $\hat{X}_i$ that is the empty valuation (otherwise $\alpha > \beta > \cdots$ would be an infinite descending chain, contradicting the well-foundedness of the ordinals). This means any play in the constructed strategy will eventually stabilize in some approximant game. This ensures that the parity condition is satisfied in any play in ζ. Moreover, when X_i is a μ^N-variable, this ensures that the value of counter γ_i is at most n.

However, it remains to show that any other counters used by $\mathcal{B}^p_{\eta(\boldsymbol{x})}$ never exceed value n. Notice that in between challenges by Adam, each counter in $\mathcal{B}^p_{\eta(\boldsymbol{x})}$ can be incremented at most n times, by the assumption on the inductively-defined strategies. Moreover, at each challenge, any counter that was used in the previous approximant game $\mathcal{G}^\alpha$, $\mathcal{G}^\beta$, etc. is reset (because any counter that is used in some play that eventually returns to X_i must correspond to a μ^N-variable X_j with $X_j \sqsubset_\varphi X_i$. Overall, this means that the constructed strategy in $\mathcal{G}$ is n-winning for Eve as desired.

Next, assume $[\![p\psi]\!](\mathfrak{D}(t), [v, \boldsymbol{a}]) > n$ and $\lambda = \mu^N$. We show how to construct an $(n+1)$-winning strategy for Adam in $\mathcal{G}$.

Consider the valuation $\hat{X}_i$ for X_i such that $X_{i,\boldsymbol{b}}(w)$ holds iff $\mathfrak{D}(t), n \models \eta^n([w, \boldsymbol{b}])$. Because $[\![p\psi]\!](\mathfrak{D}(t), [v, \boldsymbol{a}]) > n$, this means that $[v, \boldsymbol{a}]$ is not in this valuation. Let ζ' be an optimal strategy for Adam in $\mathcal{G}' := \mathcal{G}(\mathcal{B}^p_{\eta(\boldsymbol{a})}, t, \hat{X}_i, v)$, based on this valuation. Note that this strategy might not be $(n+1)$-winning for Adam.

We construct Adam's strategy in $\mathcal{G}$ so that he starts by playing from ζ'. If Eve deviates from the valuation for X_i described above (by guessing additional tuples), then Adam challenges this tuple, and starts using a new strategy based on an appropriate approximant game and the new valuation such that $X_{i,\boldsymbol{b}}(w)$ holds iff $\mathfrak{D}(t), n \models \eta^{n-1}([w, \boldsymbol{b}])$.

If Adam challenges at least n times, then the γ_i counter will achieve value $n+1$. Otherwise, the play stabilizes in some approximant game $\mathcal{G}'$ based on the m-th unfolding $(0 \leq m \leq n)$ of η for some challenged tuple $[w, \boldsymbol{b}]$. If Adam has an $(n+1)$-winning strategy in this game, then we are done. If Adam's optimal strategy is not $(n+1)$-winning in this approximant game, then there is an n-winning strategy for Eve in this approximant game selecting an X_i valuation with only tuples in approximant $m-1$. By the inductive hypothesis, this would imply that $\mathfrak{D}(t), n \models \eta^m([w, \boldsymbol{b}])$, which would contradict Adam's challenge of this particular tuple.

If $\lambda = \mu$, then the reasoning would be similar, starting with the valuation $\hat{X}_i$ for X_i such that $X_{i,\boldsymbol{b}}(w)$ holds iff $\mathfrak{D}(t), n \models [\mu X_i, \boldsymbol{x}.\eta]([w, \boldsymbol{b}])$. Let ζ' be an optimal strategy for Adam from $\mathcal{G}' := \mathcal{G}(\mathcal{B}^p_{\eta(\boldsymbol{a})}, t, \hat{X}_i, v)$, based on this valuation.

We construct Adam's strategy in $\mathcal{G}$ so that he starts by playing from ζ'. If Eve deviates from the valuation for X_i described above (by guessing additional tuples), then Adam challenges this tuple, and starts using a new strategy based on the appropriate approximant game (but with the same valuation).

If Adam challenges infinitely many times during the play, then it will be $(n+1)$-winning since it fails to satisfy the parity condition. Otherwise, the play stabilizes in some approximant game $\mathcal{G}'$ based on η for some challenged tuple $[w, \boldsymbol{b}]$. If Adam has an $(n+1)$-winning strategy in this game, then we are done. If Adam's optimal strategy is not $(n+1)$-winning in this approximant game, then there is an n-winning strategy for Eve in this approximant game selecting an X_i valuation with only tuples that were actually in the least fixpoint (tuples that would not be challenged by Adam). By the inductive hypothesis, this would imply that $\mathfrak{D}(t), \mathfrak{D}(\hat{X}_i), n \models \eta([w, \boldsymbol{b}])$, so $\mathfrak{D}(t), n \models [\mu X_i, \boldsymbol{x}.\eta]([w, \boldsymbol{b}])$, a contradiction.

Case $p = -$ and $\lambda = \mu$. Recall that μ^N cannot appear negatively in φ, so this is the only case with $p = -$.

Assume $[\![p\psi]\!](\mathfrak{D}(t), [v, \boldsymbol{a}]) \leq n$. We describe an n-winning strategy ζ for Eve in $\mathcal{G}$.

Let $\hat{X}^n_i$ be the valuation for X_i such that $X_{i,\boldsymbol{b}}(w)$ holds iff $[\![p\psi]\!](\mathfrak{D}(t), [w, \boldsymbol{b}]) > n$. Let ζ' be an optimal strategy for Eve in $\mathcal{G}(\mathcal{B}^p_{\eta(\boldsymbol{a})}, t, \hat{X}^n_i, v)$, based on this valuation. We construct Eve's strategy ζ in $\mathcal{G}$ so that she starts by playing from ζ'. If Adam asserts some $X_{i,\boldsymbol{b}}(w)$ that is not in $\hat{X}^n_i$, then Eve challenges this tuple, starts using an optimal strategy in $\mathcal{G}(\mathcal{B}^p_{\eta(\boldsymbol{b})}, t, \hat{X}^n_i, w)$, and we continue as before.

First observe that Eve's strategy in any of the approximant games must be n-winning. Suppose not. Then $[\![\mathcal{B}^p_{\eta(\boldsymbol{b})}]\!](t, w, \hat{X}^n_i) > n$, which by the inductive hypothesis implies that $[\![p\eta]\!](t, [w, \boldsymbol{b}], \hat{X}^n_i) > n$. This in turn implies that $[\![\neg\eta[\psi(\boldsymbol{y})/X_i(\boldsymbol{y})]]\!](\mathfrak{D}(t), [w, \boldsymbol{b}]) > n$. But $[\![\neg\eta[\psi(\boldsymbol{y})/X_i(\boldsymbol{y})]]\!] = [\![\neg\psi]\!]$, so we have $[\![\neg\psi]\!](\mathfrak{D}(t), [w, \boldsymbol{b}]) > n$, which contradicts the fact that in the constructed strategy ζ, Eve only challenges tuples not in the valuation $\hat{X}^n_i$. Moreover, when Eve challenges some new tuple, all of the counters in the previous approximant game are reset.

If Eve challenges infinitely many times during the play, then the play is n-winning since $2j$ is the highest even priority occurring infinitely often, where $j = \mathrm{al}_\varphi(X_i)$. Otherwise, the play eventually stabilizes in some approximant game for some challenged tuple, and as argued above, the strategy in this approximant game must be n-winning.

Next, assume $[\![p\psi]\!](\mathfrak{D}(t), [v, \boldsymbol{a}]) > n$. We show how to construct an $(n+1)$-winning strategy ζ for Adam in $\mathcal{G}$.

