Complexity Classifications via Algebraic Logic

Reijo Jaakkola
Tampere University, Finland

Antti Kuusisto
Tampere University, Finland
University of Helsinki, Finland

Abstract

Complexity and decidability of logics is an active research area involving a wide range of different logical systems. We introduce an algebraic approach to complexity classifications of computational logics. Our base system GRA, or general relation algebra, is equiexpressive with first-order logic FO. It resembles cylindric algebra but employs a finite signature with only seven different operators, thus also giving a very succinct characterization of the expressive capacities of first-order logic. We provide a comprehensive classification of the decidability and complexity of the systems obtained by limiting the allowed sets of operators of GRA. We also discuss variants and extensions of GRA, and we provide algebraic characterizations of a range of well-known decidable logics.

2012 ACM Subject Classification Theory of computation → Logic

Keywords and phrases Decidability, complexity, algebraic logic, fragments of first-order logic

Digital Object Identifier 10.4230/LIPIcs.CSL.2023.27

Related Version *Full Version*: https://arxiv.org/abs/2005.01184

Funding The authors were funded by the Academy of Finland project Theory of computational logics, grant numbers 324435, 328987 (to December 2021) and 352419, 352420, 353027 (2022 onwards).
Reijo Jaakkola: Academy of Finland project Theory of Computational Logics, grant 328987
Antti Kuusisto: Academy of Finland project Theory of Computational Logics, grants 324435, 328987, 352419, 352420, 353027

1 Introduction

The fall of Hilbert's program and the realization of the undecidability of first-order logic FO put an end to the most prestigious plans of automating mathematical reasoning. However, research with more modest aims continued right away. Perhaps the most direct descendant of Hilbert's program was the work on the *classical decision problem*, i.e., the initiative to classify the quantifier prefix classes of FO according to whether they are decidable or not. This major program was successfully completed in the 1980's, see [7] for an overview.

Subsequent work has been more scattered but active. The current state of the art on decidability and complexity of fragments of FO divides roughly into two main branches: research on variants of *two-variable logic* FO^2 and the *guarded fragment* GF. Two-variable logic FO^2 is the fragment of FO that allows only two variable symbols. It was shown decidable in [33] and NExpTime-complete in [12]. The extension of FO^2 with counting quantifiers, or C^2, was proved decidable in [13, 35] and NExpTime-complete in [36]. Research on variants of FO^2 is currently very active, see, e.g., [5, 8, 21, 22, 29, 43] for some recent work. See also [14, 20] where the *uniform one-dimensional fragment* U_1 is defined. This system extends FO^2 to a logic that allows an arbitrary number of variables but remains NExpTime-complete.

The guarded fragment GF was initially conceived as an extension of modal logic, being a system where quantification is similarly localized as in the Kripke semantics for modal logic. After its introduction in [1], it was soon shown 2ExpTime-complete in [11]. The guarded fragment has proved successful in relation to applications, and it has been extended in several


© Reijo Jaakkola and Antti Kuusisto;
licensed under Creative Commons License CC-BY 4.0

ways, see, e.g., the *loosely guarded* [6], *clique guarded* [10] and *packed* [30] fragments. All these logics have the same 2ExpTime-complete complexity as GF (see, e.g., [4]). The more recently introduced *guarded negation fragment* GNFO [4] is a very expressive extension of GF based on restricting the use of negation in the same way GF restricts quantification. The logic GNFO also extends the *unary negation fragment* UNFO [42] which is orthogonal to GF in expressive power. GNFO shares the 2ExpTime-completeness of GF, and so does UNFO.

Other known decidable fragments of FO include the *monadic fragment* of FO [27]; the *Maslov class* [31]; the *fluted logic* [40], [37]; the *binding form* systems [32] and generalizations of prefix classes in, e.g., [44]. Moreover, in the highly active field of *description logics* [2], complexities of FO-fragments are classified in great detail, operator by operator.

The current state of the art involves a *huge number* of different logical frameworks, tailored for different purposes. Consequently the related research can be a bit scattered, and could benefit from more systematic approaches. The current article suggests an algebraic framework that enables a possible systematic approach to related studies.

Our contributions. We introduce a research method for classifying complexity and decidability of fragments of FO (and beyond) within an algebraic framework. We believe the setting nicely enables reasonably general and fine-grained studies of related questions. One of the key ideas is to always identify a *finite* collection of operators to capture the expressive power of FO or some other logic of interest – so our algebras have finite signatures. In FO, there are essentially infinitely many quantifiers $\exists x_i$ due to the different variable symbols x_i, and this issue gives rise to the infinite signature of *cylindric set algebras*, which are the principal algebraic formulation of FO. Basing our investigations on finite signatures leads to a nicely controlled setting that elucidates how the expressive power of FO arises. This is achieved by listing a finite set of operators that the expressivity of FO is based on.

The principal system we introduce is built on the algebraic signature $(e, p, s, I, \neg, J, \exists)$. The atomic terms of the related algebra are simply relation symbols (of any arity), and complex algebraic terms are built from atoms by applying the operators in the signature in the usual way. This defines an algebra over every relational structure $\mathfrak{M}$. The atomic terms R are interpreted as the corresponding relations $R^{\mathfrak{M}}$.

The operators correspond to functions that modify relations into new relations over $\mathfrak{M}$ as follows. e is the constant operator denoting the equality relation over $\mathfrak{M}$. p is a cyclic permutation operator while s is a swap operator (swapping the last two elements of tuples). I is a substitution operator, which deletes tuples whose last two members are not identical and then projects away the last coordinate of the remaining tuples (this correspond to variable substitutions in FO). $\neg$ and J are the complementation and join operators, respectively. Finally, $\exists$ is the projection operator (projects the last element away from tuples).

