A survey on guarded negation

Luc Segoufin

A survey on guarded negation

Luc Segoufin, INRIA and ENS-Cachan

We consider a logical framework building on existential positive formulas and then adding guarded negations and guarded fixpoints, where the guards are atomic formulas containing all free variables. The resulting first-order and fixpoint logics turn out to have nice algorithmic properties and nice expressive power. We survey some of them.

1. INTRODUCTION

First-order logic is widely used in mathematics, philosophy and computer science. It offers a good trade-off between simplicity and expressive power. In particular it is at the core of the relational database query language SQL.

However for certain applications it is too expressive. In particular it can describe the runs of a Turing Machine, making its satisfiability problem (asking whether a given formula is satisfied by at least one model) undecidable. This implies that first-order logic cannot be used for the formal methods approach in many areas of computer science, such as verification where satisfiability is crucial.

Several decidable fragments of first-order logic have been proposed. The most popular ones being the two-variable fragment [Mortimer 1975], and Modal Logic.

The case of Modal Logic is intriguing as many of its variations and extensions remain decidable (like adding two-way navigation, fixpoints etc.), while so many extensions of the two-variable fragment of first-order logic are undecidable [Grädel et al. 1999]. This led Moshe Vardi to wonder "why is modal logic so robustly decidable?" [Vardi 1996] and to answer by a combination of several key properties: (i) the tree model property (if a formula has a model then it has a model that is a tree) and (ii) translability into tree automata (each formula can be effectively translated into a tree automata recognizing its tree models). From the decidability of the emptiness problem of tree automata, these two properties already imply the decidability of the existence of a (possibly infinite) model. Finite satisfiability follows from a third key property: (iii) finite model property (if a formula has a model then it has a finite one). The first two properties are satisfied by all extensions of Modal Logic, in particular its two-way and fixpoint extensions. However, the finite model property is no longer satisfied by the fixpoint extension of Modal Logic, and if finite satisfiability is still decidable in this setting, it requires more complicated arguments [Bojańczyk 2003].

The situation is similar for guarded quantification first-order logic[1]. It was introduced in [Andréka et al. 1998] as an extension of Modal Logic with good algorithmic properties and again many of its extensions remain decidable, in particular the one with guarded fixpoints. As noticed by Erich Grädel [Grädel 2001], the reason is that it enjoys similar properties as Modal Logic. The tree model property is replaced by the tree-like model property: any satisfiable formula has a model of bounded tree-width. The translability into tree automata should now be understood as transforming the formula into a tree automata recognizing the tree decompositions of those models of the formula that have bounded tree-width.

It turns out that the story continues. By moving the guards from quantifications to negations one gets a richer logic, guarded negation first-order logic, that has the tree-like model property and the translability to automata [Bárány et al. 2015]. This is the framework that we study in this survey.

[1] In order to avoid confusion with **guarded negation** logics, we use the terminology **guarded quantification** first-order logic instead of the usual one: guarded first-order logic.

We will see that guarded negation first-order logic, together with its extension with guarded fixpoints, has many nice properties. The first one being invariance under a notion of bisimulation that we survey in Section 3. This property, denoted guarded negation bisimulation, generalizes modal and guarded quantification bisimulation, yields the tree-like model property and opens the door to decidability.

Satisfiability for infinite models follows by translability into tree automata as explained in Section 4. For finite models, this is solved in the first-order case with the finite model property and for the fixpoint case this is achieved by a reduction to the modal case.

The nice algorithmic properties of guarded negation logic are further studied in Section 5, where the complexity of the model checking problem is studied.

Unlike the modal and the guarded quantification fragments, guarded negation first-order logic also has nice model theoretical properties. We will see in Section 6 that it has Craig Interpolation and the Projective Beth Property. These are the key to many applications, some of them being discussed in section 7.

2. PRELIMINARIES

Guarded negation logics are fragments of first-order logic and least fixpoint logic over relational schemes. Their models are relational structures. Before defining the logics, we recall some classical definitions.

Relational structures. A relational schema is a finite set of relation symbols, each having an associated arity. A relational *structure* M over a relational schema consists of a set, the *domain* of M, together with an interpretation of each relation symbol R of arity k of the schema as a k-ary relation over the domain denoted $R(M)$. A structure M is said to be *finite* if its domain is finite.

Homomorphisms. An *homomorphism* from a structure M to a structure N (assuming they have the same schema) is a mapping h from the domain of M to the domain of N such that for any relation symbol R of the schema and any tuple $\bar{a}$ that belongs to the interpretation of R in M, its image $h(\bar{a})$ belongs to the interpretation of R in N. This is not to be confused with the notion of *isomorphism* that requires moreover that the mapping is bijective and that the converse holds: $\bar{a}$ belongs to the interpretation of R in M iff $h(\bar{a})$ belongs to the interpretation of R in N. An homomorphism or an isomorphism is said to be *partial* if it is defined only on a subset of the domain of the structure.

Logic. We assume familiarity with first-order logic, FO, and least-fixpoint logic, LFP, over relational structures. We use classical syntax and semantics for FO and LFP. In particular we write $\phi(\bar{x})$ to denote the fact that the free variables of ϕ are exactly the variables in $\bar{x}$. We also write $M \models \phi(\bar{u})$ for the fact that the tuple $\bar{u}$ of elements of the model M makes the formula $\phi(\bar{x})$ true over M. A *sentence* is a formula with no free variable. It is either true or false over a structure and therefore defines a property of structures. The *size of a formula* ϕ is the number of symbols needed to write down the formula.

This note surveys guarded negation logics. There is a first-order variant, that we now define. Later we will add fixpoints to it.

Guarded negation first-order logic is a fragment of first-order logic where any negated subformula must be *guarded* by an atom containing all its free variables.