For ordinals α, let $\hat{X}^\alpha_i$ denote the valuation for X_i such that $X_{i,\boldsymbol{b}}(w)$ iff $[\![\neg\eta^\alpha]\!](\mathfrak{D}(t), [w, \boldsymbol{b}]) > n$.

Observe that $[\![\neg\psi]\!](\mathfrak{D}(t), [v, \boldsymbol{a}]) > n$ implies that $\mathfrak{D}(t), [v, \boldsymbol{a}], n \not\models \neg\psi'$ where ψ' is the result of substituting the n-th approximants for each μ^N-subformula in ψ. Hence, $\mathfrak{D}(t), [v, \boldsymbol{a}] \models \psi'$, If η' is the result of doing this substitution

in η, then $\mathfrak{D}(t), [v, \boldsymbol{a}] \models (\eta')^\alpha$ for some least ordinal α. But this means that $\mathfrak{D}(t), [v, \boldsymbol{a}] \not\models \neg(\eta')^\alpha$, and $\mathfrak{D}(t), n \not\models \neg\eta^\alpha$ and hence $[\![\neg\eta^\alpha]\!](\mathfrak{D}(t), [v, \boldsymbol{a}]) > n$.

Hence, there is some least ordinal α such that $[\![\neg\eta^\alpha]\!](\mathfrak{D}(t), [v, \boldsymbol{a}]) > n$. Moreover, α is a successor ordinal and $[\![\neg\eta[\eta^{\alpha-1}/X_i]]\!](\mathfrak{D}(t), [v, \boldsymbol{a}]) > n$. Hence, $[\![\neg\eta]\!](\mathfrak{D}(t), [v, \boldsymbol{a}], \hat{X}_i^{\alpha-1}) > n$. By the inductive hypothesis, there is an $(n+1)$-winning strategy ζ' for Adam in $\mathcal{G}(\mathcal{B}^p_{\eta(\boldsymbol{a})}, t, \hat{X}_i^{\alpha-1}, v)$. We construct a strategy ζ in $\mathcal{G}$ in which Adam selects the valuation $\hat{X}_i^{\alpha-1}$ defined above, and uses this strategy ζ' from $\mathcal{G}^\alpha$. If Eve challenges some $X_{i,\boldsymbol{b}}(w)$ in the valuation, then there is some least ordinal $\beta \leq \alpha - 1$ such that $[\![\neg\eta^\beta]\!](\mathfrak{D}(t), [w, \boldsymbol{b}]) > n$. Using similar reasoning as above, Adam has an $(n+1)$-winning strategy in $\mathcal{G}(\mathcal{B}^p_{\eta(\boldsymbol{b})}, t, \hat{X}_i^{\beta-1}, w)$. Hence, when Eve challenges some tuple, Adam switches to selecting the valuation $\hat{X}_i^{\beta-1}$ and playing his inductively-defined $(n+1)$-winning strategy from $\mathcal{G}(\mathcal{B}^p_{\eta(\boldsymbol{b})}, t, \hat{X}_i^{\beta-1}, w)$. Adam continues updating the approximant and his strategy in the approximant game like this after each challenge by Eve.

On any play, there can be only a finite number of challenges by Eve (otherwise $\alpha > \beta > \cdots$ would be an infinite descending chain, contradicting the well-foundedness of the ordinals). This means that any play in the constructed strategy will eventually stabilize in some approximant game, where Adam can play his n-winning strategy.

This concludes the proof of Lemma 28. ∎

The desired B∧parity automaton $\mathcal{A}_\varphi$ in Theorem 9 checking φ in $\mathfrak{D}(t)$ is the result of running the B $\wedge$ parity automaton $\mathcal{B}^+_\varphi$ starting from the root of t. This concludes the proof of Theorem 9.

Refined analysis. Recall that Corollary 10 states results for both general cGNFP formulas and for restricted classes:

> Let φ be a sentence in cGNFP[σ] in weak GN-normal form such that $\mathrm{width}(\varphi) = k$. Then there exists a B $\wedge$ parity tree automaton $\mathcal{B}_\varphi$ such that $[\![\varphi]\!]$ is bounded over all σ-structures iff $[\![\mathcal{B}_\varphi]\!]$ is bounded over all $\mathbb{K}_{\sigma,k}$-trees.
>
> The size of $\mathcal{B}_\varphi$ is exponential in $|\mathrm{cl}_+(\varphi, U_k)|$, but the number of states is polynomial in $|\mathrm{cl}_+(\varphi, U_k)|$. The number of priorities and counters is linear in $|\varphi|$.
>
> Moreover, $\mathcal{B}_\varphi$ is a dist $\wedge$ parity automaton if φ is in distance cGNFP, and is counter-weak if φ is alternation-free.

In order to conclude the last sentence, we will need further analysis of the translation given in Lemma 28.

Alternation-free case. Observe that if φ is alternation-free, then the alternation-level of every fixpoint variable in φ must be 0. This means that in any cycle in a run of $\mathcal{B}^+_\varphi$ that visits both even and odd priorities, there must be some μ^N-variable that subsumes the formulas responsible for the even priorities and the formulas responsible for the odd priorities. Hence, there is some counter that is incremented and not reset, so the resulting automaton is counter-weak. This means that if φ is alternation-free, then $\mathcal{B}_\varphi$ is counter-weak, as stated in Corollary 10.

Optimization for distance cGNFP *case.* In the automaton $\mathcal{B}^+_\varphi$ constructed using Lemma 28, a different counter is used for each μ^N-variable X_i in φ. This leads to a slightly cleaner proof because it means the bound n is preserved exactly between $[\![\varphi]\!]$ and $[\![\mathcal{B}^+_\varphi]\!]$.

When the original formula is in distance cGNFP, however, we can optimize the construction so that the resulting automaton is a dist $\wedge$ parity automaton, i.e. an automaton using a single counter that is only incremented or left unchanged, never reset. This ensures that in Corollary 10 we can construct a dist∧parity automaton when starting from a distance cGNFP formula.

Before we describe the optimization, we show that when the construction in Lemma 28 is applied to a distance cGNFP formula ψ, the following property holds.

Claim 29. *In any n-winning play for Eve in $\mathcal{G}(\mathcal{B}^p_{\psi(\boldsymbol{a})}, t, \hat{\boldsymbol{Z}}, v)$, the total number of increments is at most $(2n)^j$ where j is the number of μ^N-variables in ψ (not including any μ^N-variables that are free).*

Proof: The proof is by induction on the structure of ψ. Again, the interesting case is when ψ is a fixpoint predicate $[\lambda X_i, \boldsymbol{x}.\eta](\boldsymbol{x})$.

If $p = +$ and $\lambda = \mu^N$, then we observed in the proof above that any play must stabilize in an approximant game after at most n increments of counter γ_i. By the inductive hypothesis, each of the at most n different approximant games can contribute $(2n)^{j-1}$ increments. Hence, the total number of increments is $n(2n)^{j-1} + n \leq (2n)^j$.

If $p \in \{+, -\}$ and $\lambda = \mu$, then by the definition of distance cGNFP, X_i cannot occur free in η (otherwise, the counting level of X_i would be greater than 0). Consider a partial play ending in a challenge that starts a new approximant game. Such a challenge is only possible in the approximant game if the automaton is currently processing some formula with X_i as a dependency (i.e. X_i occurs free). Thus, as soon as any counter is incremented in the approximant game, it means that no additional challenges are possible (by definition of distance cGNFP, no μ^N-variable can depend on a least or greatest fixpoint variable). Hence, the only approximant game that can contribute increments during the play is the approximant game in which the play stabilizes. This has at most $(2n)^j$ increments by the inductive hypothesis. This concludes the proof of the claim. ∎

This means we can replace the individual counters by a single counter γ that is incremented any time one of the original counters would have been incremented, but is otherwise left unchanged. If we write $[\![\mathcal{B}]\!]$ for the original automaton and $[\![\mathcal{B}']\!]$ for this optimized version, then $[\![\mathcal{B}]\!]$ is bounded (by n) over all consistent $\mathbb{K}_{\sigma,k}$-trees iff $[\![\mathcal{B}']\!]$ is bounded (by $(2n)^j$) over all consistent $\mathbb{K}_{\sigma,k}$-trees.

