Characterizing Definability in Decidable Fixpoint Logics*

Michael Benedikt[1], Pierre Bourhis[2], and Michael Vanden Boom[3]

1 **Department of Computer Science, University of Oxford, Oxford, UK**
2 **CNRS CRIStAL UMR 9189, INRIA Lille, Lille, FR**
3 **Department of Computer Science, University of Oxford, Oxford, UK**

Abstract

We look at characterizing which formulas are expressible in rich decidable logics such as guarded fixpoint logic, unary negation fixpoint logic, and guarded negation fixpoint logic. We consider semantic characterizations of definability, as well as effective characterizations. Our algorithms revolve around a finer analysis of the tree-model property and a refinement of the method of moving back-and-forth between relational logics and logics over trees.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Guarded logics, bisimulation, definability, automata

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.107

1 Introduction

A major line of research in computational logic has focused on obtaining extremely expressive decidable logics. The guarded fragment (GF) [1], the unary negation fragment (UNF) [23], and the guarded negation fragment (GNF) [3] are rich decidable fragments of first-order logic. Each of these has extensions with a fixpoint operator that retain decidability: GFP [18], UNFP [23], and GNFP [3] respectively. In each case the argument relies on "moving to trees". This involves showing that the logic possesses the tree-like model property: whenever there is a satisfying model for a formula, it can be taken to be of tree-width that can be effectively computed from the formula. Such models can be coded by trees, thus reducing satisfiability of the logic to satisfiability of a corresponding formula over trees, which can be decided using automata-theoretic techniques. This method has been applied for decades (e.g. [25, 16]).

A question is how to recognize formulas in these logics, and more generally how to distinguish the properties of the formulas in one logic from another. Clearly if we start with a formula in an undecidable logic, such as first-order logic or least fixed point logic (LFP), we have no possibility for effectively recognizing any non-trivial property. But we could still hope for an insightful semantic characterization of the subset that falls within the decidable logic. One well-known example of this is van Benthem's theorem [24] characterizing modal logic within first-order logic – a first-order sentence is equivalent to a modal logic sentence exactly when it is bisimulation-invariant. For fixpoint logics, an analogous characterization is the Janin-Walukiewicz theorem [20], stating that the modal mu-calculus (L_μ) captures the bisimulation-invariant fragment of monadic second-order logic (MSO). If we start in one decidable logic and look to characterize another decidable logic, we could hope for a

* Benedikt and Vanden Boom were funded by the EPSRC grants PDQ (EP/M005852/1), ED^3 (EP/N014359/1), and DBOnto (EP/L012138/1). Bourhis was funded by the DeLTA project (ANR-16-CE40-0007).

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 107; pp. 107:1–107:14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

characterization that is effective. For example, Otto [22] showed that if we start with a formula of L_μ, we can determine whether it can be expressed in modal logic.

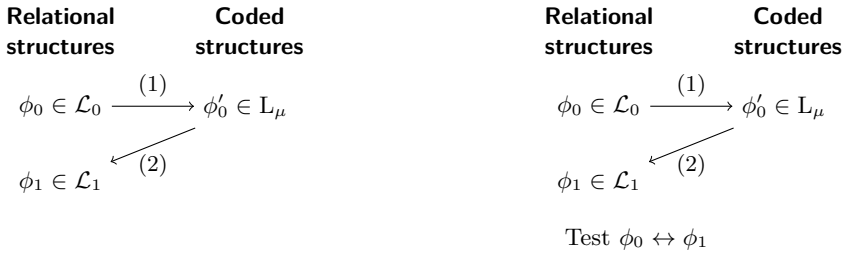
In this work we will investigate both semantic and effective characterizations. We will begin with GFP. Grädel, Hirsch, and Otto [16] have already provided a characterization of GFP-definability within a very rich logic extending MSO called guarded second-order logic (GSO). The characterization is exactly analogous to the van Benthem and Janin-Walukiewicz results mentioned above: GFP captures the "guarded bisimulation-invariant" fragment of GSO. The characterization makes use of a refinement of the method used for decidability of these logics, which moves *back-and-forth between relational structures and trees*: (1) define a *forward mapping* taking a formula ϕ_0 in the larger logic (e.g. GSO invariant under guarded bisimulation) over relational structures to a formula ϕ'_0 over trees that describes codes of structures satisfying ϕ_0; (2) define a *backward mapping* based on the invariance going back to some ϕ_1 in the restricted logic (e.g. GFP). The method is shown in Figure 1a.

Our first main theorem is an effective version of the above result: if we start with a formula in certain richer decidable fixpoint logics, such as GNFP, we can decide whether the formula is in GFP. At the same time we provide a refinement of [16] which accounts for two signatures, the one allowed for arbitrary relations and the one allowed for "guard relations" that play a key role in the syntax of all guarded logics. We extend this result to deciding membership in the "k-width fragment", GNFP^k; roughly speaking this consists of formulas built up from guarded components and positive existential formulas with at most k variables. We provide a semantic characterization of this fragment within GSO, as the fragment closed under the corresponding notion of bisimulation (essentially, the GN^k-bisimulation of [3]). As with GFP, we show that the characterization can be made effective, provided that one starts with a formula in certain larger decidable logics. The proof also gives an effective characterization for the k-width fragment of UNFP.

As in the method for invariance and decidability above, we apply a forward mapping to move from a formula ϕ_0 in a larger logic $\mathcal{L}_0$ on relational structures to a formula ϕ'_0 on tree encodings. But then we can apply a *different backward mapping*, tuned towards the smaller logic $\mathcal{L}_1$ and the special properties of its tree-like models. The backward mapping of a tree property ϕ'_0 is always a formula ϕ_1 in the smaller logic $\mathcal{L}_1$ (e.g. GFP). But it is no longer guaranteed to be "correct" unconditionally – i.e. to always characterize structures whose codes satisfy ϕ'_0. Still, we show that *if* the original formula ϕ_0 is definable in the smaller logic $\mathcal{L}_1$, then the backward mapping applied to the forward mapping gives such a definition. Since we can check equivalence of two sentences in these logics effectively, this property suffices to get decidability of definability. The revised method is shown schematically in Figure 1b.

The technique above has a few inefficiencies; first, the general forward mapping passes through MSO and has non-elementary complexity. Secondly, the technique implicitly moves between relational structures and trees *twice*: once to construct ϕ_0, and a second time to check that ϕ_0 is equivalent to ϕ_1, which in turn requires first forming a formula ϕ'_1 over trees via a forward mapping and then checking its equivalence with ϕ'_0. We show that in some cases we can optimize this, allowing us to get tight bounds on the equivalence problem.

We show that our results "restrict" to fragments of these guarded logics, including their first-order fragments. In particular, our results give effective characterizations of GF definability when the input is in FO. They can be thus seen as a generalization of well-known effective characterizations of the conjunctive existential formulas in GF, the *acyclic queries*. We show that we can apply our techniques to the problem of transforming conjunctive formulas to a well-known efficiently-evaluable form (acyclic formulas) relative to GF theories. These results complement previous results on query evaluation with constraints from [6, 13].



(a) Back-and-forth for Semantic Characterization.

(b) Back-and-forth for Effective Characterization.

Figure 1

This refined back-and-forth method can be tuned in a number of ways, allowing us to control the signature as well as the sublogic. We show this can be adapted to give an approximation of the formula ϕ_0 within the logic $\mathcal{L}_1$, which is a kind of *uniform interpolant*.

2 Preliminaries

We work with finite relational signatures σ. 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 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 in any subformula of ϕ, and the width of a signature σ is the maximum arity of its relations.