Definition 2.1. The formulas of guarded negation first-order logic, denoted GNFO, are given by the following grammar, where R ranges over relation symbols of the

$(P)^*$	is	$P(x)$
$(\phi \wedge \psi)^*$	is	$\phi^*(x) \wedge \psi^*(x)$
$(\neg\phi)^*$	is	$\neg\phi^*(x)$
$(\langle R\rangle\phi)^*$	is	$\exists y\ R(x,y) \wedge \phi^*(y)$
$(\langle R^{-1}\rangle\phi)^*$	is	$\exists y\ R(y,x) \wedge \phi^*(y)$
$(\mathbf{S}\phi)^*$	is	$\exists y\ \phi^*(y)$

Fig. 1. Inductive translation of a modal logic formula ϕ to an equivalent UNFO-formula $\phi^*(x)$

schema and $\alpha(\bar{x}\bar{y})$ is an atomic formula:

$$\varphi ::= \quad R(\bar{x}) \mid x = y \mid \exists x\ \varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \alpha(\bar{x}\bar{y}) \wedge \neg\varphi(\bar{y}) \mid \neg\varphi(x) \mid \neg\varphi()$$

In words, any existential positive first-order formula, i.e. with no negation nor universal quantification, is a formula of GNFO. The syntax builds on existential positive first-order formulas by adding the negation of sentences, the negation of formulas with one free variable and formulas of the form $\alpha(\bar{x}\bar{y}) \wedge \neg\varphi(\bar{y})$, requiring that all the free variables $\bar{y}$ of φ occur in the atom $\alpha(\bar{x}\bar{y})$, called *the guard* of the negation.

If the formula only negates sentences or formulas with one free variable, then we say that it has *unary negations*. We denote by UNFO the set of unary negation first-order formulas. UNFO is a fragment of GNFO of independent interest. Unary negation can be seen as a special case of guarded negation, as $\neg\varphi(x)$ is equivalent to $x = x \wedge \neg\varphi(x)$, where the guard is an equality atom.

Notice that $x \neq y$ is not a formula of GNFO but $R(x,y,z) \wedge x \neq y$ is. Universal quantifications can only be used via double negations. For example $\forall\bar{x}\ \alpha(\bar{x}) \rightarrow \varphi(\bar{x})$ is equivalent to the GNFO formula $\neg(\exists\bar{x}\ \alpha(\bar{x}) \wedge \neg\varphi(\bar{x}))$.

Example 2.2. It is is easy to see that modal logic, and many of its extensions, is a fragment of GNFO, actually of UNFO. To see this consider the *global two-way modal logic*[2] defined by the grammar

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

where P is a unary relation symbol (also called *proposition* in this setting), and R is a binary relation symbol (also called an *accessibility relation* in this context). Its semantics can be given via an inductive translation into UNFO as depicted in Figure 1.

Example 2.3. It turns out that GNFO also generalizes the *guarded quantification fragment of first-order logic*[3], denoted GFO, introduced by [Andréka et al. 1998] as a generalization of modal logic, requiring a guard on quantifications instead on negations. The logic GFO is the fragment of FO defined by the following grammar, where, again, $\alpha(\bar{x}\bar{y}\bar{z})$ is an atomic formula (possibly an equality statement):

$$\varphi ::= R(\bar{x}) \mid x = y \mid \varphi \vee \varphi | \varphi \wedge \varphi | \neg\varphi \mid \exists\bar{x}\ \alpha(\bar{x}\bar{y}\bar{z}) \wedge \varphi(\bar{x}\bar{y}) \mid \forall\bar{x}\ \alpha(\bar{x}\bar{y}\bar{z}) \rightarrow \varphi(\bar{x}\bar{y})$$

By pushing the guards on the quantifications down to the atoms, it is easy to see that every GFO sentence is equivalent to a GNFO sentence. For instance the sentence $\exists x_1x_2x_3E(x_1,x_2,x_3) \wedge E(x_1,y) \wedge \neg E(x_2,x_3)$ is equivalent to $\exists x_1x_2x_3E(x_1,x_3) \wedge E(x_1,x_2,x_3) \wedge \neg E(x_2,x_3)$. This is only true for *sentences*, as $\neg R(xy)$ is a formula of GFO but is not expressible in GNFO. Guarded negation is strictly more expressive than

[2]Traditionally, the basic modal logic is defined without the backward and global features and can only navigate by traversing an edge in the forward direction.

[3]Recall that in order to avoid confusion with guarded **negation** logics, we refer to GFO as guarded **quantification** first-order logic, instead of its usual name: guarded first-order logic.

guarded quantification as witnessed by the following sentence:

$$q = \exists xy\big(E(x,y) \wedge \neg(\exists uvw\ E(x,u) \wedge E(u,v) \wedge E(v,w) \wedge E(w,y))\big). \quad (1)$$

The formula q asks for the existence of an edge whose endpoints cannot be connected by a path of length 4. It is not equivalent to any GFO sentence as q defines a property that is not invariant under guarded quantification bisimulation, see Section 3.

GNFO has one important parametrization, denoted the *width*. Given a GNFO formula, one can push the existential quantifications up until a negation, a disjunction, or another existential quantification is reached. The maximal length of a block of existential quantifications in the resulting formula is called the width of the initial formula (see [Bárány et al. 2015] for a formal definition). We will see that the width of a formula is related to the tree-width of its models.

Guarded negation fixpoint logic is the extension of the guarded negation first-order logic with a guarded fixpoint modality. It is a syntactic fragment of least-fixpoint logic, from which it inherits the semantics.

Definition 2.4. For each relational schema τ, guarded negation fixpoint logic, denoted GNFP, is defined by the following grammar (notice that the first line correspond to the grammar of GNFO):

$$\begin{array}{rl} \phi \ ::= & R(\bar{x}) \mid x = y \mid \phi \wedge \phi \mid \phi \vee \phi \mid \exists x\, \phi \mid \alpha(\bar{x}\bar{y}) \wedge \neg\phi(\bar{x}) \mid \neg\varphi(x) \mid \neg\varphi() \mid \\ & \beta(\bar{u}\bar{w}) \wedge Z(\bar{u}) \mid \mu_{Z,\bar{z}}[\,\phi(\bar{Y}, Z, \bar{z})\,](\bar{x}) \end{array}$$

where R is any relational symbol in τ, $\alpha(\bar{x}\bar{y})$ and $\beta(\bar{u}\bar{w})$ are atomic τ-formulas (possibly equality statements) and, in the last clause of the definition, the fixpoint variable Z occurs only positively in $\phi(\bar{Y}, Z, \bar{z})$, the *matrix* of the fixpoint, i.e. always under an even number of negations. For any value of $\bar{Y}$, the matrix formula can be seen as a function computing a relation, the valuations of $\bar{z}$ making it true, from the relation Z. The fact that Z occurs under an even number of negations ensures that this function is monotonic, and therefore that it has a least fixpoint, which is the semantics of the fixpoint formula.