We let $\mathrm{GRA}(e, p, s, I, \neg, J, \exists)$ refer to the system based on these operators, with GRA standing for *general relation algebra*. To simplify notation, we also let GRA stand for $\mathrm{GRA}(e, p, s, I, \neg, J, \exists)$ in the current article. Furthermore, by $\mathrm{GRA} \setminus f_1, \ldots, f_k$ we refer to GRA with the operators $f_1, \ldots, f_k \in \{e, p, s, I, \neg, J, \exists\}$ removed.

We begin our study by proving that GRA *and* FO *are equiexpressive*. The next aim is to classify the decidability and complexity properties of the principal subsystems of GRA. Firstly, $\mathrm{GRA} \setminus \neg$ is trivially decidable, every term being satisfiable. Nevertheless, $\mathrm{GRA} \setminus \neg$ is interesting, as we show it can define precisely all conjunctive queries with equality. Then we establish that $\mathrm{GRA} \setminus \exists$ is NP-complete. We then show that satisfiability of $\mathrm{GRA} \setminus J$ can be checked by a finite automaton, and futhermore, we prove $\mathrm{GRA} \setminus I$ to be NP-complete. We thereby identify new decidable low-complexity fragments of FO. Including

the discovery of the algebraic systems, we identify, e.g., the NP-complete fragment $\mathcal{F}$ of FO based on the restriction that when forming conjunctions $\varphi(x_1,\dots,x_m) \wedge \psi(y_1,\dots,y_n)$, the sets $\{x_1,\dots,x_m\}$ and $\{y_1,\dots,y_n\}$ of variables should be disjoint.

On the negative side, we show that $\mathrm{GRA}(p, I, \neg, J, \exists)$ is Π^0_1-complete, so removing either e or s (or both) from GRA does *not* lead to decidability. Thus we have the following close to complete first classification: removing any of the operators $\neg, \exists, I, J$ gives a decidable system, while dropping e or s (or both) keeps the system undecidable. The only open case concerning the removal of a single operator is $\mathrm{GRA} \setminus p$. We leave the study of the complexity and decidability of subsystems of GRA there in this introductory article.

To push our approach further, we define a general notion of a *relation operator* which puts connectives and (generalized) quantifiers under the same umbrella concept. The definition is a slight generalization of the notion of a generalized quantifier due to Mostowski [34] and Lindström [26]. We then study variants of GRA with different sets of relation operators.

In particular, we characterize the guarded fragment, two-variable logic, fluted logic and unary negation fragment by algebras. The guarded fragment corresponds to the algebra $\mathrm{GRA}(e, p, s, \setminus, \dot{\cap}, \exists)$ where the symbol $\setminus$ denotes relative complementation and $\dot{\cap}$ the *suffix intersection* operator. The suffix intersection is similar to standard intersection but makes sense also when intersecting relations of different arities. Two-variable logic – over vocabularies with at most binary relation symbols – corresponds to $\mathrm{GRA}(e, s, \neg, \dot{\cap}, \exists)$, and fluted logic to $\mathrm{GRA}(\neg, \dot{\cap}, \exists)$. Finally, the unary negation fragment corresponds to $\mathrm{GRA}(e, p, s, \neg_1, J, \overline{J}, \exists)$, where $\neg_1$ denotes 1-*dimensional negation* and $\overline{J}$ the dual operator of J.

Observe that the algebras for fluted logic and two-variable logic are clearly rather intimately related (note that we do not impose restrictions on relation symbol arities for fluted logic). Also, as the guarded fragment is characterized by $\mathrm{GRA}(e, p, s, \setminus, \dot{\cap}, \exists)$, and as $\mathrm{GRA}(e, s, \neg, \dot{\cap}, \exists)$ and $\mathrm{GRA}(e, p, s, \neg, \dot{\cap}, \exists)$ can be shown equiexpressive over vocabularies with at most binary relations, also two-variable logic and the guarded fragment are very nicely and closely linked. These kinds of results demonstrate the explanatory power and potential usefulness of comparing FO-fragments *under the same umbrella framework* based on different kinds of *finite signature algebras*. Indeed, *each finite operator set specifies precisely what the building blocks of the related logic are,* giving a nice, compact expressivity characterization.

The contributions of this article can be summarized by the following four points. **(1)** The main objective is to introduce an algebraic approach for classifying the complexity and decidability properties of logics via different finite signature algebras. **(2)** Concretely, we provide a comprehensive classification of the principal subsystems of $\mathrm{GRA}(e, p, s, I, \neg, J, \exists)$ (which itself characterizes FO). In each solved case, we also pinpoint the complexity of the system. In the process, we also identify interesting new decidable fragments of FO. **(3)** We find algebraic characterizations for FO and some of its main decidable fragments such as FO^2, guarded fragment, fluted logic and unary negation fragment. (Furthermore, we additionally find algebras that characterize conjunctive queries, equality-free FO, quantifier-free FO and the set of first-order atoms.) We also provide a 2ExpTime-completeness result for the algebra for the guarded fragment GF. This turns out to require some nice proof techniques and new notions (e.g., the notion of a *term guard*) to keep the translations between GF and the algebra polynomial. **(4)** We define a general notion of a relation operator.

Further notes on related work. We already extensively surveyed the *related work* concerning our program. We now give further related information on algebraic issues.

There are various algebraic approaches to FO, e.g., Tarski's *cylindric algebras*, their semantic counterparts *cylindric set algebras* and the *polyadic algebras* of Halmos. The book [16] gives a comprehensive and relatively recent account of the these systems. Also,

variants of Codd's *relational algebra* [9] are very important. The main systems studied within that setting relate to domain independent first-order logic. The closest approach to our system is Quine's *predicate functor logic* [38, 39, 41]. This system comes in several variants, with different sets of operators used. Variants of predicate functor logic can be naturally considered to be within the scope of our research program. Predicate functor logic has been studied very little, and we are not aware of any work relating to complexity theory that predates our current work. The notable works within this framework include the complete axiomatizations given in [23] and [3]. Concerning further algebraic settings, Tarski's *relation algebra* (see [16]) is also related to our work, but focuses on binary relations. For some recent results on algebras of relations, see, e.g., [15].