The *Guarded Negation Fragment* of FO [3] (denoted GNF) is built up inductively according to the grammar $\phi ::= R\,\boldsymbol{x} \mid \exists x.\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \alpha(\boldsymbol{x}) \wedge \neg\phi(\boldsymbol{x})$ where R is either a relation symbol or the equality relation, and α is an atomic formula or equality such that $\mathrm{free}(\alpha) \supseteq \mathrm{free}(\phi)$. Such an α is a *guard*. If we restrict α to be an equality, then each negated formula can be rewritten to use at most one free variable; this is the *Unary Negation Fragment*, UNF [23]. GNF is also related to the *Guarded Fragment* [1] (GF), typically defined via the grammar $\phi ::= R\,\boldsymbol{x} \mid \exists \boldsymbol{x}.\alpha(\boldsymbol{xy}) \wedge \phi(\boldsymbol{xy}) \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi(\boldsymbol{x})$ where R is either a relation symbol or the equality relation, and α is an atomic formula or equality that uses all of the free variables of ϕ. Here it is the quantification that is guarded, rather than negation. GNF subsumes GF sentences and UNF formulas.

The fixpoint extensions of these logics (denoted GNFP, UNFP, and GFP) extend the base logic with formulas $[\mathbf{lfp}_{X,\boldsymbol{x}}.\alpha(\boldsymbol{x}) \wedge \phi(\boldsymbol{x}, X, \boldsymbol{Y})](\boldsymbol{x})$ where (i) $\alpha(\boldsymbol{x})$ is an atomic formula or equality guarding $\boldsymbol{x}$, (ii) X only appears positively in ϕ, (iii) second-order variables like X cannot be used as guards. Some alternative (but equi-expressive) ways to define the fixpoint extension are discussed in [3]; in all of the definitions, the important feature is that tuples in the fixpoint are guarded by an atom in the original signature. In UNFP, there is an additional requirement that only unary or 0-ary predicates can be defined using the fixpoint operators. GNFP subsumes both GFP sentences and UNFP formulas. These logics are all contained in LFP, the fixpoint extension of FO, so the semantics are inherited from there.

It is often helpful to consider the formulas in a normal form. *Strict normal form* GNFP

formulas can be generated using the following grammar:

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

where α is an atomic formula or equality statement such that $\text{free}(\alpha) = \text{free}(\phi)$; we call such an α a *strict guard*. Every GNFP-formula can be converted into this form in a canonical way with an exponential blow-up in size. We denote by GNFP^k the set of GNFP-formulas that are of width k when they are brought into this normal form. For convenience in proofs, we are using a slightly different normal form than previous papers on these logics.

These guarded fixpoint logics are expressive: the μ-calculus is contained in each of these fixpoint logics, and every positive existential formula is expressible in UNFP and GNFP (and even UNF and GNF). Nevertheless, these logics are decidable and have nice model theoretic properties. In particular satisfiability and finite satisfiability is 2-ExpTime-complete for GNF and GNFP [5]. The same holds for UNFP and GFP [23, 18]. GNFP (and hence UNFP and GFP) has the tree-like model property [5]: if ϕ is satisfiable, then ϕ is satisfiable over structures of bounded tree-width. In fact satisfiable GNFP^k formulas have satisfying structures of tree-width $k-1$. GNF (and hence UNF and GF) has the finite-model property [5]: if ϕ is satisfiable, then ϕ is satisfiable in a finite structure. This does not hold for the fixpoint extensions. In this paper we will be concerned with equivalence over all structures.

In this work we will be interested in varying the signatures considered, and in distinguishing more finely which relations can be used in guards. If we want to emphasize the relational signature σ being used, then we will write, e.g., $\text{GNFP}[\sigma]$. For $\sigma_g \subseteq \sigma$, we let $\text{GNFP}[\sigma, \sigma_g]$ denote the logic built up as in GNFP but allowing only relations $R \in \sigma$ at the atomic step and only guards α using equality or relations $R \in \sigma_g$. We define $\text{GFP}[\sigma, \sigma_g]$ similarly. Note that $\text{UNFP}[\sigma]$ is equivalent to $\text{GNFP}[\sigma, \emptyset]$, since if the only guards are equality guards, then the formula can be rewritten to use only unary negation and monadic fixpoints.

Guarded second-order logic over a signature σ (denoted $\text{GSO}[\sigma]$) is a fragment of second-order logic in which second-order quantification is interpreted only over guarded relations, i.e. over relations where every tuple in the relation is guarded by some predicate from σ. We refer the interested reader to [16] for more background and some equivalent definitions of this logic. The logics UNFP, GNFP, and GFP can all be translated into GSO.

A special kind of signature is a *transition system signature* Σ consisting of a finite set of unary predicates (corresponding to a set of propositions) and binary predicates (corresponding to a set of actions). A structure for such a signature is a *transition system*. Trees allowing both edge-labels and node-labels have a natural interpretation as transition systems. We will be interested in two logics over transition system signatures. One is *monadic second-order logic* (denoted MSO) – where second-order quantification is only over unary relations. MSO is contained in GSO, because unary relations are trivially guarded. While MSO and GSO can be interpreted over arbitrary signatures, there are logics like *modal logic* that have syntax specific to transition system signatures. Another is the *modal μ-calculus* (denoted L_μ), an extension of modal logic with fixpoints. Given a transition system signature Σ, formulas $\phi \in \text{L}_\mu[\Sigma]$ can be generated using the grammar $\phi ::= P \mid X \mid \phi \wedge \phi \mid \neg\phi \mid \langle\rho\rangle\phi \mid \mu X.\phi$ where P is a unary relation in Σ and ρ is a binary relation in Σ. The formulas $\mu X.\phi$ are required to use the variable X only positively in ϕ, and the semantics define a least-fixpoint operation based on ϕ. It is easy to see that L_μ can be translated into MSO.

It is well-known that σ-structures of tree-width $k-1$ can be encoded by labelled trees over an alphabet that depends only on the signature of the structure and k, which we denote $\Sigma^{\text{code}}_{\sigma,k}$. Our encoding scheme will make use of trees with both node and edge labels, i.e. trees over a

transition system signature $\Sigma^{\text{code}}_{\sigma,k}$. Roughly speaking, a node label is a set of unary relations (like $R_{i_1,\ldots,i_n}$) from $\Sigma^{\text{code}}_{\sigma,k}$ that encodes the set of atomic formulas (like $R(a_{i_1},\ldots,a_{i_n})$) that hold of the elements represented at that node, and an edge label ρ is a binary relation from $\Sigma^{\text{code}}_{\sigma,k}$ that indicates the relationship between the names of encoded elements in neighboring nodes. This scheme differs slightly from the one used in [16]. The exact coding conventions are not important for understanding the ideas in the rest of the paper. Given some $\Sigma^{\text{code}}_{\sigma,k}$-tree $\mathcal{T}$, we say $\mathcal{T}$ is *consistent* if it satisfies certain natural conditions that ensure that the tree actually corresponds to a code of some tree decomposition of a σ-structure. A consistent $\Sigma^{\text{code}}_{\sigma,k}$-tree $\mathcal{T}$ can be decoded to an actual σ-structure, denoted $\mathfrak{D}(\mathcal{T})$.

Bisimulation games and unravellings. The logic L_μ over transition system signatures lies within MSO. Similarly the guarded logics GFP, UNFP, and GNFP all lie within GSO and apply to arbitrary-arity signatures. It is easy to see that these containments are proper. In each case, what distinguishes the smaller logic from the larger is *invariance* under certain equivalences called *bisimulations*, each of which is defined by a certain player having a winning strategy in a two-player infinite game played between players Spoiler and Duplicator.