Formulas of GNFP can be naturally thought of as being built up from atomic formulas using (i) guarded negation first-order formulas and (ii) guarded fixpoint operators. It is important to note that:

— no first-order parameters (i.e., free variables other than those $\bar{z}$ bound by the fixpoint operator) are permitted in the matrix of a fixpoint operator,
— free fixpoint variables $\bar{Y}$ other than Z are still allowed, enabling nesting and alternation of fixpoint definitions;
— fixpoint variables cannot be used as guards, and in fact, all atomic formulas involving fixpoint variables must be guarded by atomic τ-formulas or equalities.

As for the first-order case, we have a unary fixpoint variant denoted UNFP. It is the fragment of GNFP restricted to unary negations and unary fixpoints predicates. The width of a GNFP formula is defined as the width of its first-order parts.

Example 2.5. The fixpoint formula

$$\mu_{Z,x,y}[\,E(x,y) \vee \exists z\ (Z(x,z) \wedge E(z,y))\,](u,v)$$

computing the transitive closure of E is not a formula of GNFP as the matrix formula does not guard the variables x, z occurring in $Z(x,z)$.

The fixpoint formula

$$\mu_{Z,z}[\,y = z \vee \exists y'(Z(y') \wedge E(y',z))\,](x)$$

computing the connected component of y is also not a formula of GNFP as the matrix formula has y as a parameter.

However the fixpoint formula

$$\mu_{Z,z}[\, B(z) \vee \exists y'(Z(y') \wedge E(y', z))\,](x)$$

computing the set of nodes reachable from a node in B is in GNFP.

One could imagine different other syntaxes for guarding the fixpoint matrices in GNFP. For instance we could require that the whole matrix formula is guarded with a single atom or with a specific predicate signifying guardedness without expressly declaring any concrete guard. It turns out that this would not change the expressive power nor the complexity for satisfiability. However it does affect the succinctness of the formula and the complexity of the model checking. See the discussion concerning the syntax in [Bárány et al. 2015].

Example 2.6. Consider the following formula over a binary relation symbol E:

$$\mu_{X,y}[\, \neg\exists z(E(z, y) \wedge \neg X(z))\,]$$

It is a valid formula of UNFP as the monadic fixpoint variable X occurs within the scope of two negations. When evaluated on a directed graph, the first stage of the fixpoint computation contains all nodes y of in-degree 0. At any further stage of the fixpoint computation we add to X all points whose incoming nodes where all already in X. Hence if all the backwards paths starting from a given node are finite, this node will eventually be part of X. Conversely if a node has an infinite backward path, it will never be part of X.

Hence the GNFP formula:

$$\exists x\, \neg\mu_{X,y}[\, \neg\exists z(E(z, y) \wedge \neg X(z))\,](x) \tag{2}$$

expresses the fact that the graph contains an infinite backward path. In particular, in the case of finite structures, it expresses the fact that the graph contains a directed cycle.

Example 2.7. Greatest fixpoints can be simulated via the usual triple negations. It can be verified that these negations are legal (this follows from the fact that the matrix formula is guarded). For instance the formula of the previous example computes exactly the complement of the greatest fixpoint of the matrix formula $\exists z E(z, y) \wedge X(z)$, denoted $\neg\nu_{X,y}[\, \exists z E(z, y) \wedge X(z)\,]$.

Allowing simultaneous fixpoints in GNFP formulas does not increase the expressive power of the logic (although it can facilitate more succinct definitions).

Example 2.8. Guarded negation fixpoint logic generalizes the extension of the two-way modal logic of Example 2.2 with monadic fixpoint. It also generalizes the *guarded quantification fragment of fixpoint logic* (GFP) [Grädel and Walukiewicz 1999], in the sense that every sentence of GFP is equivalent to a sentence of GNFP.

Guarded quantification fixpoint logic is the fragment of least-fixpoint logic obtained by extending guarded quantification first-order logic with least fixpoints: given a formula $\phi(\bar{Y}, Z, \bar{z})$ that is positive in Z, has no free first-order variables other than $\bar{z}$ and Z has arity the number of variables in $\bar{z}$, the formula $\mu_{Z,\bar{z}}[\, \phi(\bar{Y}, Z, \bar{z})\,]$ is also a formula of GFP. Although the occurrences of fixpoint variables are not required to be guarded, in the context of a GFP *sentence*, every occurrence of an atom using a fixpoint relation is implicitly guarded, namely by the atom guarding the closest quantifier whose scope includes the occurrence in question). This implies that every sentence of GFP is equivalent to a sentence of GNFP, via a polynomial time transformation.

3. INVARIANCE BY BISIMULATION

The key to understand the expressive power of guarded negation logics, and also the key for its decidability, is the notion of *guarded negation bisimulation*. Bisimulation is a notion of similarity between two structures that is weaker than isomorphism. The desired property, Theorem 3.3 below, is that any two similar structures satisfy the same sentences of the logic.

This theorem has many important consequences. We describe two of them below.

The first one is a limitation on the expressive power of the logic. A property is said to be *closed under bisimulation* if any two similar structures either both satisfy the property or both falsify the property. It is important to understand that this notion has two flavors, depending on whether we consider only finite structures or all structures, finite or infinite. If we consider only finite structures, we say that a property is *closed under finite bisimulation* if any two similar *finite* structures either both satisfy the property or both falsify the property. Clearly closure under bisimulation implies closure under finite bisimulation. But the opposite is false. For instance if the property requires infinite models (this can be enforced in many ways, see for instance the paragraph after Theorem 4.3) then any two finite models falsify the property hence the property is trivially closed under finite bisimulation; however the property can additionally require another property that is not closed under bisimulation.

The key theorem mentioned above implies that a property that is not closed under bisimulation is not definable in the logic and a property that is not closed under finite bisimulation is not definable in the logic over finite models. We then say that the logic is *closed under bisimulation*.

The second consequence is the decidability of the satisfiability problem by showing that any structure is similar to a simple one, here of bounded tree-width, and then showing that it is decidable whether a formula has a simple model, see Section 4 below.

This is a classical track that has been successfully used for many logics. For instance modal logics, such as those defined in Section 2, define only properties closed under a notion of similarity known as *modal bisimulation*. Modal bisimulation has been extended to a notion of *guarded quantification bisimulation* that corresponds to the guarded quantification logics GFO and GFP.

We now introduce the appropriate notion of bisimulation for guarded negation logics, namely *guarded negation bisimulation* [4].