The article [19] is the arXiv-preprint of the current article with full proofs of all results, with the exception of Theorem 6 which will be proved in the final full version. [19] has already been followed-up by [17], where several natural extensions of so-called *ordered logic*, fluted logic and FO^2 were studied within the algebraic framework introduced below. [18] is the first version of the prerint [19] of the current article. The article [18] contains many of the results below, but using a slightly different set of algebraic operators. The research program realized in the current article was proposed in [24]. That article discusses the FO-equivalence of the operators of [18] and suggests, for example, studying systems with limited permutations.

2 Preliminaries

Let A be an arbitrary set. As usual, a ***k*-tuple** over A is an element of A^k. When $k = 0$, we let ϵ denote the unique **0-tuple** in $A^k = A^0$. Note that $A^0 = B^0 = \varnothing^0 = \{\epsilon\}$ for all sets A and B. Note also that $\varnothing^k = \varnothing$ for all positive integers k. If k is a non-negative integer, then a k-ary **AD-relation** over a set A is a pair (R, k) where $R \subseteq A^k$ is a k-ary relation in the usual sense, i.e., simply a set of k-tuples. We call $(\varnothing, k)$ the **empty *k*-ary AD-relation**. For a non-negative integer k, we let $\top_k$ (respectively, $\bot_k$) denote an operator that maps any set A to the AD-relation $\top_k(A) := (A^k, k)$ (respectively, $\bot_k(A) := (\varnothing, k)$). We may write $\top_k^A$ for $\top_k(A)$ and simply $\top_0$ for $\top_0(A)$, and we may write $\bot_k^A$ or $\bot_k$ for $\bot_k(A)$. We note that $\bot_k^{\varnothing} = \top_k^{\varnothing}$ iff $k \neq 0$. When $T = (R, k)$ is a k-ary AD-relation, we let $rel(T)$ denote R and write $ar(T) = k$ to refer to the arity of T. If T is an AD-relation, $t \in T$ *always means that* $t \in rel(T)$.

The notion of a model is defined as usual in model theory, assuming domains are never empty. For simplicity, we restrict attention to relational models, i.e., vocabularies of models do not contain function or constant symbols. The domain of a model $\mathfrak{A}$ is denoted by A, the domain of $\mathfrak{B}$ by B, et cetera. We let $\hat{\tau}$ denote the **full relational vocabulary** containing countably infinitely many relation symbols of each arity $k \geq 0$. We let $\mathrm{VAR} = \{v_1, v_2, \dots\}$ denote the countably infinite set of exactly all variables used in first-order logic FO. We also use metavariables (e.g., $x, y, z, x_1, x_2 \dots$) to refer to symbols in VAR. The syntax of FO is built as usual, starting from the set of atoms consisting of **equality atoms** (i.e., atoms with the equality symbol $=$) and **relation atoms** $R(x_1, \dots, x_n)$ where $R \in \hat{\tau}$. By an FO-formula $\varphi(x_1, \dots, x_k)$ we refer to a formula whose free variables are exactly $x_1, \dots, x_k$. An FO-formula φ (without a list of variables) may or may not have free variables. $Free(\varphi)$ denotes the set of free variables of φ. Now, let $(x_1, \dots, x_k)$ be a tuple of pairwise distinct variables and consider a formula $\varphi(x_1, \dots, x_k)$. Let also $(y_1, \dots, y_k)$ be a tuple of pairwise distinct variables. Then we let $\varphi(y_1, \dots, y_k)$ be the formula obtained from $\varphi(x_1, \dots, x_k)$ by simultaneously replacing each free variable x_i by y_i for all $i \leq k$ (and avoiding variable capture by renaming bounded variables, if necessary).

Let $k \geq 0$ and consider an FO-formula $\varphi(v_{i_1}, \ldots, v_{i_k})$ where $i_1 < \cdots < i_k$. The formula $\varphi(v_{i_1}, \ldots, v_{i_k})$ **defines** the AD-relation $\big(\{(a_1, \ldots, a_k) \in A^k \mid \mathfrak{A} \models \varphi(a_1, \ldots, a_k)\}, k\big)$ in the model $\mathfrak{A}$. Notice that we make crucial use of the linear ordering of the subindices of the variables $v_{i_1}, \ldots, v_{i_k}$ in VAR. We let $\varphi^{\mathfrak{A}}$ denote the AD-relation defined by φ in $\mathfrak{A}$. Notice – to give an example – that $\varphi(v_1, v_2, v_3)$ and $\varphi(v_6, v_8, v_9)$ define the same AD-relation over any model. It is important to recall this phenomenon below. When using the six metavariables x, y, z, u, v, w, we henceforth always assume $(x, y, z, u, v, w) = (v_{i_1}, v_{i_2}, v_{i_3}, v_{i_4}, v_{i_5}, v_{i_6})$ for some indices $i_1 < i_2 < i_3 < i_4 < i_5 < i_6$. Now, to clarify a further technical issue, let us consider the formulas $R(v_1, v_2)$ and $R(v_1, v_1)$. Observe that while $R(v_1, v_2)$ defines a binary AD-relation, the atom $R(v_1, v_1)$ defines a unary one since the only variable in it is v_1.

Consider the formulas $\varphi := v_1 \neq v_1$ and $\psi := v_1 \neq v_1 \wedge v_2 \neq v_2$. Now $\varphi^{\mathfrak{A}}$ is the empty unary AD-relation and $\psi^{\mathfrak{A}}$ the empty binary AD-relation. The negated formulas $\neg\varphi$ and $\neg\psi$ then define the universal unary and binary AD-relations $(\neg\varphi)^{\mathfrak{A}} = (A, 1)$ and $(\neg\psi)^{\mathfrak{A}} = (A \times A, 2)$, respectively. This demonstrates why we consider AD-relations rather than ordinary relations: if φ and ψ both defined the ordinary empty relation $\varnothing$ in $\mathfrak{A}$, then the action of $\neg$ in $\mathfrak{A}$ on the input $\varnothing$ would appear ambiguous.