For L_μ, the appropriate game is the classical *bisimulation game* between transition systems $\mathfrak{A}$ and $\mathfrak{B}$. It is straightforward to check that $\mathrm{L}_\mu[\Sigma]$-formulas are Σ-bisimulation invariant, i.e. they cannot distinguish between Σ-bisimilar transition systems. We will make use of a stronger result of Janin and Walukiewicz [20] that the μ-calculus is the bisimulation-invariant fragment of MSO (we state it here for trees because of how we use this later): A class of trees is definable in $\mathrm{L}_\mu[\Sigma]$ iff it is definable in MSO$[\Sigma]$ and closed under Σ-bisimulation within the class of all Σ-trees. Moreover, the translation between these logics is effective.

We now describe a generalization of these games between structures $\mathfrak{A}$ and $\mathfrak{B}$ over a signature σ with arbitrary arity relations, parameterized by some subsignature σ' of the structures. Each position in the game is a partial σ' homomorphism h from $\mathfrak{A}$ to $\mathfrak{B}$, or vice versa. The *active structure* in position h is the structure containing the domain of h. The game starts from the empty partial map from $\mathfrak{A}$ to $\mathfrak{B}$. In each round of the game, Spoiler chooses between one of the following moves:

- *Extend:* Spoiler chooses some set X of elements in the active structure such that $X \supseteq \mathrm{dom}(h)$, and Duplicator must then choose h' extending h (i.e. such that $h(c) = h'(c)$ for all $c \in \mathrm{dom}(h)$) such that h' is a partial σ' homomorphism; Duplicator loses if this is not possible. Otherwise, the game proceeds from the position h'.
- *Switch:* Spoiler chooses to switch active structure. If h is not a partial σ' isomorphism, then Duplicator loses. Otherwise, the game proceeds from the position h^{-1}.
- *Collapse:* Spoiler selects some $X \subseteq \mathrm{dom}(h)$ and the game continues from position $h \restriction_X$.

Duplicator wins if she can continue to play indefinitely.

We will consider several variants of this game. These were essentially known already in the literature (see, e.g., [16, 17, 3]), sometimes with different names or minor technical differences in the definitions. For $k \in \mathbb{N}$ and $\sigma_g \subseteq \sigma'$:

1. **k-width guarded negation bisimulation game**: The $\mathrm{GN}^k[\sigma', \sigma_g]$-game is the version of the game where the domain of every position h is of size at most k, and Spoiler can only make a switch move at h if $\mathrm{dom}(h)$ is strictly σ_g-guarded in the active structure.
2. **block k-width guarded negation bisimulation game**: The $\mathrm{BGN}^k[\sigma', \sigma_g]$-game is like the $\mathrm{GN}^k[\sigma', \sigma_g]$-game, but additionally Spoiler is required to alternate between extend/switch moves and moves where he collapses to a strictly σ_g-guarded set. We call it the "block" game since Spoiler must select all of the new extension elements in a single block, rather than as a series of small extensions. The key property is that the game alternates

between positions with a strictly σ_g-guarded domain, and positions of size at most k. The restriction mimics the alternation between formulas of width at most k and strictly σ_g-guarded formulas within normalized GNFP^k formulas.

3. **guarded bisimulation game**: The $\mathrm{G}[\sigma', \sigma_g]$-game is the version of the game where the domain of every position must be strictly σ_g-guarded in the active structure. Note that in such a game, every position h satisfies $|\mathrm{dom}(h)| \leq \mathrm{width}(\sigma_g)$.

We say $\mathfrak{A}$ and $\mathfrak{B}$ are $\mathrm{GN}^k[\sigma', \sigma_g]$-*bisimilar* if Duplicator has a winning strategy in the $\mathrm{GN}^k[\sigma', \sigma_g]$-game starting from the empty position. We say a sentence ϕ is $\mathrm{GN}^k[\sigma', \sigma_g]$-*invariant* if for any pair of $\mathrm{GN}^k[\sigma', \sigma_g]$-bisimilar σ'-structures, $\mathfrak{A} \models \phi$ iff $\mathfrak{B} \models \phi$. A logic $\mathcal{L}$ is $\mathrm{GN}^k[\sigma', \sigma_g]$-invariant if every sentence in $\mathcal{L}$ is $\mathrm{GN}^k[\sigma', \sigma_g]$-invariant. When the guard signature is the entire signature, we will write, e.g., $\mathrm{GN}^k[\sigma']$ instead of $\mathrm{GN}^k[\sigma', \sigma']$.

It is known that the bisimulation games characterize certain fragments of FO: $\mathrm{GF}[\sigma']$ is the $\mathrm{G}[\sigma']$-invariant fragment of $\mathrm{FO}[\sigma']$ [1] and $\mathrm{GNF}^k[\sigma']$ can be characterized as either the $\mathrm{BGN}^k[\sigma']$-invariant or the $\mathrm{GN}^k[\sigma']$-invariant fragment of $\mathrm{FO}[\sigma']$ (this follows from work in [3] and [7]). Likewise, for fixpoint logics and fragments of GSO, $\mathrm{GFP}[\sigma']$ is the $\mathrm{G}[\sigma']$-invariant fragment of $\mathrm{GSO}[\sigma']$ [16], while $\mathrm{UNFP}^k[\sigma']$ is the $\mathrm{BGN}^k[\sigma', \emptyset]$-invariant fragment of $\mathrm{GSO}[\sigma']$ (this follows from [9]). The survey in [17] also describes some of these invariance results.

In this paper, we will prove a corresponding characterization for $\mathrm{GNFP}^k[\sigma']$ in terms of $\mathrm{BGN}^k[\sigma']$-invariance: $\mathrm{GNFP}^k[\sigma']$ is the $\mathrm{BGN}^k[\sigma']$-invariant fragment of $\mathrm{GSO}[\sigma']$ (see Theorem 16). Note that for fixpoint logics, $\mathrm{GN}^k[\sigma']$-invariance is strictly weaker than $\mathrm{BGN}^k[\sigma']$-invariance, and applies to other decidable logics (e.g. [7]).

Unravellings. Given a σ-structure $\mathfrak{A}$ and $k \in \mathbb{N}$ and $\sigma_g \subseteq \sigma' \subseteq \sigma$, we would like to construct a structure that is $\mathrm{GN}^k[\sigma', \sigma_g]$-bisimilar to $\mathfrak{A}$ but has a tree-decomposition of bounded tree-width. A standard construction achieves this, called the $\mathrm{GN}^k[\sigma', \sigma_g]$-unravelling of $\mathfrak{A}$. Let Π_k be the set of finite sequences of the form $Y_0 Y_1 \dots Y_m$ such that $Y_0 = \emptyset$ and each Y_i is a set of elements from $\mathfrak{A}$ of size at most k. Each such sequence can be seen as the projection to $\mathfrak{A}$ of a play in the $\mathrm{GN}^k[\sigma', \sigma_g]$-bisimulation game between $\mathfrak{A}$ and some other structure. For Y a set of elements from $\mathfrak{A}$, let $\mathrm{AT}_{\mathfrak{A},\sigma'}(Y)$ be the set of atoms that hold of the elements in Y: $\{R(a_1, \dots, a_l) : R \in \sigma', \{a_1, \dots, a_l\} \subseteq Y, \mathfrak{A} \models R(a_1, \dots, a_l)\}$. Now define a $\Sigma^{\mathrm{code}}_{\sigma',k}$-tree $\mathcal{U}_{\mathrm{GN}^k[\sigma',\sigma_g]}(\mathfrak{A})$ where each node corresponds to a sequence in Π_k, and the sequences are arranged in prefix order. Roughly speaking, the node label of every $v = Y_0 \dots Y_{m-1} Y_m$ is an encoding of $\mathrm{AT}_{\mathfrak{A},\sigma'}(Y_m)$, and the edge label between its parent u and v indicates the relationship between the shared elements $Y_{m-1} \cap Y_m$ encoded in u and v. We define $\mathfrak{D}(\mathcal{U}_{\mathrm{GN}^k[\sigma',\sigma_g]}(\mathfrak{A}))$ to be the $\mathrm{GN}^k[\sigma', \sigma_g]$-unravelling of $\mathfrak{A}$. By restricting the set Π_k to reflect the possible moves in the games, we can define unravellings based on the other bisimulation games in a similar fashion. We summarize the two unravellings that will be most relevant:

1. **block k-width guarded negation unravelling**: The $\mathrm{BGN}^k[\sigma', \sigma_g]$-unravelling is denoted $\mathfrak{D}(\mathcal{U}_{\mathrm{BGN}^k[\sigma',\sigma_g]}(\mathfrak{A}))$. Its encoding $\mathcal{U}_{\mathrm{BGN}^k[\sigma',\sigma_g]}(\mathfrak{A})$ is obtained by considering only sequences $Y_0 \dots Y_m \in \Pi_k$ such that for all *even* i, $Y_{i-1} \supseteq Y_i$ and $Y_i \subseteq Y_{i+1}$ and Y_i is strictly σ_g-guarded in $\mathfrak{A}$. The tree $\mathcal{U}_{\mathrm{BGN}^k[\sigma',\sigma_g]}(\mathfrak{A})$ is consistent and is called a *σ_g-guarded-interface tree* since it alternates between *interface nodes* with strictly σ_g-guarded domains – corresponding to collapse moves in the game – and *bag nodes* with domain of size at most k that are not necessarily σ_g-guarded.
2. **guarded unravelling**: The $\mathrm{G}[\sigma', \sigma_g]$-unravelling is denoted $\mathfrak{D}(\mathcal{U}_{\mathrm{G}[\sigma',\sigma_g]}(\mathfrak{A}))$ and its encoding $\mathcal{U}_{\mathrm{G}[\sigma',\sigma_g]}(\mathfrak{A})$ is obtained by considering only sequences $Y_0 \dots Y_m \in \Pi_k$ such that for all i, Y_i is strictly σ_g-guarded in $\mathfrak{A}$. The tree $\mathcal{U}_{\mathrm{G}[\sigma',\sigma_g]}(\mathfrak{A})$ is consistent and is called a *σ_g-guarded* tree since the domain of every node in the tree is strictly σ_g-guarded.

All of these unravellings are bisimilar to $\mathfrak{A}$, with respect to the appropriate notion of bisimilarity. Because these unravellings have tree decompositions of some bounded tree-width, this proposition implies that these guarded logics have tree-like models. The structural differences in the tree decompositions will be exploited for our definability decision procedures.

3 Decidability via back-and-forth and equivalence

We now give the main components of our approach, and explain how they fit together.

The first component is a *forward mapping*, translating an input GSO formula ϕ_0 to an MSO formula ϕ_0' over tree-codes, holding on the codes that correspond to tree-like models of ϕ_0. We will be interested only in formulas that are invariant under a form of guarded bisimulation or guarded negation bisimulation, so we assume the input is a GN^l-invariant formula, for some $l \geq \mathrm{width}(\sigma)$. For such formulas, we can actually define a forward mapping that produces a μ-calculus formula.

▶ **Lemma 1** (Fwd, adapted from [16]). *Given a $\mathrm{GN}^l[\sigma]$-invariant sentence $\phi \in \mathrm{GSO}[\sigma]$ and given some $k \geq \mathrm{width}(\sigma)$, we can construct $\phi^\mu \in \mathrm{L}_\mu[\Sigma^{code}_{\sigma,\max\{k,l\}}]$ such that for all consistent $\Sigma^{code}_{\sigma,\max\{k,l\}}$-trees $\mathcal{T}$, $\mathcal{T} \models \phi^\mu$ iff $\mathfrak{D}(\mathcal{T}) \models \phi$.*

The second component will depend on our target sublogic $\mathcal{L}_1$. It requires a mapping (not necessarily effective) taking a σ-structure $\mathfrak{B}$ to a tree structure $\mathcal{U}_{\mathcal{L}_1}(\mathfrak{B})$ such that $\mathfrak{D}(\mathcal{U}_{\mathcal{L}_1}(\mathfrak{B}))$ agrees with $\mathfrak{B}$ on all $\mathcal{L}_1$ sentences. Informally, $\mathcal{U}_{\mathcal{L}_1}(\mathfrak{B})$ will be the encoding of some unravelling of $\mathfrak{B}$ appropriate for $\mathcal{L}_1$, perhaps with additional properties. A *backward mapping for* $\mathcal{L}_1$ takes sentences ϕ_0' over tree codes (with some given k and σ) to a sentence $\phi_1 \in \mathcal{L}_1$ such that: for all σ-structures $\mathfrak{B}$, $\mathfrak{B} \models \phi_1$ iff $\mathcal{U}_{\mathcal{L}_1}(\mathfrak{B}) \models \phi_0'$.

The formula ϕ_1 will depend on simplifying the formula ϕ_0' based on the fact that one is working on an unravelling. For $\mathcal{L}_1 = \mathrm{GFP}[\sigma', \sigma_g]$ over subsignatures σ', σ_g of the original signature σ, $\mathcal{U}_{\mathcal{L}_1}(\mathfrak{B})$ will be a guarded unravelling; the results of [16] can easily be refined to give the formula component in $\mathrm{GFP}[\sigma', \sigma_g]$. For GNFP^k, providing both the appropriate unravelling and the formula in the backward mappings will require more work.

The $\mathcal{L}_1$ *definability problem for logic* $\mathcal{L}$ asks: given some input sentence $\phi \in \mathcal{L}$, is there some $\psi \in \mathcal{L}_1$ such that ϕ and ψ are logically equivalent? The forward and backward method of Figure 1b gives us a generic approach to this problem. The algorithm consists of applying the forward mapping to get ϕ_0', applying the backward mapping to ϕ_0' based on $\mathcal{L}_1$ to get ϕ_1, and then checking if ϕ_1 is equivalent to ϕ_0. We claim ϕ_0 is $\mathcal{L}_1$ definable iff ϕ_0 and ϕ_1 are equivalent. If ϕ_0 and ϕ_1 are logically equivalent then ϕ_0 is clearly $\mathcal{L}_1$ definable using ϕ_1. In the other direction, suppose that ϕ_0 is $\mathcal{L}_1$-definable. Fix $\mathfrak{B}$, and let $\mathcal{U}_{\mathcal{L}_1}(\mathfrak{B})$ be given by the backward mapping. Then

$$\begin{aligned}\mathfrak{B} \models \phi_0 &\Leftrightarrow \mathfrak{D}(\mathcal{U}_{\mathcal{L}_1}(\mathfrak{B})) \models \phi_0 \text{ since } \mathfrak{D}(\mathcal{U}_{\mathcal{L}_1}(\mathfrak{B})) \text{ agrees with } \mathfrak{B} \text{ on } \mathcal{L}_1 \text{ sentences} \\ &\Leftrightarrow \mathcal{U}_{\mathcal{L}_1}(\mathfrak{B}) \models \phi_0' \text{ by Lemma Fwd} \Leftrightarrow \mathfrak{B} \models \phi_1 \text{ by Backward Mapping for } \mathcal{L}_1.\end{aligned}$$

Hence, ϕ_0 and ϕ_1 are logically equivalent, as required. Thus, we get the following general decidability result:

▶ **Proposition 2.** *Let $\mathcal{L}_1$ be a subset of $\mathrm{GN}^l[\sigma]$-invariant $\mathrm{GSO}[\sigma]$ such that we have an effective backward mapping for $\mathcal{L}_1$. Then the $\mathcal{L}_1$ definability problem is decidable for $\mathrm{GN}^l[\sigma]$-invariant $\mathrm{GSO}[\sigma]$.*

Above, we mean that there is an algorithm that decides $\mathcal{L}_1$ definability for any input $\mathrm{GSO}[\sigma]$ sentence that is $\mathrm{GN}^l[\sigma]$-invariant, with the output being arbitrary otherwise. The

approach above gives a definability test in the usual sense for inputs in GNFP[σ], since these are all GN^l[σ]-invariant for some l. In particular we will see that we can test whether a GNFP^l[σ] sentence is in GFP[σ'] or in GNFP^k[σ']. But there are larger GN^l-invariant logics (e.g. [7]), and the algorithm immediately applies to these as well.