Using this optimization of the translation in Lemma 28, this means that if φ is in distance cGNFP, then $\mathcal{B}_\varphi$ is a dist∧parity automaton, as stated in Corollary 10.

C. Extension of Theorem 9 to handle constants and equality

Thus far, we have been assuming that the cGNFP formulas do not make use of constants or equality.

We can introduce constants with minimal changes. It is straightforward to prove that for a cGNFP formula φ with constants we can restrict to tree-like structures as before. However, as part of the definition of a consistent tree, we enforce that the encodings of these tree-like structures include information about the constants *in every node* of the encoding. This ensures that information about these constants is accessible to the automaton at any point during its run. This is important because constants do not have to be guarded. Hence, when it comes time to checking some answer-guarded subquery $\psi(\boldsymbol{x})$, we may need to be tracking the valuation of $\boldsymbol{x}$, which includes some guarded set of element names as well as other constants that do not need to be guarded. Having this information about the constants at each node means that it is still possible to launch an automaton checking this answer-guarded subquery subformula from a single node in the tree. The state set of the constructed automata is now based on $\mathrm{cl}_+(\varphi, U_k \cup C)$ where C is the set of constants used by φ. Note that the size of $\mathrm{cl}_+(\varphi, U_k \cup C)$ is $(2k + |C|)^k \cdot |\mathrm{cl}_+(\varphi)|$ (versus $(2k)^k \cdot |\mathrm{cl}_+(\varphi)|$ without constants), but this does not affect the overall complexity.

Accommodating equality requires some additional technical work. The idea is to first normalize the use of equality in the cGNFP formula. This can be done so that the conversion of the cGNFP to weak GN-normal form and this equality-normal form is exponential in the size of the input (similar to the general case in Theorem 7). Based on this, we can then place additional requirements on the use of equality in the encodings of the tree decompositions of these equality-normalized formulas. After these adjustments, we can treat equality like the other relations during the automata construction. We now describe this equality-normalization process in more detail.

The first thing we show is that any cGNFP formula with equality can be converted to a form with a very limited use of equality. Let us say that a cGNFP formula ϕ is *equality-normalized* if

(i) every occurrence of $R\,\boldsymbol{t}$ or $X\,\boldsymbol{t}$ in ϕ appears in conjunction with

$$\text{all-distinct}(\boldsymbol{t}) := \bigwedge_{t \in \boldsymbol{t}} (t = t) \wedge \bigwedge_{t,t' \in \boldsymbol{t}, t \neq t'} \neg(t = t'),$$

(ii) whenever equalities are used as guards for negations, then these equality guards are of the form $x = x$, and

(iii) every occurrence of equality in ϕ is either an equality comparison of constants, or comes from (i) or (ii).

Let $\phi(\boldsymbol{x})$ be a GNF formula containing constants $\boldsymbol{e}$, and let $\equiv$ be any equivalence relation over $\boldsymbol{x} \cup \boldsymbol{e}$. We denote by $\xi_\equiv$ the first-order formula

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

that is, the conjunction of all equalities and inequalities corresponding to $\equiv$. Note that this formula does not necessarily belong to GNF as it may contain unguarded inequalities.

Lemma 30. *Let $\phi(\boldsymbol{x})$ be a* GNF *formula in weak* GN*-normal form containing constants $\boldsymbol{e}$, and let $\equiv$ be any equivalence relation over the free variables and constants in ϕ. We can construct a DAG-representation of an equality-normalized $\phi'_\equiv(\boldsymbol{x})$ in* GN*-normal form such that*

- $\models \forall \boldsymbol{x}\big(\xi_\equiv(\boldsymbol{x}) \to (\phi(\boldsymbol{x}) \leftrightarrow \phi'_\equiv(\boldsymbol{x}))\big)$*;*
- *the size of the DAG-representation of $\phi'_\equiv$ is at most $p(|\phi|^k)$, where k is the width of ϕ, and p is a polynomial function independent of ϕ;*
- $\mathrm{width}(\phi'_\equiv) \leq \mathrm{width}(\phi)$*;*
- $\mathrm{rank}_{\mathrm{CQ}}(\phi'_\equiv) \leq \mathrm{rank}_{\mathrm{CQ}}(\phi)$.

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

For each $\equiv$-equivalence class, we fix an arbitrary representative — a constant whenever possible. We replace all occurrences of each variables and constant in $\boldsymbol{x} \cup \boldsymbol{e}$ by the representative of its equivalence class. Next, we replace subformulas of the form $s = t$, with $s, t \in \boldsymbol{x} \cup \boldsymbol{e}$, by $\top$ if $s = t$ and by $\bot$ otherwise; finally, we conjoin every relational atom containing distinct $s, t \in \boldsymbol{x} \cup \boldsymbol{e}$ with $\neg(s = t)$.

At this point, the conditions (i) and (ii) in the definition of equality-normalized formulas are satisfied *for equalities that have a free variables or constants on both sides*. It remains only to take care of the equalities that involve a quantified variable.

Consider any subformula of the form

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

We will essentially do a case distinction, for each quantified variable $y_i \in \boldsymbol{y}$, of the possible values that y_i may take. More precisely, we replace $\psi(\boldsymbol{z})$ by the disjunction, for each map $f : \boldsymbol{y} \to (\boldsymbol{y} \cup \boldsymbol{z} \cup \boldsymbol{e})$, of the formula ψ_f obtained from ψ by (i) replacing each $y_i \in \boldsymbol{y}$ by $f(y_i)$, (ii) replacing $y_i = y_j$ by $\top$ if $f(y_i) = f(y_j)$ and by $\bot$ otherwise, (iii) replacing $y_i = t$ or or $t = y_i$ for $t \in \boldsymbol{z} \cup \boldsymbol{e}$ by $\top$ if $f(y_i) = t$ and by $\bot$ otherwise (and dropping the quantifiers corresponding to quantified variables that no longer occur in the formula). Finally, we conjoin every atom $R(\boldsymbol{t})$ with $\bigwedge_{t \in \boldsymbol{t}} (t = t) \wedge \bigwedge_{t,t' \in \boldsymbol{t}, t \neq t'} \neg(t = t')$. This clearly preserves the semantics of the formula over structures satisfying $\xi_\equiv$. It does not change the CQ-rank because $\boldsymbol{t}$ is guarded. The size blowup involved in this procedure is at most exponential in the width of ϕ.

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

weak GN-normal form. The bound on the width is immediate from the construction. ∎

Combining Lemma 30 with Lemma 22 we can obtain the following result.

Proposition 31. *Let ϕ be a* cGNFP *(respectively,* cGFP*) sentence not necessarily in weak* GN*-normal form. We can construct a DAG-representation of an equality-normalized ϕ' in weak* GN*-normal form such that*

- $[\![\phi]\!] \approx [\![\phi']\!]$*;*
- *the size of the DAG-representation of ϕ' is at most exponential in $|\phi|$ (respectively, $p(|\phi|)^k$ where k is the width of ϕ and p is a polynomial function independent of ϕ);*
- $\mathrm{width}(\phi') \leq |\phi|$ *(respectively,* $\mathrm{width}(\phi') \leq \mathrm{width}(\phi)$*);*
- $\mathrm{rank}_{\mathrm{CQ}}(\phi') \leq |\phi|$ *(respectively,* $\mathrm{rank}_{\mathrm{CQ}}(\phi') = 1$*).*

There is no change in the alternation-depth and counting alternation-depth.