A **conjunctive query** (CQ) is a *formula* $\exists x_1 \ldots \exists x_k \psi$ where ψ is a conjunction of atoms $R(y_1, \ldots, y_n)$. For example $\exists y \exists z(R(x, y, z) \wedge S(y, z, u, v))$ is a CQ with the free variables x, u, v. A **conjunctive query with equality** (CQE) is like a CQ but also allows equality atoms in addition to relation atoms $R(y_1, \ldots, y_n)$.

3 An algebra for first-order logic

In this section we define an algebra equiexpressive with FO. To this end, consider the algebraic signature $\big(e, p, s, I, \neg, J, \exists\big)$ where e is an algebraically nullary symbol (i.e., a constant), the symbols $p, s, I, \neg, \exists$ have arity one, and J has arity two. Let τ be a vocabulary, i.e., a set of relation symbols. The vocabulary τ defines a set of **terms** (or **τ-terms**) built by starting from the the symbols e and $R \in \tau$ and composing terms by using the symbols $p, s, I, \neg, J, \exists$ in the usual way. Thereby e and each $R \in \tau$ are terms, and if $\mathcal{T}$ and $\mathcal{T}'$ are terms, then so are $p(\mathcal{T})$, $s(\mathcal{T})$, $I(\mathcal{T})$, $\neg(\mathcal{T})$, $J(\mathcal{T}, \mathcal{T}')$, $\exists(\mathcal{T})$. We often leave out brackets when using unary operators and write, for example, IpR instead of $I(p(R))$. Each term $\mathcal{T}$ is associated with an arity $ar(\mathcal{T})$ which, as we will see later on, equals the arity of the AD-relation that $\mathcal{T}$ defines on a model. We define that $ar(R)$ is the arity of the relation symbol R, and we define $ar(e) = 2$; $ar(p\mathcal{T}) = ar(\mathcal{T})$; $ar(s\mathcal{T}) = ar(\mathcal{T})$; $ar(\neg\mathcal{T}) = ar(\mathcal{T})$; $ar(J(\mathcal{T}, \mathcal{T}')) = ar(\mathcal{T}) + ar(\mathcal{T}')$. Finally, for I and $\exists$, we define and $ar(I\mathcal{T}) = ar(\exists\mathcal{T}) = ar(\mathcal{T}) - 1$ if $ar(\mathcal{T}) \geq 1$ and $ar(I\mathcal{T}) = ar(\exists\mathcal{T}) = 0$ when $ar(\mathcal{T}) = 0$.

Given a model $\mathfrak{A}$ of vocabulary τ, each τ-term $\mathcal{T}$ defines an AD-relation $\mathcal{T}^{\mathfrak{A}}$ over A, and the arity of $\mathcal{T}^{\mathfrak{A}}$ will indeed be equal to the arity of $\mathcal{T}$. Consider terms $\mathcal{T}$ and $\mathcal{S}$ and assume we have defined AD-relations $\mathcal{T}^{\mathfrak{A}}$ and $\mathcal{S}^{\mathfrak{A}}$. Then the below conditions hold.

R) Let R be a k-ary relation symbol in τ, so R is a constant term in the algebra. We define $R^{\mathfrak{A}} = \big(\{(a_1, \ldots, a_k) \,|\, \mathfrak{A} \models R(a_1, \ldots, a_k)\}, k\big)$.

e) We define $e^{\mathfrak{A}} = \big(\{(a, a) | \, a \in A\}, 2\big)$. We call e the **equality** constant.

p) If $ar(\mathcal{T}) = k \geq 2$, we define $(p(\mathcal{T}))^{\mathfrak{A}} = \big(\{(a_k, a_1, \ldots, a_{k-1}) \,|\, (a_1, \ldots, a_k) \in \mathcal{T}^{\mathfrak{A}}\}, k\big)$, where $(a_k, a_1, \ldots, a_{k-1})$ is the k-tuple obtained from the k-tuple $(a_1, \ldots, a_k)$ by moving the last element a_k to the beginning of the tuple. If $ar(\mathcal{T})$ is 1 or 0, we define $(p(\mathcal{T}))^{\mathfrak{A}} = \mathcal{T}^{\mathfrak{A}}$. We call p the **cyclic permutation** operator.

s) If $ar(\mathcal{T}) = k \geq 2$, we define $\big(s(\mathcal{T})\big)^{\mathfrak{A}} = \big(\{(a_1, \ldots, a_{k-2}, a_k, a_{k-1}) \,|\, (a_1, \ldots, a_k) \in \mathcal{T}^{\mathfrak{A}}\}, k\big)$, where $(a_1, \ldots, a_{k-2}, a_k, a_{k-1})$ is the k-tuple that is obtained from the k-tuple $(a_1, \ldots, a_k)$ by swapping the last two elements a_{k-1} and a_k but keeping the other elements as they are. If $ar(\mathcal{T})$ is 1 or 0, we define $\big(s(\mathcal{T})\big)^{\mathfrak{A}} = \mathcal{T}^{\mathfrak{A}}$. We call s the **swap** operator.

I) If $ar(\mathcal{T}) = k \geq 2$, we let $\big(I(\mathcal{T})\big)^{\mathfrak{A}}$ be the AD-relation

$\big(\{(a_1, \ldots, a_{k-1}) \,|\, (a_1, \ldots, a_{k-1}, a_k) \in \mathcal{T}^{\mathfrak{A}} \text{ and } a_{k-1} = a_k\}, k-1\big)$.

If $ar(\mathcal{T})$ is 1 or 0, then $\big(I(\mathcal{T})\big)^{\mathfrak{A}} = \mathcal{T}^{\mathfrak{A}}$. We call I the **substitution** operator.