Let M be a relational structure. If a tuple of elements $\bar{a}$ from the domain of M belongs to the interpretation of a relation symbol R, then we say that $R(\bar{a})$ is a *fact* of M. We say that a tuple of elements of M is *guarded* if it is a singleton or there is a fact of M containing all its components. We denote by $\textit{guarded}(M)$ the set of guarded tuples of M. For a number k, we say that a tuple is k-*guarded* if it is guarded by a fact of M using at most k elements of M. We denote by $k\text{-}\textit{guarded}(M)$ the set of all k-guarded tuples of M. In particular $\textit{guarded}(M) = \bigcup_k k\text{-}\textit{guarded}(M)$.

Definition 3.1. Let M, N be two structures and k a strictly positive integer. A guarded negation bisimulation, GN-bisimulation in short, of width k is a non-empty binary relation $Z \subseteq k\text{-}\textit{guarded}(M) \times k\text{-}\textit{guarded}(N)$ such that the following hold for every pair $(\bar{a}, \bar{b}) \in Z$, where $\bar{a} = a_1, \ldots, a_m$ and $\bar{b} = b_1, \ldots, b_n$

— the mapping sending $\bar{a}$ to $\bar{b}$ is a partial isomorphism (and in particular, $m = n$)
— [**Forward clause**] For every set X included in the domain of M and of size bounded by k there is a partial homomorphism $h : M \to N$ whose domain is X, such that h is consistent with the mapping sending $\bar{a}$ to $\bar{b}$ (i.e. $h(a_i) = b_i$ for all a_i in X), and such

[4] There is a stronger variant of guarded negation bisimulation introduced in [Bárány et al. 2013] that already characterizes GNFO.

that for every $\bar{a}' \in k\text{-}\mathit{guarded}(M)$ consisting of elements in X, the pair $(\bar{a}', h(\bar{a}'))$ belongs to Z.

— [**Backward clause**] For every set X included in the domain of N and of size bounded by k there is a partial homomorphism $h : N \to M$ whose domain is X, such that h is consistent with the mapping sending $\bar{b}$ to $\bar{a}$ (i.e. $h(b_i) = a_i$ for all b_i in X), and such that for every $\bar{a}' \in k\text{-}\mathit{guarded}(N)$ consisting of elements in X, the pair $(h(\bar{a}'), \bar{a}')$ belongs to Z.

If we only require X to be finite and replace $k\text{-}\mathit{guarded}(M)$ and $k\text{-}\mathit{guarded}(N)$ by $\mathit{guarded}(M)$ and $\mathit{guarded}(N)$ in the forward and backward clauses, we then say that Z is a GN-bisimulation between M and N. In particular a GN-bisimulation is a GN-bisimulation of width k, for all k.

We write $M \approx_{GN} N$ if there is a GN-bisimulation between M and N and write $M \approx^k_{GN} N$ if there is a GN-bisimulation of width k.

Discussion. In the definition of guarded negation bisimulation, if the sets X are restricted to guarded tuples, instead of arbitrary finite sets, we then have the definition of guarded quantification bisimulation that was designed for guarded quantification logics [Flum et al. 2008].

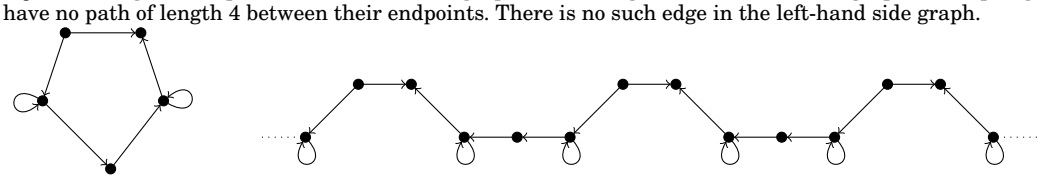
If we restrict in the definition of guarded negation bisimulation $\mathit{guarded}(M)$ and $\mathit{guarded}(N)$ to respectively the domain of M and the domain of N (in particular $Z \subseteq \mathit{dom}(M) \times \mathit{dom}(N)$) then we get the notion of *unary negation bisimulation* [ten Cate and Segoufin 2013] that is well suited for UNFO and UNFP as we shall see.

The notion of modal bisimulation, designed for two way modal logics, corresponds to the restriction of unary negation bisimulation, as defined previously, where in the forward and backward clauses for a pair (a, b) of Z, we only consider the sets X with two elements, the element a together with one element connected to a in the forward case, and similarly with b in the backward case.

This means that the existence of a guarded negation bisimulation implies the existence of a guarded quantification bisimulation which implies the existence of a modal bisimulation. This is to be expected as we have seen that modal logic is a fragment of guarded quantification logic which is also a fragment of guarded negation logic.

Example 3.2. Consider again the query q defined in GNFO by the formula (1), asking for the existence of an edge whose endpoints connot be connected by a path of length 4. This property is not closed under modal and guarded bisimulation. To see this consider the graph depicted in the left-hand side of Figure 2. The query is not satisfied by this graph: the top edge has its endpoints connected by a path of length 4 and any other edge starts or ends with a loop and therefore has its endpoints connected by a path of length 4. However this graph is guarded quantification bisimilar, and therefore also modal bisimilar, to the one depicted in the right-hand side of the figure, which makes the query true with any top edges. As the right-hand side graph is infinite, this

Fig. 2. Two guarded quantification bisimilar graphs. In the right-hand side infinite graph, the top edges have no path of length 4 between their endpoints. There is no such edge in the left-hand side graph.



implies that the property defined by q is not expressible in modal or guarded quantifi-

cation logics over arbitrary graphs. However the right-hand side graph can be made finite by identifying top edges that are sufficiently far away in order to make sure that no loops of length 4 are created, hence still satisfying q. This implies that the property is also not definable over finite graphs.

The key property mentioned in the discussion above states that the existence of a guarded negation bisimulation implies indistinguishability by guarded negation fixpoint sentences. More precisely we have the following result:

THEOREM 3.3. *[Bárány et al. 2015]*

— *For every sentence φ of* GNFP *there is a k such that, if $M \approx^k_{GN} N$ then M and N agree on φ.*
— *If $M \approx_{GN} N$ then M and N satisfy the same sentences of* GNFP.

The number k associated to φ in Theorem 3.3 turns out to be its width as defined in Section 2. Hence any two structures that are GN-bisimilar with width k satisfy the same GNFP sentences of width k. A similar result can be shown for unary negation fragments:

THEOREM 3.4. *[ten Cate and Segoufin 2013]*

— *For every sentence φ of* UNFP *there is a k such that, if $M \approx^k_{UN} N$ then M and N agree on φ.*
— *If $M \approx_{UN} N$ then M and N satisfy the same sentences of* UNFP.