Proof sketch: Without loss of generality, we can assume ϕ is a fixpoint subformula (since we can always wrap the sentence in a new 0-ary fixpoint predicate).

We proceed with the conversion starting from the innermost fixpoint formulas. For $[\lambda X, \boldsymbol{x}.\psi(\boldsymbol{x}, X, \boldsymbol{Y})]$, we create a simultaneous λ-fixpoint on variables $X_{\equiv_1}, \ldots, X_{\equiv_m}$ for each equivalence class $\equiv_i$ over $\boldsymbol{x} \cup \boldsymbol{e}$ (where $\boldsymbol{e}$ are the constants in ψ). The fixpoint body corresponding to $X_{\equiv_i}$ is the weak GN-normal form formula equivalent to $\mathrm{gdd}(\boldsymbol{x}) \wedge (\xi_{\equiv_i} \to \psi'_{\equiv_i})$, obtained by applying Lemma 22 or Lemma 23 to convert ψ to weak GN-normal form and then applying Lemma 30 to convert to equality normal form based on $\equiv_i$. Fixpoint variables or other fixpoint subformulas are treated as atomic during this transformation, so any occurrence of $X\boldsymbol{t}$ will be conjoined with $\text{all-distinct}(\boldsymbol{t})$. We replace $X\boldsymbol{t}$ with $X_{\equiv_i}\boldsymbol{t}$ where $\equiv_i$ describes the equivalence class over $\boldsymbol{x} \cup \boldsymbol{e}$ induced by $\text{all-distinct}(\boldsymbol{t})$. Other fixpoint variables or fixpoint subformulas are treated in a similar way. Repeating this procedure for all fixpoint subformulas, we get a DAG-representation of the desired size. This is only possible because we are using a DAG-representation, and hence can avoid additional exponential blow-ups from the nesting of fixpoints.

Finally, the simultaneous fixpoints can be eliminated with only a polynomial blow-up, again taking advantage of the DAG representation of the formula.

Overall, this results in a DAG-representation for ϕ' of the desired size. The other properties of ϕ' are straightforward to check. The difference in bounds between cGNFP and cGFP comes from the difference between Lemma 22 and Lemma 23.

We note that the value of $[\![\phi]\!]$ is not preserved exactly because we split each μ^N fixpoint variable X into $X_{\equiv_1}, \ldots, X_{\equiv_m}$ as described above. However, this distortion in the value is acceptable up to $\approx$-equivalence, so $[\![\phi]\!] \approx [\![\phi']\!]$. ∎

We say that a consistent tree t is *equality-trivial* if for all nodes v in t (i) $s = s'$ is asserted at v only if s and s' are the same term, and (ii) $\neg(s = s')$ is asserted at v only if s and s' are distinct terms. Note that an encoding of a tree decomposition is naturally equality-trivial, and if t is an equality-trivial consistent tree then distinct terms are realized by distinct elements in $\mathfrak{D}(t)$. Moreover, for equality-normalized sentences, equality-trivial consistent trees contain all of the information necessary to evaluate whether the sentence holds in the corresponding structure, just treating equality as any other relation.

Therefore, given some cGNFP sentence with equality and constants, we can convert into weak GN-normal form and equality-normalized form using Proposition 31, and then use the automaton construction from Lemmas 25 and 28, treating equality like any other relation. Proposition 31 ensures that the conversion to this equality-normalized version and subsequent automaton construction can be done with the same complexity bounds as Theorem 9. Hence, we have the following extension of Theorem 9.

Theorem 32 (Extension of Theorem 9 with equality and constants)**.** *Let φ be a sentence in* cGNFP$[\sigma]$ *(possibly using equality and constants) such that* $\mathrm{width}(\varphi) = k$*. Then there is a* B$\wedge$*parity tree automaton $\mathcal{A}_\varphi$ using directions $\{0, \updownarrow\}$ such that $[\![\varphi]\!]$ is bounded over all σ-structures iff $[\![\mathcal{A}_\varphi]\!]$ is bounded over all equality-trivial consistent $\mathbb{K}_{\sigma,k}$-trees.*

The size of $\mathcal{A}_\varphi$ is exponential in $|\mathrm{cl}_+(\varphi, U_k)|$, but the number of states is polynomial in $|\mathrm{cl}_+(\varphi, U_k)|$. Moreover, $\mathcal{A}_\varphi$ uses priorities from $\{0, \ldots, 2 \cdot \mathrm{ad}(\varphi) + 1\}$, and counters from $\{\gamma_X : X \text{ is a } \mu^N\text{-variable in } \varphi\}$.

APPENDIX C
MISSING PROOFS FROM SECTION V

A. Proof of Proposition 13 (Positional strategies in $\overline{\text{dist} \wedge \text{parity}}$ games)

Observe that any $\overline{\text{dist} \wedge \text{parity}}$ game can be viewed as a $\overline{\text{distance}} \vee$ parity game simply by incrementing the priorities by one (turning 'good' even priorities into 'bad' odd priorities, and vice versa). Hence it suffices to prove the following:

> If Eve has an N-winning strategy in a $\overline{\text{distance}} \vee$ parity game $\mathcal{G}$, then Eve has an N-winning positional strategy in $\mathcal{G}$.

Consider a $\overline{\text{distance}} \vee$ parity game $\mathcal{G}$ where Eve has an N-winning strategy from the initial position v_0. Let O be the corresponding $\overline{\text{distance}} \vee$ parity objective. A play π in such a game is n-winning for Eve if it satisfies at least one of the following conditions:

- π satisfies the parity condition, or
- π has counter value at least n.

We associate to each position v in the game $\mathcal{G}$ the greatest $n \leq N$ such that Eve has a n-winning strategy in $\mathcal{G}$ starting from v; let W_n denote the set of positions associated with n.

Let O_n be the objective such that a play π is winning if π satisfies the parity condition or π has value at least n (note that this is a traditional boolean winning condition, rather than a cost objective). Let $\mathcal{G}_n$ be the boolean game with objective O_n obtained by restricting $\mathcal{G}$ to positions in $W_0 \cup \cdots \cup W_n$, replacing any transitions to W_m for $m > n$ with a winning exit transition, and any transition outside of $W_0 \cup \cdots \cup W_n$ with a losing exit transition. Technically, these winning (respectively, losing) exit transitions require adding a sink node to the game graph with even (respectively, odd) priority. Note that by definition of the winning regions W_i, Eve has a winning strategy in $\mathcal{G}_n$ starting from any position in W_n. We proceed by induction on $0 \leq n \leq N$ to show that

> there exists a positional strategy ζ_n that is winning in $\mathcal{G}_n$ starting from any position in W_n.

The desired result follows by using ζ_N in $\mathcal{G}$ starting from the initial position v_0, which is in W_N.

For the base case when $n = 0$, an arbitrary positional strategy will work since any play will always have counter value at least 0.

Now consider $n > 0$. Let $\mathcal{W}_n$ be the parity game obtained from $\mathcal{G}_n$ by restricting to positions in W_n, and replacing any transition with an increment that stays in W_n with a winning exit transition. Moreover, any transitions that become exit transitions (because their target is outside of W_n) are made winning or losing as follows:

- transitions to W_m for $m > n$ are winning;
- transitions to W_{n-1} that perform an increment are winning;
- transitions to W_{n-1} without an increment are losing;
- transitions to W_m for $m < n - 1$ are losing.