¬) Let $ar(\mathcal{T}) = k$. We define $\big(\neg(\mathcal{T})\big)^{\mathfrak{A}} = \big(\{(a_1, \ldots, a_k) \,|\, (a_1, \ldots, a_k) \in A^k \setminus rel(\mathcal{T}^{\mathfrak{A}})\}, k\big)$. Note, in particular, that if $\mathcal{T}^{\mathfrak{A}} = (\varnothing, 0) = \bot_0^A$, then $\big(\neg(\mathcal{T})\big)^{\mathfrak{A}} = (\{\epsilon\}, 0) = \top_0^A$, and vice versa, if $\mathcal{T}^{\mathfrak{A}} = \top_0^A$, then $\big(\neg(\mathcal{T})\big)^{\mathfrak{A}} = \bot_0^A$. We call ¬ the **complementation** operator.

J) Let $ar(\mathcal{T}) = k$ and $ar(\mathcal{S}) = \ell$. We define $\big(J(\mathcal{T}, \mathcal{S})\big)^{\mathfrak{A}}$ to be the AD-relation

$\big(\{(a_1, \ldots, a_k, b_1, \ldots, b_\ell) \,|\, (a_1, \ldots, a_k) \in \mathcal{T}^{\mathfrak{A}} \text{ and } (b_1, \ldots, b_\ell) \in \mathcal{S}^{\mathfrak{A}}\}, k + \ell\big)$.

Note that ϵ is interpreted as the identity of concatenation, so if $rel(\mathcal{T}^{\mathfrak{A}}) = \{\epsilon\}$, then $\big(J(\mathcal{T}, \mathcal{S})\big)^{\mathfrak{A}} = \big(J(\mathcal{S}, \mathcal{T})\big)^{\mathfrak{A}} = \mathcal{S}^{\mathfrak{A}}$ and $\big(J(\mathcal{T}, \mathcal{T})\big)^{\mathfrak{A}} = (\{\epsilon\}, 0)$. We call J the **join** operator.

∃) If $ar(\mathcal{T}) = k \geq 1$, we let $\big(\exists(\mathcal{T})\big)^{\mathfrak{A}}$ be the AD-relation

$\big(\{(a_1, \ldots, a_{k-1}) \,|\, (a_1, \ldots, a_k) \in \mathcal{T}^{\mathfrak{A}} \text{ for some } a_k \in A\}, k-1\big)$

where $(a_1, \ldots, a_{k-1})$ is the $(k-1)$-tuple obtained by removing the last element of $(a_1, \ldots, a_k)$. When $ar(\mathcal{T}) = 0$, then $\big(\exists(\mathcal{T})\big)^{\mathfrak{A}} = \mathcal{T}^{\mathfrak{A}}$. We call ∃ the **projection** operator.

We denote this algebra by $\mathrm{GRA}(e, p, s, I, \neg, J, \exists)$ where GRA stands for **general relation algebra**. A set $\{f_1, \ldots, f_k\}$ of operators defines the general relation algebra $\mathrm{GRA}(f_1, \ldots, f_k)$. In this paper – only to simplify notation – we write GRA for $\mathrm{GRA}(e, p, s, I, \neg, J, \exists)$. We identify $\mathrm{GRA}(f_1, \ldots, f_k)$ with the set of $\hat{\tau}$-terms of this algebra, where $\hat{\tau}$ is the full relational vocabulary. On the logic side, we similarly identify FO with the set of $\hat{\tau}$-formulas.

Let $\mathcal{G}$ be some set of terms of some general relation algebra $\mathrm{GRA}(f_1, \ldots, f_k)$. Formally, the satisfiability problem for $\mathcal{G}$ takes as input a term $\mathcal{T} \in \mathcal{G}$ and returns '*yes*' iff there exists a model $\mathfrak{A}$ such that $\mathcal{T}^{\mathfrak{A}}$ is not the empty AD-relation of arity $ar(\mathcal{T})$.

An FO-formula φ and term $\mathcal{T}$ are **equivalent** if $\varphi^{\mathfrak{A}} = \mathcal{T}^{\mathfrak{A}}$ for every τ-model $\mathfrak{A}$ (where τ is an arbitrary vocabulary that is large enough so that φ is a τ-formula and $\mathcal{T}$ a τ-term). For example, the formula $R(v_1, v_2)$ is equivalent to R, while $R(v_2, v_1) \wedge (P(v_1) \vee \neg P(v_1))$ is equivalent to sR. Note that under our definition, $R(v_3, v_6)$ and $R(v_1, v_2)$ are both equivalent to the term R while the formulas are not equivalent to each other. This causes no ambiguities as long as we use the terminology carefully. Also, $R(v_1, v_2) \wedge v_3 = v_3$ is *not* equivalent to the term R as it defines a ternary rather than a binary relation. Furthermore, recall that in our setting, the formula $T(v_1, v_1, v_2)$ defines a binary relation and $v_8 = v_8$ a unary relation.

It is useful to remember below how the use of the operator p is reflected to corresponding FO-formulas: if $rel\big(R^{\mathfrak{A}}\big) = \{(a, b, c, d)\} = rel\big((Rxyzu)^{\mathfrak{A}}\big)$, then $rel\big((pR)^{\mathfrak{A}}\big) = \{(d, a, b, c)\} = rel\big((Ryzux)^{\mathfrak{A}}\big)$, so the tuple (a, b, c, d) has its last element moved to the beginning of the tuple, while $Rxyzu$ has the first variable x moved to the end of the tuple of variables. It is also useful to understand how the operator I works. For example, if $rel\big(R^{\mathfrak{A}}\big) = \{(a, b, c, d)\} = rel\big((Rxyzu)^{\mathfrak{A}}\big)$, then $rel\big((IR)^{\mathfrak{A}}\big) = \{(a, b, c)\}$ if $c = d$, and else $rel\big((IR)^{\mathfrak{A}}\big) = \varnothing$. Thus IR is equivalent to $Rxyzz$ which is obtained from $Rxyzu$ by the variable substitution $u \mapsto z$.