We now argue that guarded negation bisimulation is the best possible notion of similarity under which guarded negation logics are invariant. Indeed, guarded negation bisimulation invariance can be used to *characterize* GNFO: any property that is definable in first-order logic and closed under guarded negation bisimulation is actually definable in GNFO. This result works for both types of closure, closure under finite guarded negation bisimulation and closure under arbitrary guarded negation bisimulation. Similar results were obtained earlier showing that modal bisimulation characterizes modal logic [van Benthem 1983; Rosen 1997] and that guarded quantification bisimulation characterizes guarded quantification first-order logic [Andréka et al. 1998; Otto 2010].

THEOREM 3.5.

— *a first-order property is definable in* GNFO *iff it is closed under GN-bisimulation.*
— *a first-order property is definable in* GNFO *over finite structures iff it is closed under finite GN-bisimulation.*

The characterization of GNFO was obtained in [Bárány et al. 2015] for the infinite case and in [Otto 2013] for the (considerably harder) finite case.

Similar results were obtained for UNFO.

THEOREM 3.6. *[ten Cate and Segoufin 2013]*

— *a first-order property is definable in* UNFO *iff it is closed under UN-bisimulation.*
— *a first-order property is definable in* UNFO *over finite structures iff it is closed under finite UN-bisimulation.*

The results of Theorem 3.5 and Theorem 3.6 can actually be refined further by showing in each setting, that for each k, a first-order property is closed under GN-bisimulation of width k iff it is definable by a GNFO sentence of width k (and similarly for UN-bisimulation and UNFO).

The situation for fixpoint logics is not yet entirely solved. It is known that the μ-calculus is the modal bisimulation invariant fragment of monadic second order logic (MSO) [Janin and Walukiewicz 1996] and that the guarded quantification fixpoint logic is the guarded quantification bisimulation invariant fragment of guarded second order logic (GSO, where second order quantification is limited to relations containing guarded tuples) [Grädel et al. 2002]. However those results only hold for infinite structures. We don't know yet whether they are true over finite structures. For guarded negation fixpoint logic, only the unary negation case has been partially answered. It is known that the sentences of UNFP of width k are precisely the properties expressible in GSO and invariant under unary negation bisimulations of width k [Benedikt et al. 2015].

4. DECIDABILITY

In this section we consider the satisfiability problem. For this problem the input is a sentence of the logic and the question is whether this sentence has a model, i.e. a structure making the sentence true. This problem has two variants depending on whether we ask for a *finite* model or an arbitrary one. It is usually easier to decide whether a sentence has a model while it is harder to decide whether it has a finite model. In some cases, both the finite and infinite satisfiability are equivalent in the sense that a sentence has a model iff it has a finite model. We then say that the logic has the *finite model property*.

In general the satisfiability problem for first-order logic is undecidable, both in its finite or infinite variants. However several fragments of first-order logic are decidable. This is in particular the case for the frameworks already mentioned in this note: for instance it is decidable whether a sentence of the μ-calculus, with its two-way extension, has a model. The problem is ExpTime-complete both for infinite satisfiability [Vardi 1998] and for finite satisfiability [Bojańczyk 2003]. For guarded quantification first-order logic, GFO, satisfiability is 2EXPTIME-complete [Grädel 2001]. Because GFO has the finite model property, it does not make any difference whether we consider finite or infinite satisfiability. Note that the complexity lowers to ExpTime-complete if the maximal arity of the relations in the schema is fixed (for instance graphs). For guarded quantification fixpoint logic, GFP, the satisfiability problem is ExpTime-complete [Grädel and Walukiewicz 1999]. Finite satisfiability was considerably harder to achieve but turns out to be also decidable within the same complexity bounds [Bárány and Bojańczyk 2012].

An important consequence of GN-bisimulation, actually GN-bisimulation of width k for some k, is decidability. It is easy to see that any structure is GN-bisimilar of width k to a structure of tree-width k. It is not important to know the definition of tree-width to understand this note. A structure of tree-width 1 is a tree and the smaller the tree-width is the more the structure resembles a tree. The interested reader is refered to [Flum et al. 2008] for more details. The important result relating tree-width and decidability is Courcelle's Theorem stating that, for any k, it is decidable whether a sentence of monadic second-order logic has a model of tree-width k [Courcelle 1990].

A GN-bisimulation can be seen as a game between two players. A configuration of the game consists of a k-guarded tuple $\bar{a}$ of structure M and a k-guarded tuple $\bar{b}$ of structure N. In a round of the game the first player chooses a set X of size at most k in either M or N. The second player must respond with a partial homomorphism h from X to the other structure. Finally the first player chooses a k-guarded tuple $\bar{a}'$ within X and the game resumes with the pair $\bar{a}'$ and $h(\bar{a}')$. It is now easy to verify that the existence of a GN-bisimulation is equivalent to the fact that the second player has a strategy for playing this game forever. This strategy can be encoded as a tree structure where every node corresponds to the current configuration and the edges to

its children correspond to the answers of the second player to each possible move of the first player. The resulting tree is infinite as the play goes on forever.

Consider any structure M. There is a trivial GN-bisimulation between M and itself, hence a trivial strategy for the second player to play forever. As explained above, this strategy yields an infinite tree structure that can be seen as a tree decomposition of a new (infinite) structure N, witnessing that N has tree-width k and that M is GN-bisimilar to a structure of tree-width k. The structure N obtained this way is called the *GN-unraveling of M of width k*.

Hence, by invariance under GN-bisimulation, if a sentence φ of GNFP has a model then there is a k such that φ has a model of tree-width k. This is known as the *tree-like model property*.

THEOREM 4.1. *[Bárány et al. 2015]* GNFP *has the tree-like model property.*

Theorem 4.1 can be used to decide whether a sentence of GNFP has a model. For this, one constructs from a sentence φ of GNFP a MSO formula recognizing the unravelingsof the models of φ of width k, where k is the width of φ (the width is computable from φ). This implies decidability using Courcelle's Theorem. However this strategy does not give a good complexity bound. In order to obtain the optimal complexity bound given below one needs to directly compute a small automaton working on the tree decompositions of the structures of width k, and this requires more work.

THEOREM 4.2. *[Bárány et al. 2015] It is 2*EXPTIME*-complete to decide whether a sentence of* GNFP *is satisfiable.*

Note that if the upper bound applies to GNFP the lower bound already holds for UNFO [ten Cate and Segoufin 2013] and already in the case of graphs.