Note that $\mathcal{W}_n$ is a parity game since all increments were removed in the construction. The winning strategy in $\mathcal{G}_n$ starting from a position $v \in W_n$ induces a winning strategy in $\mathcal{W}_n$. Because $\mathcal{W}_n$ is a parity game, there is a positional winning strategy τ_n for Eve in $\mathcal{W}_n$ [26].

Using τ_n together with the inductively defined positional winning strategy ζ_{n-1}, we now construct a winning positional strategy for Eve in $\mathcal{G}_n$. The strategy ζ_n plays like τ_n while in W_n (which can happen indefinitely). If the strategy τ_n uses an exit transition that cannot be matched in $\mathcal{G}_n$ then there must be a corresponding transition in $\mathcal{G}_n$ that either increments the counter and stays in W_n, or increments the counter and moves to W_{n-1} (otherwise it would contradict the fact that τ_n is winning in $\mathcal{W}_n$). In the former case, we continue playing like τ_n. In the latter case, we switch to using the strategy ζ_{n-1} while we are in $W_0 \cup \cdots \cup W_{n-1}$. If there is an exit transition that corresponds to a move to W_n, then we return to using τ_n, and continue as before.

Now consider an infinite play π consistent with ζ_n that starts in W_n. Either π eventually stabilizes in $W_0 \cup \cdots \cup W_{n-1}$, or π is in W_n infinitely often. If it stabilizes in $W_0 \cup \cdots \cup W_{n-1}$, then it must have incremented the counter at least once before entering W_{n-1}. Once it stabilizes in $W_0 \cup \cdots \cup W_{n-1}$, the resulting play must be winning for O_{n-1} by the inductive hypothesis. Overall, this means π is winning for O_n. Otherwise, π returns infinitely often to W_n. Notice that each return to W_n from W_n must be preceded by an increment. Likewise, each return that goes via some W_m for $m < n$, must have incremented the counter at least once when descending from W_n to W_{n-1}. Hence, the counter value is ∞ and the play is winning in $\mathcal{G}_n$.

Although this is not necessary for our purposes, we note that the positional strategy in this case preserves the value of the game exactly. That is, if Eve can win with value n, then she has a positional strategy that can win with value n.

B. Reduction to trees with finite branching

Recall that in Section III, step (2) of the conversion process claimed that we could reduce to trees with some fixed finite branching. We now prove this claim, formally stated here.

Proposition 33. *Let $\mathcal{A}$ be a $\overline{\text{dist} \wedge \text{parity}}$ automaton. Then $[\![\mathcal{A}]\!]$ is bounded over trees with arbitrary branching iff $[\![\mathcal{A}]\!]$ is bounded over trees with branching degree bounded by the number of states of $\mathcal{A}$.*

Proof: Let Q be the state set for $\mathcal{A}$. Suppose for the sake of contradiction that $[\![\mathcal{A}]\!]$ is bounded over trees with finite branching at most $|Q|$, but unbounded over all trees.

Since $[\![\mathcal{A}]\!]$ is unbounded and $\overline{\text{dist} \wedge \text{parity}}$ games have positional strategies by Proposition 13, this means that for every n, there is a tree t_n such that Eve has a positional strategy of value at least n in $\mathcal{G}(\mathcal{A}, t)$. Such a strategy induces a finite set of relevant nodes that the strategy actually uses for each $v \in \text{dom}(t_n)$. Note that the positional strategy from v could send state q to many different neighbors w of v, with possibly different output actions. However, we could always send all of these copies to the position w' that the strategy used for the worst possible action associated with q. Hence, for each

$v \in \mathrm{dom}(t_n)$, the set of relevant nodes can be bounded by $|Q|$.

For each n, construct a new tree t'_n, obtained from t_n by starting at the root, and keeping only relevant children. This new tree t'_n has finite branching degree at most $|Q|$, and the original positional strategy witnesses that $[\![\mathcal{A}]\!](t'_n) \geq n$.

Hence, $(t'_n)_{n\in\mathbb{N}}$ is a family of trees with finite branching witnessing unboundedness of $[\![\mathcal{A}]\!]$. ∎

As stated in step (2) in Section III, this means we can use more traditional automata that operate on trees with some fixed, finite branching degree.

Proposition 34. *Let $\mathcal{A}$ be a $\overline{\textit{dist} \wedge \textit{parity}}$ automaton using directions $\{0, \updownarrow\}$ and m states. Then there is a $\overline{\textit{dist} \wedge \textit{parity}}$ automaton $\mathcal{A}'$ using directions $\{-1, 0, \ldots, m\}$ such that $[\![\mathcal{A}]\!]$ is bounded over all trees iff $[\![\mathcal{A}']\!]$ is bounded over all trees with branching degree m. The size of $\mathcal{A}'$ is polynomial in the size of $\mathcal{A}$. The number of states, counters, and priorities is the same in $\mathcal{A}$ and $\mathcal{A}'$ (only the transition function changes).*

Note that we can also enforce at this stage that the trees that we operate on with $\mathcal{A}'$ are full m-ary trees, i.e. every node has precisely m children. This can be handled in the usual way, by introducing some special symbol to label parts of the tree that would otherwise be missing.

C. Proof of Lemma 15 (2-way universal $\overline{\textit{dist} \wedge \textit{parity}}$ to 1-way alternating $\overline{\textit{dist} \wedge \textit{parity}}$ automata on annotated trees)

Let $\mathcal{U}$ be the 2-way $\overline{\mathrm{dist} \wedge \mathrm{parity}}$ universal automaton $\mathcal{U}$ described in the body that runs on (t, ζ) and computes the value of ζ, where (t, ζ) is a strategy annotation of t based on a positional strategy ζ in $\mathcal{G}(\mathcal{A}, t)$ for a 2-way alternating $\overline{\mathrm{dist} \wedge \mathrm{parity}}$ automaton $\mathcal{A}$.

We seek to prove:

> We can construct a 1-way alternating $\overline{\mathrm{dist} \wedge \mathrm{parity}}$ tree automaton $\mathcal{B}$ such that $[\![\mathcal{U}]\!] \approx [\![\mathcal{B}]\!]$. The size of $\mathcal{B}$ is at most $|\mathcal{A}|^{f(m')}$ where m' is the number of states and priorities of $\mathcal{A}$, and f is some polynomial function independent of $\mathcal{A}$. Moreover, the number of states in $\mathcal{B}$ is polynomial in m', and the number of priorities and counters is the same as $\mathcal{A}$.

Recall that a *strategy annotation* is a labelling of t such that each node x is annotated with a function ζ_x mapping a state q to the set of tuples $(d, (p, c), r)$ such that it is possible to move from (q, x) to (r, xd) with output (p, c) when using the strategy ζ in the game in $\mathcal{G}(\mathcal{A}, t)$. For technical reasons, we will also assume that the annotation at each node x also includes the functions ζ_{xd} for $d \in \{1, \ldots, m\}$.

Let Act be the set of actions for $\mathcal{U}$; each action is of the form (p, c) where p is a priority from some set P of priorities (for technical reasons in the construction below, the "empty" priority ε is also included in P) and $c \in \{\mathtt{ic}, \varepsilon\}$ is the action on the single distance counter. Recall that with this $\overline{\mathrm{dist} \wedge \mathrm{parity}}$ objective, Eve is trying to ensure that the parity condition is not satisfied or the counter achieves a high value.

In state q at position x, $\mathcal{U}$ is designed such that Adam chooses one of the moves $(d, (p, c), r) \in \zeta_x(q)$. After outputting action (p, c) and moving in direction d, play continues from state r. The automaton is universal because Adam controls every choice (Eve's choices in $\mathcal{A}$ are dictated by ζ). It is of size $|\mathcal{A}|^{f(m')}$ where m' is the number of states and priorities in $\mathcal{A}$ and f is a polynomial function independent of $\mathcal{A}$, since the number of strategy labels it can read is exponential in m'. It has the same number of states, priorities, and counters as $\mathcal{A}$, however.