Let S_1 be a set of terms of our algebra and S_2 a set of FO-formulas. Then S_1 and S_2 are **equiexpressive** if each term in S_1 has an equivalent formula in S_2 and conversely each formula in S_2 an equivalent term in S_1. The sets S_1 and S_2 are called **sententially equiexpressive** if each *sentence* of S_2 has an equivalent 0-ary term in S_1 and conversely each 0-ary term S_1 has an equivalent sentence in S_2.

▶ **Theorem 1.** FO *and* GRA *are equiexpressive.*

Proof. Translating terms to FO formulas is straightforward. Suppose then that $\varphi \in$ FO. Consider first the cases where φ is one of the atoms $x = x$, $x = y$. Then the corresponding terms are, respectively, Ie and e. Assume then that φ is $R(v_{i_1}, \ldots, v_{i_k})$ for $k \geq 0$. Suppose first that no variable symbol gets repeated in the tuple $(v_{i_1}, \ldots, v_{i_k})$ and that $i_1 < \cdots < i_k$. Then the term R is equivalent to φ. We then consider the cases where $(v_{i_1}, \ldots, v_{i_k})$ may have repetitions and the variables may not be linearly ordered (i.e., $i_1 < \cdots < i_k$ does not necessarily hold). We can permute any relation in every possible way by using the operators p and s; for the sake of completeness, we present the following steps that prove this claim.

Consider a tuple $(a_1, \ldots, a_i, \ldots, a_k)$ of the relation $R^{\mathfrak{A}}$ in a model $\mathfrak{A}$. Now, we can move the element a_i an arbitrary number n of steps to the left (while keeping the rest of the tuple otherwise in the same order) by doing the following: **(1)** repeatedly apply p to the term R, making a_i the rightmost element of the tuple; **(2)** apply then the *composed* function ps (so s first and then p) precisely n times; **(3)** Apply p repeatedly to put the tuple into the ultimate desired order. Moving a_i to the right is similar. Intuitively, we keep moving a_i to the *left* and continue even when it has gone past the leftmost element of the original tuple. Formally, we can move a_i by n steps to the right by performing the above three steps so that in step 2, we apply the composed function ps exactly $k - n - 1$ times.

This shows that we can move an arbitrary element anywhere in the tuple, and thereby it is clear that with p and s, we can permute a relation in all possible ways. Since we indeed can permute tuples without restrictions, we can also deal with the possible repetitions of variables in $R(v_{i_1}, \ldots, v_{i_k})$. Indeed, we can bring any two elements to the right end of a tuple and then use I. We discussed this phenomenon already above, but for extra clarity, we once more illustrate the issue by providing a related, concrete example. So let us consider the formula $R(v_1, v_2, v_1)$ (which defines a *binary* relation). We observe that $R(v_1, v_2, v_1)$ is equivalent to the term $pIpp(R)$, so we first use p twice to permute R, then we use I to identify coordinates, and finally we use p once more.

So, to sum up, we permute tuples by p and s and we use I for identifying variables. Thus, using p, s, I, we can find an equivalent term for every quantifier-free formula $R(v_{i_1}, \ldots, v_{i_k})$.

Now suppose we have equivalent terms $\mathcal{S}$ and $\mathcal{T}$ for formulas φ and ψ, respectively. We will discuss how to translate $\neg\varphi$, $\varphi \wedge \psi$ and $\exists v_i \varphi$. Firstly, clearly $\neg\varphi$ can be translated to $\neg\mathcal{S}$. Translating $\varphi \wedge \psi$ is done in two steps. Suppose φ and ψ have, respectively, the free variables $v_{i_1}, \ldots, v_{i_k}$ and $v_{j_1}, \ldots, v_{j_\ell}$. We first write the term $J(\mathcal{S}, \mathcal{T})$ which is equivalent to $\chi(v_1, \ldots, v_{k+\ell}) := \varphi(v_1, \ldots, v_k) \wedge \psi(v_{k+1}, \ldots, v_{k+\ell})$; note here the new lists of variables. We then deal with the possible overlap in the original sets $\{v_{i_l}, \ldots, v_{i_k}\}$ and $\{v_{j_l}, \ldots, v_{j_\ell}\}$ of variables of φ and ψ. This is done by repeatedly applying p, s and I to $J(\mathcal{S}, \mathcal{T})$ in the same way as used above when dealing with atomic formulas. Indeed, we above observed that we can arbitrarily permute relations and identify variables by using p, s, I. Finally, translating $\exists v_i \varphi$ is easy. We first repeatedly apply p to the term $\mathcal{S}$ corresponding to φ to bring the element to be projected away to the right end of each tuple. Then we use $\exists$. After this we again use p repeatedly to put the term into the final wanted form. ◀

The following corollary is now easy to extract from the above proof. It gives a nice, *algebraic characterization of atomic formulas* of FO.

▶ **Corollary 2.** GRA(p, s, I) *is equiexpressive with the set of relation atoms of* FO*, and* GRA(e, p, s, I) *is equiexpressive with the set of all atoms of* FO*.*

4 Relation operators and fragments of first-order logic

The FO-equivalent algebra $\mathrm{GRA} = \mathrm{GRA}(e, p, s, I, \neg, J, \exists)$ is *only one of many interesting related systems*. Defining alternative algebras equiexpressive with FO is one option, but it is also interesting to consider weaker, stronger and orthogonal systems. We next give a general definition that enables classifying all such algebras in a systematic way. In the definition, AD_A is the set of all AD-relations (of every arity) over a set A. If $T_1, \ldots, T_k$ are AD-relations over A, then $(A, T_1, \ldots, T_k)$ is called an **AD-structure**. A bijection $g : A \to B$ is an isomorphism between AD-structures $(A, T_1, \ldots, T_k)$ and $(B, S_1, \ldots, S_k)$ if $ar(T_i) = ar(S_i)$ for each i and g is an ordinary isomorphism between $(A, rel(T_1), \ldots, rel(T_k))$ and $(B, rel(S_1), \ldots, rel(S_k))$.