What about finite satisfiability? How can we decide whether a sentence has a finite model? We cannot use anymore the tree-like model property for finite satisfiability as the unravelings are inherently infinite.

Like for modal and guarded quantification logics, the situation is then different depending on whether we consider the first-order case or the fixpoint case. The simplest of the two is the case of GNFO. In this case one can show that the logic has the finite model property: if a sentence has a model it has a finite one. Hence finite satisfiability is equivalent to satisfiability and Theorem 4.2 applies. More precisely we can show that GNFO sentences have a small model property:

THEOREM 4.3. *[Bárány et al. 2015] Every satisfiable sentence φ of* GNFO *has a finite model of size double exponential in the size of φ.*

The finite model property does not hold for GNFP. To see this recall the formula (2) of Example 2.6 expressing the fact that a graph has an infinite backward path: $\exists x\ \neg\mu_{X,y}[\ \neg\exists z(E(z,y) \wedge \neg X(z))\ ](x)$.

Consider now the following GNFP formula:

$$\exists x\ \neg\exists y\ E(x,y) \vee \exists x \neg\mu_{X,y}[\ \neg\exists z(E(z,y) \wedge \neg X(z))](x)$$

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

The decidability for finite satisfiability turns out to be considerably more difficult. It has been achieved by reducing the finite satisfiability problem for GNFP to the finite satisfiability problem for GFP. The latter reduces to the finite satisfiability problem for the two-way μ-calculus [Bárány and Bojańczyk 2012], whose finite satisfiability

was obtained in [Bojańczyk 2003]. Altogether it can be achieved within the same complexity bounds as for the infinite case.

THEOREM 4.4. *[Bárány et al. 2015] It is 2*EXPTIME*-complete to decide whether a sentence of* GNFP *has a a finite model.*

5. MODEL CHECKING

In this section we study the model checking problem for guarded negation logics. In the model checking problem, the input consists of a sentence and a structure and the goal is to decide whether the structure satisfies the sentence.

In the case of modal logics and guarded quantification logics, the model checking for GFO is PTime-complete and the one of GFP is the same as for the μ-calculus and lies between PTime and NP $\cap$ coNP [Berwanger and Grädel 2001]. It is actually a famous open problem to know whether there exists a polynomial time algorithm for the model checking problem of μ-calculus. Equivalently this amounts to find a polynomial time algorithm for solving parity games (see [Flum et al. 2008]).

Before stating the results for guarded negation logics we briefly review the complexity classes involved.

The first class we use is denoted P^{NP}, also known as Δ_2^p. It consists of all problems that are computable by a Turing machine running in time polynomial in the size of its input, where the Turing machine, at any point during its computation, can ask yes/no queries to an NP oracle, and take the answers of the oracle into account in subsequent steps of the computation (including subsequent queries to the NP oracle). Analogously, one can define the classes $\mathrm{NP}^{\mathrm{NP}}$ and $\mathrm{coNP}^{\mathrm{NP}}$, which are also known as Σ_2^p and Π_2^p, respectively. An example of a P^{NP}-complete problem is LEX(SAT), which takes as input a Boolean formula $\phi(x_1, \dots x_n)$ and asks what is the value of x_n in the lexicographically maximal solution (where x_n is treated as the least significant bit in the ordering) [Wagner 1987].

Inside P^{NP} lies a hierarchy of classes $\mathrm{P}^{\mathrm{NP}[O(\log^i n)]}$ with $i \geq 1$. They are defined in the same way as P^{NP}, except that the number of yes/no queries that can be asked to the NP oracle is bounded by $O(\log^i(n))$, where n is the size of the input. An example of complete problem for $\mathrm{P}^{\mathrm{NP}[O(\log^i n)]}$ is the problem LEX_i(SAT) testing, given a Boolean formula $\phi(x_1, \dots, x_n)$ and a number $k \leq \log^i(n)$, whether the value of x_k is 1 in the lexicographically maximal solution [ten Cate and Segoufin 2013].

We are now ready to state the model checking results for guarded negation logics.

THEOREM 5.1. *[Bárány et al. 2015] The model checking problem for* GNFO *is* $\mathrm{P}^{\mathrm{NP}[O(\log^2 n)]}$*-complete under polynomial time reductions. For* GNFP *it is in* $\mathrm{NP}^{\mathrm{NP}} \cap \mathrm{coNP}^{\mathrm{NP}}$ *and hard for* P^{NP}.

The hardness results already hold for unary negated formulas.

The model checking problem for GNFO provides one of the few natural complete problems for the complexity class $\mathrm{P}^{\mathrm{NP}[O(\log^2 n)]}$.

For UNFP, the gap between the upper bound and the lower bound reflects the similar open problem for GFP and the μ-calculus where the model checking problem lies between PTime and NP $\cap$ coNP [Berwanger and Grädel 2001].

The result for GNFP is sensitive to the syntax. Various ways of guarding the fixpoint leads to different complexities. For instance it becomes ExpTime-complete if we use a syntactic clause "guard" whose semantics contains all guarded tuples [Bárány et al. 2015].

6. MODEL THEORY

A nice thing with guarded negation first-order logic is that it also behaves well in terms of model theoretic properties. In particular it has Craig-Interpolation, and therefore the Projective Beth Property. For the guarded quantification logic GFO, only Beth Property holds [Hoogland et al. 1999]. Both Craig-Interpolation and Projective Beth Property fail for GFO [Bárány et al. 2013].

Given two formulas φ and ψ, we write $\varphi \models \psi$ if any model of φ is also a model of ψ, in other words, if $\varphi \wedge \neg\psi$ is not satisfiable. A logic has Craig-Interpolation if whenever two formulas φ and ψ of the logic are such that $\varphi \models \psi$, then there is a third formula θ of the logic, using only the relation symbols occurring in both φ and ψ such that $\varphi \models \theta$ and $\theta \models \psi$.

In particular, consider a formula $\varphi(R)$ expressing a property of a relation R and such that $\exists R\ \varphi(R) \models \forall R\ \varphi(R)$. We then say that φ is R-invariant (the result does not depend on the actual value of R). Let $\varphi(S)$ be the formula constructed from $\varphi(R)$ replacing the symbols R by a fresh new symbol S. It follows from $\exists R\ \varphi(R) \models \forall R\ \varphi(R)$ that $\varphi(R) \models \varphi(S)$. As R and S do not appear in both sides, Craig-Interpolation implies in this case that there exists a formula θ, which does not mention R nor S and such that θ is equivalent to $\exists R\ \varphi(R)$. In the literature we then say that R-invariant properties are expressible without R.