4 Identifying GFP definable sentences

For GFP, we can instantiate the high-level algorithm by giving a backward mapping.

▶ **Lemma 3** (GFP-Bwd, adapted from [16]). *Given $\phi^\mu \in \mathrm{L}_\mu[\Sigma^{code}_{\sigma,m}]$ and $\sigma_g \subseteq \sigma' \subseteq \sigma$, ϕ^μ can be translated into $\psi \in \mathrm{GFP}[\sigma', \sigma_g]$ such that for all σ-structures $\mathfrak{B}$, $\mathfrak{B} \models \psi$ iff $\mathcal{U}_{\mathrm{G}[\sigma',\sigma_g]}(\mathfrak{B}) \models \phi^\mu$.*

Plugging this into our high-level algorithm, with $\mathcal{U}_{\mathrm{G}[\sigma',\sigma_g]}(\mathfrak{B})$ as $\mathcal{U}_{\mathcal{L}_1}(\mathfrak{B})$, we get decidability of the GFP-definability problem:

▶ **Theorem 4.** *The* GFP[σ', σ_g] *definability problem is decidable for* GN^k[σ]*-invariant* GSO[σ] *where $k \geq \mathrm{width}(\sigma)$ and $\sigma_g \subseteq \sigma' \subseteq \sigma$.*

There are two sources of inefficiency in the high-level algorithm. First, the forward mapping is non-elementary since we pass through MSO on the way to a μ-calculus formula. Second, testing equivalence of the original sentence with the sentence produced by the forward and backward mappings implicitly requires a second forward mapping in order to reduce the problem to regular language equivalence on trees.

For the special case of input in GNFP, we can use an optimized procedure that avoids these inefficiencies and allows us to obtain an optimal complexity bound.

▶ **Theorem 5.** *The* GFP[σ', σ_g] *definability problem is* 2-ExpTime*-complete for input in* GNFP[σ].

The main idea behind our optimized procedure is to directly use automata throughout the process. First, for input ϕ in GNFP it is known from [9] how to give a forward mapping that directly produces a tree automaton $\mathcal{A}_\phi$ (with exponentially-many states) that accepts a consistent tree $\mathcal{T}$ iff $\mathfrak{D}(\mathcal{T}) \models \phi$ – exactly the consistent trees that satisfy ϕ^μ. This direct construction avoids passing through MSO, and can be done in 2-ExpTime. We can then construct an automaton $\mathcal{A}'_\phi$ from $\mathcal{A}_\phi$ that accepts a tree $\mathcal{T}$ iff $\mathcal{U}_{\mathrm{G}[\sigma',\sigma_g]}(\mathfrak{D}(\mathcal{T}))$; we call this the G[$\sigma', \sigma_g$]*-view automaton*, since it mimics the view of $\mathcal{T}$ running on the guarded unravelling of $\mathfrak{D}(\mathcal{T})$. This can be seen as an automaton that represents the composition of the backward mapping with the forward mapping. With these constructions in place, we have the following improved algorithm to test definability of ϕ in GFP: construct $\mathcal{A}_\phi$ from ϕ, construct $\mathcal{A}'_\phi$ from $\mathcal{A}_\phi$, and test equivalence of $\mathcal{A}_\phi$ and $\mathcal{A}'_\phi$ over consistent trees. Note that with this improved procedure it is not necessary to actually construct the backward mapping, or to pass forward to trees for a second time in order to test equivalence. Overall, the procedure can be shown to run in 2-ExpTime. A reduction from GFP-satisfiability testing, which is known to be 2-ExpTime-hard, yields the lower bound.

Our results give us a corollary on definability in fragments of FO when the input is in FO:

▶ **Corollary 6.** *The* GF[σ', σ_g] *definability problem is decidable for* GN^l[σ]*-invariant* FO[σ] *where $k, l \geq \mathrm{width}(\sigma)$ and $\sigma_g \subseteq \sigma' \subseteq \sigma$.*

Note that in this work we are characterizing sublogics within fragments of fixpoint logics and within fragments of first-order logic. We do not deal with identifying first-order definable formulas within a fixpoint logic, as in [10, 8].

We also can get a version of the definability result for a restriction of fixpoint logic. One well-studied restriction is called alternation-freeness (see, e.g., [14, 2]). We say a sentence ϕ in GFP is *alternation-free* if it does not contain subformulas $\psi_1 := [\mathbf{lfp}_{Y,\boldsymbol{y}} . \chi_1](\boldsymbol{y})$ and $\psi_2 := [\mathbf{gfp}_{Z,\boldsymbol{z}} . \chi_2](\boldsymbol{z})$ such that Y occurs in χ_2 and ψ_2 is a subformula of ψ_1, or Z occurs in χ_1 and ψ_1 is a subformula of ψ_2 (recall that a greatest fixpoint can be defined in terms of least fixpoints and negations as $[\mathbf{gfp}_{Z,\boldsymbol{z}} . \chi_2](\boldsymbol{z}) \equiv \neg[\mathbf{lfp}_{Z,\boldsymbol{z}} . \neg\chi_2[\neg Z/Z]](\boldsymbol{z})$). Alternation-free fragments of GNFP and L_μ are defined by restricting the nesting of fixpoints in a similar way. It is desirable to know if a sentence is in this alternation-free fragment of GFP since this fragment has better computational properties: for instance, model checking for this alternation-free fragment can be done in linear time. This was shown in [14]. A language called DATALOG-LITE – a variant of DATALOG that has some restricted forms of negation and universal quantification – was also introduced, and shown to exactly characterize this alternation-free GFP [14]. A corollary of the definability result in this section, is that it is possible to decide definability in alternation-free GFP when the input is in alternation-free GNFP, using the same decision procedure as before. Roughly speaking, this comes from observing that if the input is in alternation-free GNFP, then the forward mapping produces alternation-free L_μ, and the backward mapping produces alternation-free GFP.

▶ **Corollary 7.** *The alternation-free* GFP$[\sigma']$ *(equivalently,* DATALOG-LITE$[\sigma']$*) definability problem is decidable in* 2-ExpTime *for input in alternation-free* GNFP$[\sigma]$.