▶ **Definition 3.** *A k-ary **relation operator** f is a map that outputs, given an arbitrary set A, a k-ary function $f^A : (\mathrm{AD}_A)^k \to \mathrm{AD}_A$. The operator f is isomorphism invariant in the sense that if the* AD*-structures $(A, T_1, \ldots, T_k)$ and $(B, S_1, \ldots, S_k)$ are isomorphic via $g : A \to B$, then $(A, f^A(T_1, \ldots, T_k))$ and $(B, f^B(S_1, \ldots, S_k))$ are, likewise, isomorphic via g.*

An **arity-regular relation operator** is a relation operator with the property that the arity of the output AD-relation depends only on the sequence of arities of the input AD-relations.

To illustrate the notion of a relation operator, let us consider some concrete examples. Suppose $\mathcal{T}$ and $\mathcal{S}$ are both of arity k. We define $(\mathcal{T} \cup \mathcal{S})^{\mathfrak{A}} = (rel(\mathcal{T}^{\mathfrak{A}}) \cup rel(\mathcal{S}^{\mathfrak{A}}), k)$, $(\mathcal{T} \cap \mathcal{S})^{\mathfrak{A}} = (rel(\mathcal{T}^{\mathfrak{A}}) \cap rel(\mathcal{S}^{\mathfrak{A}}), k)$, and $(\mathcal{T} \setminus \mathcal{S})^{\mathfrak{A}} = (rel(\mathcal{T}^{\mathfrak{A}}) \setminus rel(\mathcal{S}^{\mathfrak{A}}), k)$. And if $\mathcal{T}$ and $\mathcal{S}$ have different arities, then $\cap$ and $\cup$ return $(\varnothing, 0)$ and $\setminus$ returns $\mathcal{T}^{\mathfrak{A}}$. Suppose then that $\mathcal{T}$ and $\mathcal{S}$ have arities k and ℓ, respectively. Calling $m := max\{k, \ell\}$, we let

$$(\mathcal{T} \dot{\cap} \mathcal{S})^{\mathfrak{A}} = \big(\{(a_1, \ldots, a_m) \mid (a_{m-k+1}, \ldots, a_m) \in \mathcal{T}^{\mathfrak{A}} \text{ and } (a_{m-\ell+1}, \ldots, a_m) \in \mathcal{S}^{\mathfrak{A}}\},\ m\big),$$

so intuitively, the tuples overlap on some suffix of $(a_1, \ldots, a_m)$ (note that when k or ℓ is zero, then (a_{m+1}, a_m) denotes the 0-tuple ϵ). We call $\dot{\cap}$ the **suffix intersection**.

In the next section we prove that the guarded fragment GF is sententially equivalent to $\mathrm{GRA}(e, p, s, \setminus, \dot{\cap}, \exists)$. We note that in [25, 16], the authors define Codd-style relational algebra systems (with inherently infinite signatures) and they then prove their systems to be sententially equiexpressive with GF. The system in [25] uses, e.g., a *semijoin operator*, which is a join operation but employs also a conjunction of identity atoms as part of the input to it. The algebra of [16] employs, e.g., an essentially ternary join operator where the extra input essentially acts as an atomic guard. Both systems have an implicit access to variables via the infinite signatures and extra features, e.g., extra inputs.

The proofs of the characterizations in [25, 16] differ considerably from our corresponding argument, the translations from algebra to logic being inherently *exponential* in [25] and [16]. We carefully develop techniques that allow us to give a *polynomial* translation from $\mathrm{GRA}(e, p, s, \setminus, \dot{\cap}, \exists)$ to GF, which in turn allows us to prove a 2ExpTime upper bound for the satisfiability problem of $\mathrm{GRA}(e, p, s, \setminus, \dot{\cap}, \exists)$, the same as that for GF. Since we will also give a polynomial translation from GF to $\mathrm{GRA}(e, p, s, \setminus, \dot{\cap}, \exists)$, it follows that the satisfiability problem for the algebra is 2ExpTime-complete. Thus $\mathrm{GRA}(e, p, s, \setminus, \dot{\cap}, \exists)$ is the first algebra for GF for which the same 2ExpTime complexity has been proved.

We next give a characterization for two-variable logic. Technically, the characterization has some similarities with, e.g., the modal-logic-based characterization in [28].

▶ **Theorem 4.** FO^2 *and* $\mathrm{GRA}(e, s, \neg, \dot{\cap}, \exists)$ *are sententially equiexpressive over vocabularies with at most binary relations.*

Proof. The algebra $\mathrm{GRA}(e, s, \neg, \dot{\cap}, \exists)$ with at most binary relation symbols clearly contains only terms of arity at most two. Thus it is easy to translate the terms into FO^2.

We then consider the converse translation. We assume that FO^2 is built using $\neg, \wedge$ and $\exists$ and treat other connectives and $\forall$ as abbreviations in the usual way.

Now, let $\varphi \in \mathrm{FO}^2$ be a *sentence* with at most binary relations, and let x and y be the two variables that occur in φ. Note indeed that φ is a sentence, not an open formula. We first convert φ into a sentence that does not contain any subformulas of type $\psi(x) \wedge \chi(y)$ (or of type $\psi(x) \vee \chi(y)$) as follows. Consider any subformula $\exists x\, \eta(x, y)$ where $\eta(x, y)$ is quantifier-free. Put η into disjunctive normal form and distribute $\exists x$ over the disjunctions. Then distribute $\exists x$ also over the conjunctions as follows. Consider a conjunction $\alpha_i(x, y) \wedge \beta_i(y) \wedge \gamma_i$ where each of α_i, β_i, γ_i are conjunctions of literals; the formula γ_i contains the nullary relation symbols and $\alpha_i(x, y)$ contains the literals of type $\pi(x, y)$ and $\pi'(x)$. We distribute $\exists x$ into $\alpha_i(x, y) \wedge \beta_i(y) \wedge \gamma_i$ so that we obtain the formula $\exists x \alpha_i(x, y) \wedge \beta_i(y) \wedge \gamma_i$. Thereby the formula $\exists x\, \eta(x, y)$ gets modified into the formula $\bigvee_{i=1}^{n} (\exists x \alpha_i(x, y) \wedge \beta_i(y) \wedge \gamma_i)$ which is of the right form and does not have x as a free variable. Next we can repeat this process for other existential quantifiers in the formula (by treating the subformulas with one free variable in the way that atoms with one free variable were treated in the above translation step for $\eta(x, y)$). Having started from the *sentence* φ, we ultimately get a sentence equivalent to φ but having no subformulas of the form $\psi(x) \wedge \chi(y)$ or of the form $\psi(x) \vee \chi(y)$.