Another consequence of Craig-Interpolation is the Projective Beth Property.

Let σ, τ be two relational schemes such that $\sigma \subset \tau$ and R is a relation symbol in $\tau \setminus \sigma$. [5] Let φ be a formula over the schema τ. We say that φ *implicitly defines* R if for any structure M over σ and any two τ-expansions[6] M_1 and M_2 of M such that $M_1 \models \varphi$ and $M_2 \models \varphi$ then we have $R(M_1) = R(M_2)$. In other words the formula φ defines a partial function associating to a model M over σ an instantiation for R, namely $R(M_1)$ for an arbitrary τ-expansion of M, the definition ensuring that it does not depend on the choice of M_1. The mapping is partial because M may not have a τ-expansion satisfying φ.

Example 6.1. Consider the schema $\sigma = \{E\}$ where E is binary whose models are directed graphs. Consider the extension τ of σ by four unary predicates $R, P_0, P_1,$ and P_2.

The following formula says that nodes in R must be on a cycle of length 4.

$$\forall x\ R(x) \rightarrow \exists u, v, w\ E(x,u) \wedge E(u,v) \wedge E(v,w) \wedge E(w,x)$$

It does not implicitly define R as these nodes may or may not be in R. However, in conjunction with the following formula, implying that when a node is not in R then it cannot be on a cycle of length 4,

$$\begin{aligned} \forall x\ \neg R(x) &\rightarrow P_0(x) \wedge \neg P_1(x) \wedge \neg P_2(x) \\ \forall x, y\ P_i(x) \wedge E(x,y) &\rightarrow P_{i+1 \bmod 3}(y) \end{aligned}$$

the resulting formula implicitly defines R as the nodes lying on a cycle of length 4. Note that all formulas are in UNFO as all negations are unary. Note that the mapping is partial as in some graphs, it is not possible to assign colors such that the whole formula is true. But when this is possible, R contains all nodes in a cycle of length 4.

[5] Beth Property requires $\tau = \sigma \cup \{R\}$. The Projective Beth Property is a generalization of Beth Property where τ (and therefore the formula) may contain more relations

[6] A τ-expansion of M is a structure over the same domain as M, with the same interpretation of relation symbols in σ. Hence the τ-expansions of M only differ by their interpretations of the relation symbols in $\tau \setminus \sigma$

A logic has the Projective Beth Property if every relation implicitly definable in the logic by a formula φ over the schema τ has an explicit definition in the logic, i.e. there is a formula $\phi(\bar{x})$ over the schema σ (in particular ϕ does not use the symbol R) such that $\varphi \models \forall \bar{x}\ R(\bar{x}) \leftrightarrow \phi(\bar{x})$.

Example 6.2. For instance the formula of Example 6.1 implicitly defining the nodes sitting on a cycle of length 4 in a graph can be explicitly defined by the formula

$$\exists u, v, w\ E(x, u) \wedge E(u, v) \wedge E(v, w) \wedge E(w, x).$$

THEOREM 6.3. GNFO *has Craig-Interpolation and therefore also the Projective Beth Property.*

Theorem 6.3 was first proved in [Bárány et al. 2013]. A constructive proof was then given in [Benedikt et al. 2016]. Actually GNFO even has Lyndon Interpolation (the interpolant θ also preserves the positivity of the relations symbols) [Benedikt et al. 2016].

When both sides of $\varphi \models \psi$ have only unary negations, the interpolant can be chosen in UNFO [ten Cate and Segoufin 2013]. One can further assume that the interpolant uses the same number of variables as φ and ψ [ten Cate and Segoufin 2015].

There are very few results of this kind for fixpoint logics. However Craig Interpolation does hold for unary negation fixpoint logic, UNFP, but fails for guarded negation fixpoint logic, GNFP [Benedikt et al. 2015].

7. APPLICATIONS

Databases. There are numerous applications in databases. A *view* of a database is a new database providing a subset of its information possibly restructured in a different way. A view is usually specified using queries, each query defining a new relation of the view. Views have many applications, a notable one being privacy, when some users may not be entitled to see all of the database. In this context it may be useful to known whether a view specification leaks some important confidential information. The problem of view determinacy models this by asking whether a view specification contains enough information for answering a given target query [Nash et al. 2010]. This problem is known to be already undecidable for views specified by means of conjunctive queries and a conjunctive target query [Gogacz and Marcinkowski 2016]. However for view specifications defined by answer-guarded[7] GNFO queries and the target query is also answer-guarded the problem becomes decidable. It is a consequence of the Projective Beth Property of GNFO that if a view determines a query then a rewriting of the query can be found in GNFO. Decidability follows from the decidability of GNFO [Bárány et al. 2013].

Database systems are constantly facing incomplete or inconsistent data. In this case, computing the answers to a query requires some reasoning and therefore a decidable logic. Several decidable scenarios with integrity constraints specified using guarded negation rules where considered in [Bárány et al. 2013; Bourhis et al. 2016; Bourhis et al. 2014]. See also [Bárány et al. 2013; Bienvenu et al. 2014].

Trees. When restricted to unranked ordered tree structures (also known as XML trees), UNFO has the same expressive power as Core-XPath, the navigational fragment of XPath [ten Cate and Segoufin 2013]. This equivalence of expressive power holds when Core-XPath is viewed as a language of sentences (existence of a path) and then it corresponds to UNFO sentences. The equivalence also holds when Core-XPath is viewed as a language defining properties of a node (existence of a path starting

[7] A query is answer guarded if it is of the form $\alpha(\bar{x}) \wedge \varphi(\bar{x})$ where α is an atom.

from this node) and then it corresponds to UNFO formulas with one free variable. In this case it also has the same expressive power as FO^2 [Marx and de Rijke 2005]. The equivalence also holds when Core-XPath is viewed as a language defining pairs of nodes (with a certain path linking them) and then it corresponds to UNFO formulas with two free variables. In this setting UNFP sentences define precisely the regular tree languages.

Boundedness. In some cases it is useful to know whether a property expressed using a fixpoint formula does require a fixpoint. In other words whether the fixpoint computation can be replaced by a first-order formula. This is known as the *boundedness problem*. This problem is decidable for a fragment of guarded negation fixpoint: guarded negation Datalog [Bárány et al. 2012].