We first convert this 2-way universal automaton $\mathcal{U}$ into a 2-way alternating automaton $\mathcal{B}_{0\downarrow}$ with no upward moves but still using stationary moves and downward moves (directions $\{0, 1, \ldots, m\}$). The price we pay to eliminate the upward movement is that (i) we give some control back to Eve so the automaton is no longer universal, and (ii) we increase the size of the state set by a polynomial factor. The state set for $\mathcal{B}_{0\downarrow}$ is $Q \cup (Q \times \mathrm{Act} \times Q)$. There are two modes of the automaton, *normal mode* and *challenge mode*; states of the form $q \in Q$ correspond to normal mode, and states of the form $(q, c, q') \in Q \times \mathrm{Act} \times Q$ correspond to challenge mode.

In the normal mode, Adam can choose a stationary or downward move consistent with the annotated strategy ζ. Adam is not allowed to choose an upward move. However, he can make a request to stay in the same position, output some action, and (possibly) change state. Formally, a *request* is a triple $(q, c, q') \in Q \times \mathrm{Act} \times Q$. Intuitively, a request when the automaton is in state q in position x in the tree represents an assertion by Adam that there is some play (consistent with ζ) which would eventually return to x in state q', after performing global action c. The *global action* from a sequence of moves

$$(d_1, (p_1, c_1), q_2)(d_2, (p_2, c_2), q_3) \ldots (d_k, (p_k, c_k), q_{k+1})$$

is $(\max\{p_1, \ldots, p_k\}, \max\{c_1, \ldots, c_k\})$ where $\varepsilon < \mathtt{ic} < \bot$ and $\varepsilon < p$ for all $p \in P$. Likewise, given some sequence of actions $(p_1, c_1), (p_2, c_2), \ldots, (p_k, c_k)$ we will write $(p_1, c_1)(p_2, c_2) \ldots (p_k, c_k)$ for the corresponding global action $(\max\{p_1, \ldots, p_k\}, \max\{c_1, \ldots, c_k\})$. We write $(p, c) < (p', c')$ if $p < p'$, or $p = p'$ and $c < c'$.

When Adam makes a request (q, c, q') like this, Eve has a choice: she can either grant him the request by outputting c, moving to state q', and remaining in normal mode, or she can challenge the request by outputting $(p^{\mathrm{odd}}_{\max}, \varepsilon)$, and moving to state (q, c, q'), where $p^{\mathrm{odd}}_{\max}$ is the maximum odd priority used by $\mathcal{U}$.

In challenge mode in some state (q, c, q'), Adam tries to pick out a play that is consistent with his request (i.e. witness the loop that would lead from q to q' with global action c). He has three possibilities for moves, which we denote by (A1)–(A3).

Adam can move from (q, c, q') to a state (r, c', r') with output o if one of the following conditions holds:

(A1) there is a downward move $(d, b, r) \in \zeta_x(q)$ for $d \in \{1, \ldots, m\}$, and an upward move $(-1, b', q') \in \zeta_{xd}(r')$ such that $bc'b' = c$ and $o = (p^{\mathrm{odd}}_{\max}, \varepsilon)bb'$;

(A2) there is a stationary move $(0, b, r) \in \zeta_x(q)$ such that $bc' = c$ and $o = (p^{\mathrm{odd}}_{\max}, \varepsilon)b$;

where $p^{\text{odd}}_{\max}$ is the maximum odd priority used by $\mathcal{U}$. Note that Adam is not allowed to choose an upward move: the idea is that if he had needed an upward move, he should have made an earlier request to handle this.

The last option (A3) is that Adam can choose to request (q, a_1, q'') and (q'', a_2, q') with $a_1 a_2 = c$. In this case Eve selects which part to challenge. If she selects the first part, then the automaton outputs $(p^{\text{odd}}_{\max}, \varepsilon)a_2$ and moves to state (q, a_1, q''). Similarly, if she selects the second part, then the automaton outputs $(p^{\text{odd}}_{\max}, \varepsilon)a_1$ and moves to state (q'', a_2, q').

If Adam reaches a state $(q, (\varepsilon, \varepsilon), q)$, then the automaton moves into some sink state with even priority and no counter actions (we have omitted this from the definition of the state set for presentation purposes only). Otherwise, if he forever remains in challenge mode, then the play does not satisfy the parity condition, since $p^{\text{odd}}_{\max}$ is always output during the challenge mode. This ensures that Adam loses when he fails to witness the requested loop that was challenged by Eve.

Now that we have described the operation of $\mathcal{B}_{0\downarrow}$, we must prove that it correctly computes the value of ζ.

Lemma 35. *If $[\![\mathcal{U}]\!](t, \zeta) \leq n$, then $[\![\mathcal{B}_{0\downarrow}]\!](t, \zeta) \leq n$.*

Proof sketch: If $[\![\mathcal{U}]\!](t, \zeta) \leq n$, then there is a play π consistent with ζ that has value at most n.

The idea is to guide Adam's choice of moves in $\mathcal{B}_{0\downarrow}$ using π.

If part of the play corresponds to a loop, then Adam makes a request based on this, faithfully stating the global action of this loop. Otherwise, Adam chooses the move exactly corresponding to the move in π.

If Eve never challenges a request, then we know that the output from the play is at most value n (since in general we copy all of the actions faithfully, and any counter action from a request is at most the value of the loop itself).

If Eve does challenge a request, then Adam can continue faithfully playing according to π. Because he only makes requests on actual (finite) loops in π, once the play enters challenge mode, he will only make a finite number of additional requests, and will eventually close the loop. If a loop has actions $c_1 c_2 \dots c_k$, then (for distance actions) changing the order of these actions does not affect the value, so $c_1 c_k c_2 c_{k-1} \dots$ has the same value as $c_1 c_2 \dots c_k$, and the global action from, say, $c_1 c_k$ is at most the actual value of $c_1 c_k$ Hence, the value must still be bounded by n. ∎

Lemma 36. *If $[\![\mathcal{U}]\!](t, \zeta) > n \cdot 2^n$, then $[\![\mathcal{B}_{0\downarrow}]\!](t, \zeta) > n$.*

Proof: Assume $[\![\mathcal{U}]\!](t, \zeta) > n \cdot 2^n$.

We describe an n-winning strategy $\zeta_{\mathcal{B}_{0\downarrow}}$ for Eve in $\mathcal{G}(\mathcal{B}_{0\downarrow}, (t, \zeta))$. This strategy must specify when she should challenge requests that Adam makes.

Assume the automaton is in normal mode and Adam makes a $(q, (p, c), q')$ request at position x in t, when the value of the current play in $\mathcal{B}_{0\downarrow}$ is m. Let C (respectively, M) denote the minimum global action (respectively, actual value) over of all loops consistent with ζ starting in q in x, ending in q' in x, and with maximum priority p (and let $C = \perp$ and $M = \infty$ if there are no such loops). Define $\zeta_{\mathcal{B}_{0\downarrow}}$ such that Eve challenges $(q, (p, c), q')$ if $C > c$, or $M > 2^{n-m}$.

Likewise, assume the automaton is in challenge mode and Adam makes a request $req_1 = (q, (p_1, c_1), q'')$ and $req_2 = (q'', (p_2, c_2), q')$ at position x in t when the value of the current play in $\mathcal{B}_{0\downarrow}$ is m. Let C_i (respectively, M_i) denote the minimum global action (respectively, value) over of all loops consistent with ζ and the starting and ending states in req_i and with maximum priority p_i (let $C_i = \perp$ and $M_i = \infty$ if there are no such loops). Define $\zeta_{\mathcal{B}_{0\downarrow}}$ such that Eve challenges req_i if $C_i > c_i$ (if both $C_1 > c_1$ and $C_2 > c_2$, then we choose arbitrarily that Eve challenges req_1). Likewise, if $C_i \leq c_i$ for $i \in \{1, 2\}$, then Eve challenges req_1 if $M_1 \geq M_2$ and challenges req_2 otherwise.