We can also apply our theorem to answer some questions about *conjunctive queries (CQs)*: formulas built up from relational atoms via $\wedge$ and $\exists$. When the input ϕ to our definability algorithm is a CQ, ϕ can be written as a GF sentence exactly when it is *acyclic*: roughly speaking, this means it can be built up from guarded existential quantification (see [15]). Transforming a query to an acyclic one could be quite relevant in practice, since acyclic queries can be evaluated in linear time [26]. There are well-known methods for deciding whether a CQ ϕ is acyclic, and recently these have been extended to the problem of determining whether ϕ is acyclic for all structures satisfying a set of constraints (e.g., Guarded TGDs [6] or Functional Dependencies [13]). Using Corollary 6 above along with an equivalence between guardedness and acyclicity that follows from [4], we can get an analogous result for arbitrary constraints in the guarded fragment:

▶ **Corollary 8.** *Given a set of* GF *sentences* Σ *and a CQ sentence* Q*, we can decide whether there is a union of acyclic CQs* Q' *equivalent to* Q *for all structures satisfying* Σ*. The problem is* 2-ExpTime*-complete.*

Note that if Σ consists of universal Horn constraints ("TGDs"), then a CQ Q is equivalent to union of CQs Q' relative to Σ implies that it is equivalent to one of the disjuncts of Q'. Thus the result above implies decidability of acyclicity relative to universal horn GF sentences, one of the main results of [6].

5 Identifying GNFP^k and UNFP^k sentences

We now turn to extending the prior results to GNFP and UNFP. In order to make use of the back-and-forth approach described in the previous section, we must be able to restrict to structures of some bounded tree-width. For GFP this tree-width depends only on the signature σ', so this width-restriction was implicit in the GFP$[\sigma', \sigma_g]$-definability problems. However, for GNFP and UNFP this bound on the tree-width depends on the width of the formula, so for definability questions, we must state this width explicitly. Hence, in this section, we consider definability questions related to GNFP^k and UNFP^k.

We apply the high-level algorithm of Proposition 2, using the forward mapping of Lemma 1. The unravelling and backward mapping for GNFP^k is more technically challenging than the corresponding construction for GFP.

We first need an appropriate notion of unravelling. We use a variant of the block k-width guarded negation unravelling discussed in Section 2, but we will need to assume we have a certain repetition of facts. This idea of modifying a classical unravelling to include extra copies of certain pieces of the structure has been used before (e.g. the ω-expansions in [21], and "shrewd" unravellings for UNFP^k in [9]). We will need a new, subtler property for GNFP^k, which we call "plumpness".

In order to define the property that this special unravelling has, we need to define how we can modify copies of certain parts of the structure in a way that still leads to a GN^k-bisimilar structure. Let τ and τ' be sets of σ'-facts over some elements A. Let $I, J \subseteq A$. We say τ and τ' agree on J if for all σ'-atoms $R(a_1, \ldots, a_l)$ with $\{a_1, \ldots, a_l\} \subseteq J$, $R(a_1, \ldots, a_l) \in \tau$ iff $R(a_1, \ldots, a_l) \in \tau'$. We say τ' is an (σ_g, I)-*safe restriction* of τ if (i) $\tau' \subseteq \tau$; (ii) τ' agrees with τ on I; (iii) τ' agrees with τ on every $J \subseteq A$ that is σ_g-guarded in τ'. Note that τ itself is considered a trivial (σ_g, I)-safe restriction of τ. Here is another example:

▶ **Example 9.** Consider signatures $\sigma' = \{U, R, T\}$ and $\sigma_g = \{R\}$, where U is a unary relation, R is a binary relation, and T is a ternary relation. Consider $I = \{1, 2\}$ and $\tau = \{U(1),\, U(3),\, R(1,2),\, R(2,3),\, R(3,1),\, T(3,2,2)\}$. Then the possible (σ_g, I)-safe restrictions of τ are τ itself and

$$\tau'_1 = \left\{ \begin{array}{c} U(1),\, U(3) \\ R(1,2),\, R(2,3) \\ T(3,2,2) \end{array} \right\} \qquad \tau'_2 = \left\{ \begin{array}{c} U(1),\, U(3) \\ R(1,2),\, R(3,1) \\ T(3,2,2) \end{array} \right\} \qquad \tau'_4 = \left\{ \begin{array}{c} U(1),\, U(3) \\ R(1,2) \\ T(3,2,2) \end{array} \right\}$$

$$\tau'_3 = \left\{ \begin{array}{c} U(1),\, U(3) \\ R(1,2),\, R(3,1) \end{array} \right\} \qquad \tau'_5 = \left\{ \begin{array}{c} U(1),\, U(3) \\ R(1,2) \end{array} \right\}.$$

Note that we cannot drop facts over unary relations (since these are always trivially guarded), and we can never drop facts over I. Further, the σ_g-facts that we keep restrict what other facts we can drop, since for any σ_g-guarded set that remains we must preserve facts over that set. By a (σ_g, I)-safe restriction of a node in a tree decomposition, we mean a (σ_g, I)-safe restriction of the atoms represented by the node. We will be interested in trees with the property that for every bag node w, all safe restrictions of w are realized by siblings of w. Formally a $\Sigma^{\mathrm{code}}_{\sigma',k}$-tree has the σ_g-*plumpness property* if for all interface nodes v: if w is a ρ_0-child of v over names J with $I = \mathrm{rng}(\rho_0)$ and τ is the encoded set of σ'-atoms that hold at w, then for any (σ_g, I)-safe restriction τ' of τ, there is a ρ_0-child w' of v such that (i) τ' is the encoded set of σ'-atoms that hold at w'; (ii) for each ρ-child u' of w', there is a ρ-child u of w such that the subtrees rooted at u and u' are bisimilar; and (iii) for each ρ-child u of w such that $\mathrm{dom}(\rho)$ is strictly σ_g-guarded in τ', there is a ρ-child u' of w' such that the subtrees rooted at u' and u are bisimilar.

▶ **Example 10.** Let $\mathcal{T}$ be a plump tree. Suppose there is an interface node v in $\mathcal{T}$ with label encoding $\tau_0 = \{U(1), R(1,2)\}$, and there is a ρ_0-child w of v such that the label of w is the encoding of $\tau = \{U(1), U(3), R(1,2), R(2,3), R(3,1), T(3,2,2)\}$ from Example 9, and ρ_0 is the identity function with domain $\{1, 2\}$. Then by plumpness there must also be ρ_0-children $w_1, \ldots, w_5$ of v with labels encoding $\tau'_1, \ldots, \tau'_5$ from Example 9.

The following proposition shows that one can obtain unravellings that are plump:

▶ **Proposition 11.** *Let $\mathfrak{B}$ be a σ-structure, $k \in \mathbb{N}$, and $\sigma_g \subseteq \sigma' \subseteq \sigma$. There is a consistent, plump, σ_g-guarded-interface tree $\mathcal{U}^{\text{plump}}_{\text{BGN}^k[\sigma',\sigma_g]}(\mathfrak{B})$ such that $\mathfrak{B}$ is $\text{BGN}^k[\sigma',\sigma_g]$-bisimilar to $\mathfrak{D}(\mathcal{U}^{\text{plump}}_{\text{BGN}^k[\sigma',\sigma_g]}(\mathfrak{B}))$. We call $\mathfrak{D}(\mathcal{U}^{\text{plump}}_{\text{BGN}^k[\sigma',\sigma_g]}(\mathfrak{B}))$ the* plump unravelling *of $\mathfrak{B}$.*

Returning to the components required for the application of Proposition 2, we see that Proposition 11 says that $\mathfrak{B}$ is $\text{BGN}^k[\sigma',\sigma_g]$-bisimilar to $\mathfrak{D}(\mathcal{U}^{\text{plump}}_{\text{BGN}^k[\sigma',\sigma_g]}(\mathfrak{B}))$ as required for an application of Proposition 2. Plumpness will come into play in the backward mapping:

▶ **Lemma 12** (GNFP^k-Bwd). *Given $\phi^\mu \in \text{L}_\mu[\Sigma^{code}_{\sigma,m}]$, relational signatures σ_g and σ' with $\sigma_g \subseteq \sigma' \subseteq \sigma$, and $k \leq m$, we can construct $\psi \in \text{GNFP}^k[\sigma',\sigma_g]$ such that for all σ-structures $\mathfrak{B}$, $\mathfrak{B} \models \psi$ iff $\mathcal{U}^{\text{plump}}_{\text{BGN}^k[\sigma',\sigma_g]}(\mathfrak{B}) \models \phi^\mu$.*

There is a naïve backward mapping of the μ-calculus into LFP, by structural induction. The problem is that the formula produced by the translation fails to be in GNFP^k for two reasons. First, the inductive step for negation in the naïve algorithm simply applies negation to the recursively-produced formula. Clearly this can produce unguarded negation. Similarly, the recursive step for fixpoints may use unguarded fixpoints.