8. CONCLUSION

We have seen that guarded negation logics have nice algorithmic properties and nice expressive power.

There exists several decidable extensions of what we presented in this paper. For instance atomic guards can be replaced by "clique guards" [Bárány et al. 2015]. However a little bit of unguarded negation, like adding just inequality, kills decidability [Bárány et al. 2015]. In the fixpoint case it is possible to relax the requirement forbidding the existence of parameters. Satisfiability of an infinite model is then decidable, the finite case being open [Benedikt et al. 2016].

REFERENCES

Hajnal Andréka, Johan van Benthem, and István Németi. 1998. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic* 27 (1998), 217–274.

Vince Bárány, Michael Benedikt, and Pierre Bourhis. 2013. Access patterns and integrity constraints revisited. In *Intl. Conf. on Database Theory (ICDT)*.

Vince Bárány, Michael Benedikt, and Balder ten Cate. 2013. Rewriting Guarded Negation Queries. In *Intl. Symp. on Mathematical Foundations of Computer Science (MFCS)*.

Vince Bárány and Mikołaj Bojańczyk. 2012. Finite satisfiability for guarded fixpoint logic. *Inform. Process. Lett.* 112, 10 (2012), 371–375.

Vince Bárány, Balder ten Cate, and Martin Otto. 2012. Queries with Guarded Negation. In *Intl. Conf. on Very Large Databases (VLDB)*.

Vince Bárány, Balder ten Cate, and Luc Segoufin. 2015. Guarded Negation. *J. ACM* 62, 3 (2015), 22:1–22:26.

Michael Benedikt, Pierre Bourhis, and Michael Vanden Boom. 2016. A Step Up in Expressiveness of Decidable Fixpoint Logics. In *Symp. on Logic in Computer Science (LICS)*. 817–826.

Michael Benedikt, Balder ten Cate, and Michael Vanden Boom. 2015. Interpolation with Decidable Fixpoint Logics. In *Symp. on Logic in Computer Science (LICS)*. 378–389.

Michael Benedikt, Balder ten Cate, and Michael Vanden Boom. 2016. Effective Interpolation and Preservation in Guarded Logics. *ACM Trans. Computational Logic* 17, 2 (2016), 8:1–8:46.

Dietmar Berwanger and Erich Grädel. 2001. Games and Model Checking for Guarded Logics. In *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*. 70–84.

Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. 2014. Ontology-Based Data Access: A Study through Disjunctive Datalog, CSP, and MMSNP. *ACM Trans. Database Systems* 39, 4 (2014), 33:1–33:44.

Mikołaj Bojańczyk. 2003. The finite graph problem for two-way alternating automata. *Theoretical Computer Science* 3, 298 (2003), 511–528.

Pierre Bourhis, Marco Manna, Michael Morak, and Andreas Pieris. 2016. Guarded-Based Disjunctive Tuple-Generating Dependencies. *ACM Trans. Database Systems* 41, 4 (2016), 27:1–27:45.

Pierre Bourhis, Michael Morak, and Andreas Pieris. 2014. Acyclic Query Answering under Guarded Disjunctive Existential Rules and Consequences to DLs. In *Intl. Workshop on Description Logics.*

Bruno Courcelle. 1990. The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs. *Information and Computation* 85, 1 (1990), 12–75.

Jörg Flum, Erich Grädel, and Thomas Wilke (Eds.). 2008. *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*. Texts in Logic and Games, Vol. 2. Amsterdam University Press.

Tomasz Gogacz and Jerzy Marcinkowski. 2016. Red Spider Meets a Rainworm: Conjunctive Query Finite Determinacy Is Undecidable. In *Symp. on Principles of Database Systems (PODS)*.

Erich Grädel. 2001. Why are Modal Logics so Robustly Decidable? In *Current Trends in Theoretical Computer Science*. 393–408.

Erich Grädel, Colin Hirsch, and Martin Otto. 2002. Back and forth between guarded and modal logics. *ACM Trans. Computational Logic* 3, 3 (2002), 418–463.

Erich Grädel, Martin Otto, and Eric Rosen. 1999. Undecidability results on two-variable logics. *Archive for Mathematical Logic* 38, 4-5 (1999), 313–354.

Erich Grädel and Igor Walukiewicz. 1999. Guarded Fixed Point Logic. In *Symp. on Logic In Computer Science (LICS)*.

Eva Hoogland, Maarten Marx, and Martin Otto. 1999. Beth Definability for the Guarded Fragment. In *Logic Programming and Automated Reasoning (LPAR)*.

David Janin and Igor Walukiewicz. 1996. On the Expressive Completeness of the Propositional μ-Calculus w.r.t. Monadic Second-Order Logic. In *Intl. Conf. on Concurrency Theory (CONCUR)*.

Maarten Marx and Maarten de Rijke. 2005. Semantic characterizations of navigational XPath. *SIGMOD Record* 34, 2 (2005), 41–46.

Michael Mortimer. 1975. On languages with two variables. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* 21, 8 (1975), 135–140.

Alan Nash, Luc Segoufin, and Victor Vianu. 2010. Views and queries: Determinacy and rewriting. *ACM Trans. Database Syst.* 35, 3 (2010), 21:1–21:41.

Martin Otto. 2010. Highly Acyclic Groups, Hypergraph Covers and the Guarded Fragment. In *Symp. on Logic In Computer Science (LICS)*.

Martin Otto. 2013. Expressive completeness through logically tractable models. *Annals of Pure and Applied Logic* 164, 13 (2013), 1418–1453.

Eric Rosen. 1997. Modal logic over finite structures. *Journal of Logic, Language, and Computation* 6, 4 (1997), 427–439.

Balder ten Cate and Luc Segoufin. 2013. Unary negation. *Logical Methods in Computer Science (LMCS)* 9 (2013).

Balder ten Cate and Luc Segoufin. 2015. Craig Interpolation for bounded variable fragments of guarded negation. (2015). Unpublished manuscript.

Johan van Benthem. 1983. *Modal Logic and Classical Logic*. Bibliopolis.

Moshe Y. Vardi. 1996. Why is Modal Logic So Robustly Decidable?. In *Descriptive Complexity and Finite Models*. 149–184.

Moshe Y. Vardi. 1998. Reasoning about The Past with Two-Way Automata. In *Intl. Colloquium on Automata, Languages and Programming (ICALP)*.

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