We first prove the following claim.

Claim 37. *Let π be a finite (partial) play consistent with $\zeta_{\mathcal{B}_{0\downarrow}}$ when the initial value is $m \leq n$ and is in position x in t. If π starts in state (q, a, q') at x, ends in state $(r, (\varepsilon, \varepsilon), r)$ at x, and has value at most $n - m$, then there is a play in $\mathcal{A}$ from q at x to q' at x with global action a and value at most 2^{n-m}.*

Proof of claim: We proceed by induction on the length of π.

If the length is 1, then $\pi = (q, (\varepsilon, \varepsilon), q)$, and the result trivially holds.

If the length is greater than 1, then we must consider the different cases depending on the first move.

- If the first move is from (A1) or (A2) and uses no increments, then the result follows by the inductive hypothesis.
- If the first move is from (A1) and uses an increment, then we can have at most two actual increments (on the upper and downward part). By induction, there is a 2^{n-m-1} play from the remaining part. Overall, this means there is a play of value $2 + 2^{n-m-1} < 2^{n-m}$.
- If the first move is from (A2) and uses an increment, then similar reasoning shows that the value is at most $1 + 2^{n-m-1} < 2^{n-m}$.
- If the first move is a request for (q, a_1, q'') and (q'', a_2, q') from (A3) and both a_1 and a_2 use `ic`, and (q, a_1, q'') is selected by Eve, then by the inductive hypothesis, there is a (q, a_1, q'') loop of value at most 2^{n-m-1}. But by construction of Eve's strategy, this was only chosen if this part had the higher value. Hence, there is also a (q'', a_2, q') loop of value at most 2^{n-m-1}. These can be combined to form a (q, a, q') play of value at most 2^{n-m}.
- If the first move is a request for (q, a_1, q'') and (q'', a_2, q_1) from (A3) with `ic` in a_1 but not in a_2, then it must be (q, a_1, q'') that is challenged. But that means we can apply the inductive hypothesis to get a loop of value at most 2^{n-m}, joined together with a loop of value 0.

∎

Consider a play in $\mathcal{G}(\mathcal{B}_{0\downarrow}, (t, \zeta))$ consistent with $\zeta_{\mathcal{B}_{0\downarrow}}$. Suppose for the sake of contradiction that it has value at most n.

Assume that it never enters challenge mode. Then there can be at most n requests using $\texttt{ic}$. Since all of these request went unchallenged, by construction of Eve's strategy, this means that each request can be expanded into a partial play of value at most 2^{n-m} (where m is the value of the play from $\zeta_{\mathcal{B}_{0\downarrow}}$ before the request). All of the other ε-requests can be expanded into loops with no increments (otherwise, they would have been challenged). Hence, we can expand this into a play in $\mathcal{G}(\mathcal{U}, (t, \zeta))$ of value at most

$$\Sigma_{m=0}^{n-1} 2^{n-m} \leq n \cdot 2^n.$$

This contradicts the fact that $[\![\mathcal{U}]\!](t, \zeta) > n \cdot 2^n$.

Now assume that the play enters challenge mode. Since the play has finite value, it must reach $(q, (\varepsilon, \varepsilon), q)$ (since the priority output during the challenge mode is always odd). Consider the suffix of this play during which the automaton is in challenge mode; assume that the value of the prefix leading up to this moment is m. By the claim, there is a loop corresponding to the request that is consistent with ζ and of value at most 2^{n-m}. But this contradicts the fact that Eve only challenges if every loop consistent with her strategy is greater than 2^{n-m}.

This completes the proof of Lemma 36. ∎

Hence, with the help of the previous two lemmas, we have shown the following lemma.

Lemma 38. *We can construct a 2-way alternating automaton $\mathcal{B}_{0\downarrow}$ using directions $\{0, 1, \ldots, m\}$ such that $[\![\mathcal{B}_{0\downarrow}]\!] \approx [\![\mathcal{U}]\!]$. The size of $[\![\mathcal{B}_{0\downarrow}]\!]$ is at most $|\mathcal{A}|^{f(m')}$ where m' is the number of states and priorities in $\mathcal{A}$, and f is some polynomial function independent of $\mathcal{A}$. The number of states in $\mathcal{B}_{0\downarrow}$ is polynomial in m' and the number priorities and counters is the same as in $\mathcal{A}$.*

Observe that the constructed automata still uses stationary moves. These can be eliminated to yield the desired 1-way alternating $\overline{\text{dist} \wedge \text{parity}}$ automaton $\mathcal{B}$.

Lemma 39. *We can construct a 1-way alternating $\overline{\text{dist} \wedge \text{parity}}$ automaton $\mathcal{B}$ using directions $\{1, \ldots, m\}$ such that $[\![\mathcal{B}_{0\downarrow}]\!] \approx [\![\mathcal{B}]\!]$. The size of $[\![\mathcal{B}_{0\downarrow}]\!]$ is at most $|\mathcal{A}|^{f(m')}$ where m' is the number of states and priorities in $\mathcal{A}$, and f is some polynomial function independent of $\mathcal{A}$. The number of states in $\mathcal{B}$ is polynomial in m' and the number priorities and counters is the same as in $\mathcal{A}$.*

Proof sketch: For each state q, letter a, and goal set G of downward moves, we consider a *local game* $\mathcal{G}(q, a, G)$ describing the stationary moves that are possible before moving downwards. Downward moves are terminal positions in the game: they are winning if they are in the goal set G, and losing otherwise. This is a $\overline{\text{dist} \wedge \text{parity}}$ game. It can be shown that it is decidable whether Eve wins such a game. The idea is to transform this $\overline{\text{dist} \wedge \text{parity}}$ game into a game with an ω-regular winning condition. The Rabin winning condition expresses

- the parity condition is not satisfied or
- the counter is incremented infinitely often.

This can be done using $l+1$ Rabin pairs, where l is the number of priorities in $\mathcal{B}_{0\downarrow}$. We emphasize that the counter actions are now evaluated according to this ω-regular condition, so the resulting game is no longer a cost game, it is a finite Rabin game which can be solved using [26].

We can construct and solve all of these finite Rabin games in time $|\mathcal{B}_{0\downarrow}|^{f'(l)}$ where f' is a polynomial function independent of $\mathcal{B}_{0\downarrow}$ and l is the number of priorities used by $\mathcal{B}_{0\downarrow}$.

The desired 1-way automaton $\mathcal{B}$ has the same input alphabet, set of states, set of counters, and initial state as $\mathcal{B}_{0\downarrow}$. For a state q and input letter a, its transition function is defined as follows: Eve chooses a goal set G such that she wins in $\mathcal{G}(q, a, G)$; then Adam chooses any transition $(d, c, q') \in G$ and performs it. Hence, we can ensure that the size and time complexity of constructing this $\mathcal{B}$ is at most $|\mathcal{A}|^{f(m')}$ where m' is the number of states and priorities of $\mathcal{A}$. ∎

This completes the proof of Lemma 15.

D. Proof sketch of special case of Lemma 16 (Alternating to nondeterministic automata)

We provide more details for one case of Lemma 16. Essentially, it is the construction in [21, Section 4.3.2], however, we use slightly different notation here, and emphasize where we are using the results about cost automata on words that impact the complexity.