For example, the original μ-calculus formula can include subformulas of the form $\langle\rho\rangle\text{ExactLabel}(\tau)$ where τ is a set of unary relations from $\Sigma^{\text{code}}_{\sigma',k}$, and $\text{ExactLabel}(\tau)$ asserts P for all $P \in \tau$ and $\neg P$ for all unary relations P not in τ. This would be problematic for a straightforward backward mapping, since the backward translation of some $\neg R_{i_1,\ldots,i_n}$ would be converted into an unguarded negation $\neg R(x_{i_1},\ldots,x_{i_n})$. On the other hand the formula $\langle\rho\rangle\text{GNLabel}(\tau)$ where $\text{GNLabel}(\tau)$ asserts P for all $P \in \tau$ but only asserts $\neg P$ for unary relations P that are not in τ but whose indices are σ_g-guarded by some $P' \in \tau$ would be unproblematic, since this could be translated to a formula with σ_g-guarded negation. The key observation is that from an interface node in a plump tree, these two formulas are equivalent: if $\mathcal{T}, v \models \langle\rho\rangle\text{GNLabel}(\tau)$ at any interface node v, then plumpness ensures that if there is some ρ-child w' of v with label τ' satisfying $\text{GNLabel}(\tau)$, then there is a ρ-child w of v with label τ satisfying $\text{ExactLabel}(\tau)$ – it can be checked that τ is a $(\sigma_g, \text{rng}(\rho))$-safe restriction of τ'. Thus the proof of Lemma GNFP^k-Bwd relies on first simplifying L_μ-formulas so that problematic subformulas like $\text{ExactLabel}(\tau)$ are eliminated, with the correctness of this simplification holding only over plump trees. After this simplification, an inductive backward mapping can be applied.

Using the above lemma and Proposition 2, we obtain the following analog of Theorem 4.

▶ **Theorem 13.** *The $\text{GNFP}^k[\sigma',\sigma_g]$ definability problem is decidable for $\text{GN}^l[\sigma]$-invariant $\text{GSO}[\sigma]$ and $k, l \geq \text{width}(\sigma)$.*

Since $\text{UNFP}^k[\sigma']$ is just $\text{GNFP}^k[\sigma',\emptyset]$, we obtain the following corollary:

▶ **Corollary 14.** *The $\text{UNFP}^k[\sigma']$ definability problem is decidable for $\text{GN}^l[\sigma]$-invariant $\text{GSO}[\sigma]$ and $k, l \geq \text{width}(\sigma)$.*

We get corollaries for fragments of FO, analogous to Corollary 6:

▶ **Corollary 15.** *The $\text{GNF}^k[\sigma',\sigma_g]$ and $\text{UNF}^k[\sigma']$ definability problems are decidable for $\text{GN}^l[\sigma]$-invariant $\text{FO}[\sigma]$ and $k, l \geq \text{width}(\sigma)$.*

We can also apply the backward and forward mappings to get a semantic characterization for GNFP^k, analogous to the Janin-Walukiewicz theorem. The following extends a result of [3] characterizing GNF^k formulas as the BGN^k-invariant fragment of FO.

▶ **Theorem 16.** *$\text{GNFP}^k[\sigma',\sigma_g]$ is the $\text{BGN}^k[\sigma',\sigma_g]$-invariant fragment of $\text{GSO}[\sigma']$.*

The proof is similar to the characterizations of Janin-Walukiewicz and [16], and can also be seen as a variant of Proposition 2, where we use $\mathrm{BGN}^k[\sigma', \sigma_g]$-invariance rather than equivalence to a $\mathrm{GNFP}^k[\sigma', \sigma_g]$ sentence in justifying that the input formula is equivalent to the result of the composition of backward and forward mappings.

Interpolation. The forward and backward mappings utilized for the definability questions can also be used to prove that GFP and GNFP^k have a form of interpolation.

Let ϕ_{L} and ϕ_{R} be sentences over signatures σ_{L} and σ_{R} such that $\phi_{\mathrm{L}} \models \phi_{\mathrm{R}}$ (ϕ_{L} entails ϕ_{R}). An *interpolant* for such a validity is a formula θ for which $\phi_{\mathrm{L}} \models \theta$ and $\theta \models \phi_{\mathrm{R}}$, and θ mentions only relations appearing in both ϕ_{L} and ϕ_{R}. We say a logic $\mathcal{L}$ has *Craig interpolation* if for all $\phi_{\mathrm{L}}, \phi_{\mathrm{R}} \in \mathcal{L}$ with $\phi_{\mathrm{L}} \models \phi_{\mathrm{R}}$, there is an interpolant $\theta \in \mathcal{L}$ for it. We say a logic $\mathcal{L}$ has the stronger *uniform interpolation* property if one can obtain θ from ϕ_{L} and a signature σ', and θ can serve as an interpolant for any ϕ_{R} entailed by ϕ_{L} and such that the common signature of ϕ_{R} and ϕ_{L} is contained in σ'. A uniform interpolant can be thought of as the best over-approximation of ϕ_{L} over σ'.

Uniform interpolation holds for L_μ [11] and UNFP^k [9]. Unfortunately, $\mathrm{GFP}[\sigma]$ and $\mathrm{GNFP}^k[\sigma]$ both fail to have uniform interpolation and Craig interpolation (this follows from [19, 9]). However, if we disallow subsignature restrictions that change the guard signature, then we regain interpolation. This "preservation of guard" variant was investigated first by Hoogland, Marx, and Otto in the context of Craig interpolation [19]. The uniform variant was introduced by D'Agostino and Lenzi [12], who called it *uniform modal interpolation*. Formally, we say a guarded logic $\mathcal{L}[\sigma, \sigma_g]$ with guard signature $\sigma_g \subseteq \sigma$ has *uniform modal interpolation* if for any $\phi_{\mathrm{L}} \in \mathcal{L}[\sigma, \sigma_g]$ and any subsignature $\sigma' \subseteq \sigma$ containing σ_g, there exists a formula $\theta \in \mathcal{L}[\sigma', \sigma_g]$ such that ϕ_{L} entails θ and for any σ'' containing σ_g with $\sigma'' \cap \sigma \subseteq \sigma'$ and any $\phi_{\mathrm{R}} \in \mathcal{L}[\sigma'', \sigma_g]$ entailed by ϕ_{L}, θ entails ϕ_{R}. It was shown in [12] that GF has uniform modal interpolation. We strengthen this to GFP and GNFP^k.

▶ **Theorem 17.** *For σ a relational signature, $\sigma_g \subseteq \sigma$, $k \in \mathbb{N}$: $\mathrm{GFP}[\sigma, \sigma_g]$ and $\mathrm{GNFP}^k[\sigma, \sigma_g]$ sentences have uniform modal interpolation, and the interpolants can be found effectively.*

We sketch the argument for $\mathrm{GFP}[\sigma, \sigma_g]$. Consider $\phi_{\mathrm{L}} \in \mathrm{GFP}[\sigma, \sigma_g]$ of width k and subsignature $\sigma' \subseteq \sigma$ containing σ_g. We apply Lemma Fwd to get a formula $\phi_{\mathrm{L}}^\mu \in \mathrm{L}_\mu[\Sigma^{\mathrm{code}}_{\sigma,k}]$ that captures codes of tree-like models of ϕ_{L}. We want to go backward now, to get a formula over the subsignature σ'. We saw that the backward mapping for $\mathrm{GFP}[\sigma', \sigma_g]$ (Lemma 3) can do this: it can start with a μ-calculus formula over $\Sigma^{\mathrm{code}}_{\sigma,k}$, and produce a formula in $\mathrm{GFP}[\sigma', \sigma_g]$. As discussed earlier, the formula produced by this backward mapping has a nice property related to definability: it is equivalent to ϕ_{L} exactly when ϕ_{L} is definable in $\mathrm{GFP}[\sigma', \sigma_g]$. In general, however, we do not expect ϕ_{L} to be equivalent to a formula over the subsignature – for uniform interpolation we just want to *approximate* the formula over this subsignature. The backward mapping of ϕ_{L}^μ does not always do this. Hence, it is necessary to add one additional step before taking the backward mapping: we apply uniform interpolation for the μ-calculus [11], obtaining $\theta^\mu \in \Sigma^{\mathrm{code}}_{\sigma',k}$ which is entailed by ϕ_{L}^μ and entails each $\mathrm{L}_\mu[\Sigma^{\mathrm{code}}_{\sigma',k}]$-formula implied by ϕ_{L}^μ. Finally, we apply Lemma $\mathrm{GFP}[\sigma', \sigma_g]$-Bwd to θ^μ to get $\theta \in \mathrm{GFP}[\sigma', \sigma_g]$. We can check that $\theta \in \mathrm{GNFP}^k[\sigma', \sigma_g]$ is the required uniform modal interpolant for ϕ_{L} over subsignature σ'.

Theorem 17 also implies that UNFP^k has the traditional uniform interpolation property: since the guard signature is empty for UNFP^k, uniform modal interpolation and uniform interpolation coincide. This was shown already in [9].

6 Conclusions

In this paper we have taken a first look at effective characterizations of definability in expressive logics. We did not allow constants in the formulas in this paper, but we believe that similar effective characterization results hold for guarded fixpoint logics with constants. We leave open the question of definability in GNFP without any width restriction. For this the natural way to proceed is to bound the width of a defining sentence based in terms of the input. We also note that our results on fixpoint logics hold only when equivalence is considered over all structures, leaving open the corresponding questions over finite structures.

Acknowledgements. We thank the referees for many improvements.

References

1 Hajnal Andréka, Johan van Benthem, and István Németi. Modal languages and bounded fragments of predicate logic. *J. Phil. Logic*, 27:217–274, 1998.

2 Andre Arnold and Damin Niwinski. *Rudiments of mu-calculus*. Elsevier, 2001.

3 Vince Bárány, Balder Ten Cate, and Luc Segoufin. Guarded negation. *J. ACM*, 62(3), 2015. doi:10.1145/2701414.

4 Vince Bárány, Georg Gottlob, and Martin Otto. Querying the guarded fragment. In *LMCS*, volume 10, 2014. doi:10.2168/LMCS-10(2:3)2014.

5 Vince Bárány, Balder ten Cate, and Luc Segoufin. Guarded negation. In *ICALP*, 2011. doi:10.1007/978-3-642-22012-8_28.

6 Pablo Barceló, Georg Gottlob, and Andreas Pieris. Semantic acyclicity under constraints. In *PODS*, 2016. doi:10.1145/2902251.2902302.

7 Michael Benedikt, Pierre Bourhis, and Michael Vanden Boom. A step up in expressiveness of decidable fixpoint logics. In *LICS*, 2016. doi:10.1145/2933575.2933592.

8 Michael Benedikt, Balder ten Cate, Thomas Colcombet, and Michael Vanden Boom. The complexity of boundedness for guarded logics. In *LICS*, 2015. doi:10.1109/LICS.2015.36.

9 Michael Benedikt, Balder ten Cate, and Michael Vanden Boom. Interpolation with decidable fixpoint logics. In *LICS*, 2015. doi:10.1109/LICS.2015.43.

10 Achim Blumensath, Martin Otto, and Mark Weyer. Decidability results for the boundedness problem. *Logical Methods in Computer Science*, 10(3), 2014. doi:10.2168/LMCS-10(3:2)2014.

11 Giovanna D'Agostino and Marco Hollenberg. Logical Questions Concerning The mu-Calculus: Interpolation, Lyndon and Los-Tarski. *The Journal of Symbolic Logic*, 65(1):310–332, 2000. doi:10.2307/2586539.

12 Giovanna D'Agostino and Giacomo Lenzi. Bisimulation quantifiers and uniform interpolation for guarded first order logic. *Theor. Comput. Sci.*, 563:75–85, 2015. doi:10.1016/j.tcs.2014.08.015.

13 Diego Figueira. Semantically acyclic conjunctive queries under functional dependencies. In *LICS*, 2016. doi:10.1145/2933575.2933580.

14 Georg Gottlob, Erich Grädel, and Helmut Veith. Datalog LITE: a deductive query language with linear time model checking. *ACM Trans. Comput. Log.*, 3(1):42–79, 2002. doi:10.1145/504077.504079.

15 Georg Gottlob, Nicola Leone, and Francesco Scarcello. Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width. *J. Comput. Syst. Sci.*, 66(4):775–808, 2003. doi:10.1016/S0022-0000(03)00030-8.

16 Erich Grädel, Colin Hirsch, and Martin Otto. Back and forth between guarded and modal logics. *ACM TOCL*, 3(3):418–463, 2002. doi:10.1145/507382.507388.

17 Erich Grädel and Martin Otto. The freedoms of (guarded) bisimulation. In *Johan van Benthem on Logic and Information Dynamics*, pages 3–31. Springer, 2014. doi:10.1007/978-3-319-06025-5_1.

18 Erich Grädel and Igor Walukiewicz. Guarded fixed point logic. In *LICS*, 1999. doi:10.1109/LICS.1999.782585.

19 Eva Hoogland, Maarten Marx, and Martin Otto. Beth definability for the guarded fragment. In *LPAR*, 1999. doi:10.1007/3-540-48242-3_17.

20 David Janin and Igor Walukiewicz. Automata for the modal mu-calculus and related results. In *MFCS*, 1995. doi:10.1007/3-540-60246-1_160.

21 David Janin and Igor Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In *CONCUR*, 1996. doi:10.1007/3-540-61604-7_60.

22 Martin Otto. Eliminating recursion in the μ-calculus. In *STACS*, 1999. doi:10.1007/3-540-49116-3_50.

23 Balder ten Cate and Luc Segoufin. Unary negation. In *STACS*, 2011. doi:10.4230/LIPIcs.STACS.2011.344.

24 J. F. A. K. van Benthem. *Modal Logic and Classical Logic*. Humanities Pr, 1983.

25 Moshe Y. Vardi. "Why is Modal Logic so Robustly Decidable". In *Descriptive Complexity and Finite Models*, pages 149–184, 1997.

26 Mihalis Yannakakis. Algorithms for acyclic database schemes. In *VLDB*, 1981.