> Let $\mathcal{B}$ be a 1-way $\overline{\text{dist} \wedge \text{parity}}$ automaton $\mathcal{B}$. If $\mathcal{B}$ uses only priorities $\{0, 1\}$, then there is a 1-way nondeterministic S $\wedge$ parity automaton $\mathcal{S}$ of elementary size such that $[\![\mathcal{S}]\!] \approx [\![\mathcal{B}]\!]$. Moreover, assuming (EXP-Dual-Finite), $\mathcal{S}$ has at most $|\mathcal{B}|^{f(m)}$ states, m counters, and priorities $\{1, 2\}$ where m is the number of states of $\mathcal{B}$ and f is a polynomial function independent of $\mathcal{B}$.

Fix some n-winning strategy ζ for Eve in $\mathcal{G}(\mathcal{B}, t)$. We can view this strategy as a finitely branching strategy tree, where each branch corresponds to a different play that is possible using this strategy.

We can "slice" this tree into infinitely many sections, where each slice must witness priority 1, or the partial play leading to the slice must have witnessed value n from the counters. Indeed, if this were not possible, then König's lemma would imply that there is a play consistent with ζ with only finitely many 1 and counter value less than n, contradicting the fact that the strategy is n-winning. Formally, the slices are an infinite set of strictly increasing frontiers of the strategy tree. These slices are sometimes referred to as "breakpoints".

We now consider t annotated with a positional strategy ζ and a set E of infinitely many slices, denoted (t, ζ, E).

Let u be a finite word describing the annotations along a prefix of a given branch in (t, ζ, E). There is a nondeterministic B-automaton on finite words that guesses a partial play described by u, and assigns value ∞ if there is priority 1 in every slice, and otherwise outputs the counter value up to and including the first slice that does not have priority 1. Hence, by (Dual-Finite) (respectively, (EXP-Dual-Finite)) there is an equivalent history-deterministic S-automaton $\mathcal{H}$

of elementary size (respectively, exponential size, with only $m = |Q_{\mathcal{B}}|$ counters).

For an infinite word w describing the annotations along a given branch in (t, ζ, E), the value of the plays described by w is equal to the supremum of the values assigned by $[\![\mathcal{H}]\!]$ on each of the finite prefixes u of w. Hence, by viewing $\mathcal{H}$ as an S $\wedge$ parity automaton (with accepting states assigned priority 2, and everything else assigned priority 1), $\mathcal{H}$ is an S $\wedge$ parity automaton using priorities $\{1, 2\}$ computing the value of the plays on w.

Overall, the desired S $\wedge$ parity automaton $\mathcal{S}$ in Lemma 16 operates as follows: on input t, (i) Eve selects a positional strategy ζ in $\mathcal{G}(\mathcal{B}, t)$, (ii) Eve guesses a set E of infinitely many slices, and (iii) $\mathcal{H}$ is simulated on every branch of (t, ζ, E).

APPENDIX D
MISSING PROOFS FROM SECTION VI

A. Proof of Theorem 20 (lower bounds for boundedness)

Recall the statement:

Boundedness is 2EXPTIME-hard for answer-guarded GF and also for Monadic Datalog (hence for GN-Datalog). It is EXPTIME-hard for fixed-width answer-guarded GF without equality.

The 2EXPTIME lower bound for boundedness of answer-guarded GF follows from the 2EXPTIME lower bound for satisfiability of GF sentences (similarly for the EXPTIME lower bound for fixed width GF) [30]. This general technique is described in [5, Observation 8.2]. Let $\psi(x, X)$ be an answer-guarded GF formula that is unbounded, such as $\psi(x, X) := Sx \vee \exists y.(Rxy \wedge Xy)$ where R is a fresh binary relation and S is a fresh unary relation. Then we claim that a GF sentence ϕ is unsatisfiable iff $\psi(x, X)^{U'} \wedge \phi^U$ is bounded, where χ^U is the result of relativizing the quantification in χ to some fresh unary predicate U. If ϕ is unsatisfiable then $\psi(x, X)^{U'} \wedge \phi^U$ is unsatisfiable, so the closure ordinal of $\psi(x, X)^{U'} \wedge \phi^U$ is trivially 0, and the formula is bounded. If ϕ is satisfiable, then $\psi(x, X)^{U'} \wedge \phi^U$ is unbounded due to the $\psi(x, X)^{U'}$ conjunct.

A 2EXPTIME lower bound for boundedness of GN-Datalog can be done via a similar reduction from satisfiability. For the lower bound for GN-Datalog, observe that any GNF sentence ϕ in weak GN-normal form can be expressed as a GN-datalog program $\Pi = \langle \Pi_1, \ldots, \Pi_j \rangle$ with some 0-ary IDB predicate Z_ϕ in Π_j such that $\mathfrak{A} \models \phi$ iff $\Pi^\infty(\mathfrak{A}) \models Z_\phi$. The size of the program is polynomial in the size of ϕ, and uses the stratified negation to get the arbitrary nesting of UCQ-shaped subformulas in ϕ. Let Z_ϕ be the 0-ary IDB in Π_j, corresponding to ϕ itself. Let S, R be fresh EDB predicates, and let Y be a fresh IDB predicate. Then ϕ is unsatisfiable iff $\Pi' = \langle \Pi_1, \ldots, \Pi_j, \Pi_{j+1} \rangle$ is unbounded, where Π_{j+1} has rules

$$Yy \leftarrow \big((Sy \wedge Z_\phi)\big) \quad \text{and} \qquad Yy \leftarrow \big((Yx \wedge Rxy)\big).$$

Hence, the 2EXPTIME lower bounded for boundedness of GN-Datalog follows from the 2EXPTIME lower bound for GNF satisfiability (see Theorem 2).

To show that boundedness is hard for Monadic Datalog, we need a new idea, since satisfiability of Monadic Datalog is trivial due to the absence of negation. We use a reduction from the containment problem instead:

Claim 40. *There is a many-one reduction from the problem of containment of a Datalog query in a union of conjunctive queries to the Boundedness problem for Datalog, with the redution taking containment problems involving boolean Monadic Datalog to Monadic Datalog boundedness problems.*

Proof: Given boolean Datalog query Q_D with goal predicate $Goal_D()$, and boolean UCQ Q_U over schema S, consider a schema which adds to the extensional predicates of S an

additional binary relation R and an additional unary relation U.

Consider the Datalog program with goal predicate $Goal(y)$ formed by adding to the rules of Q_D the following rules:

$$\begin{aligned} Goal(y) &\leftarrow Goal(x) \wedge R(x,y) \\ Goal(y) &\leftarrow Goal_D() \wedge U(y) \\ Goal(y) &\leftarrow Q_U() \wedge R(z,y) \\ Goal(y) &\leftarrow Q_U() \wedge R(y,z) \\ Goal(y) &\leftarrow Q_U() \wedge U(y) \end{aligned}$$

Here $Q_U()$ abbreviates rules that would capture Q_U in Datalog.

The last three rules guarantee that when Q_U holds, every element that is in the domain of R or U is returned. Since the first two rules only return elements in the domain of U or of R, this means that whenever Q_U holds, exactly the elements in the domain of U or R are returned.

Suppose Q_D is contained in Q_U. Then by the above, we have that the query is exactly the result of the last three rules, which is bounded.

Suppose Q_D is not contained in Q_U, with instance I for schema S having Q_D holding Q_U failing.

Then considering all expansions of I to have R and U, we see that the Datalog program is equivalent to the one in the first two rules, which is easily seen to be unbounded. ■

Combining the reduction above with the 2EXPTIME lower bound for Monadic Datalog containment in UCQs from [27] we obtain:

Corollary 41. *Boundedness for Monadic Datalog is* 2EXPTIME*-hard.*