Next we translate an arbitrary sentence $\varphi \in \mathrm{FO}^2$ that satisfies the above condition to an equivalent term. We let $v \in \{x, y\}$ denote a generic variable.

Atoms of the form $P(v)$ (respectively $v = v$) translate to P (respectively $\exists e$). Relation symbols of arity 0 translate to themselves. $R(x, y)$ translates to R and $R(y, x)$ translates to sR. And $R(v, v)$ translates to $\exists(R \mathbin{\dot\cap} e)$ and the atoms $x = y$ and $y = x$ translate to e.

Now suppose we have translated ψ to $\mathcal{T}$. Then $\neg\psi$ translates to $\neg\mathcal{T}$. If ψ has one free variable v, then $\exists v\psi$ translates to $\exists\mathcal{T}$. If ψ has two free variables, then we either translate $\exists v\psi$ to $\exists\mathcal{T}$ when v is y and to $\exists s\mathcal{T}$ when v is x.

Consider now a formula $\psi \wedge \chi$ and suppose that we have translated ψ to $\mathcal{T}$ and χ to $\mathcal{S}$. If at least one of ψ and χ is a sentence, we translate $\psi \wedge \chi$ to $(\mathcal{T} \mathbin{\dot\cap} \mathcal{S})$. Otherwise, due to the form of the sentence φ to be translated, we have $\mathrm{Free}(\psi) \cap \mathrm{Free}(\chi) \neq \varnothing$. Now $\psi(x, y) \wedge \chi(x, y)$, $\psi(y) \wedge \chi(x, y)$ and $\psi(x, y) \wedge \chi(y)$ are all translated to $\mathcal{T} \mathbin{\dot\cap} \mathcal{S}$, while $\psi(x, y) \wedge \chi(x)$ and $\psi(x) \wedge \chi(x, y)$ are translated to $s(s\mathcal{T} \mathbin{\dot\cap} \mathcal{S})$ and $s(\mathcal{T} \mathbin{\dot\cap} s\mathcal{S})$, respectively. ◀

We note that limiting our algebraic characterizations of GF and FO^2 to *sentential* equiexpressivity is a choice based on the relative elegance of the results. Sentential equiexpressivity suffices for the almost all practical scenarios.

Now, let $\mathrm{GRA}_2(e, s, \neg, \dot\cap, \exists)$ denote the terms of $\mathrm{GRA}(e, s, \neg, \dot\cap, \exists)$ that use at most binary relation symbols; there are no restrictions on term arity, although clearly at most binary terms arise. The proof of Theorem 4 gives a translation of FO^2-sentences to $\mathrm{GRA}_2(e, s, \neg, \dot\cap, \exists)$. However, the translation is not polynomial, so we do not immediately get a NexpTime lower bound for the satisfiability of $\mathrm{GRA}_2(e, s, \neg, \dot\cap, \exists)$. Nevertheless, we can prove the following.

▶ **Theorem 5.** *The satisfiability problem of* $\mathrm{GRA}_2(e, s, \neg, \dot\cap, \exists)$ *is NexpTime-complete.*

We then briefly consider fluted logic (FL) and the unary negation fragment (UNFO). The logic FL is a decidable fragment of FO that has recently received increased attention in the research on first-order fragments, see. e.g., [17, 37]. Now, it is straightforward to show that fluted logic is equiexpressive with $\mathrm{GRA}(\neg, \dot\cap, \exists)$. The logic UNFO is a well-established decidable fragment of FO that enjoys many of the desirable properties that modal logics have [42]. Roughly speaking, its syntax is obtained from that of FO by restricting the use of

negation only to formulas that have at most one free variable. To characterize UNFO, we will need to introduce two additional relation operators. Suppose that $\mathcal{T}$ and $\mathcal{S}$ are terms of arity k AND ℓ respectively. We define

$(\overline{J}(\mathcal{T},\mathcal{S}))^{\mathfrak{A}} = (\{(a_1,\ldots,a_k,b_1,\ldots,b_\ell) \mid (a_1,\ldots,a_k) \in \mathcal{T}^{\mathfrak{A}} \text{ or } (b_1,\ldots,b_\ell) \in \mathcal{S}^{\mathfrak{A}}\}, k+\ell)$. Thus $\overline{J}$ is the *dual* of J. If $k \leq 1$, we define $(\neg_1(\mathcal{T}))^{\mathfrak{A}} = (\neg(\mathcal{T}))^{\mathfrak{A}}$, and otherwise $(\neg_1(\mathcal{T}))^{\mathfrak{A}} = \bot_0$. We call $\neg_1$ the **one-dimensional negation**. It can be shown that UNFO is equiexpressive with $\mathrm{GRA}(e,p,s,I,\neg_1,J,\overline{J},\exists)$.

By comparing the algebraic characterizations, we observe FO^2 and fluted logic are very interestingly and intimately related, and the full system $\mathrm{GRA}(e,s,\neg,\dot{\cap},\exists)$ obviously contains both fluted logic and FO^2. Note also the close relationship of these systems to the algebra $\mathrm{GRA}(e,p,s,\backslash,\dot{\cap},\exists)$ for GF. These connections demonstrate how the algebraic approach can nicely elucidate the relationships between seemingly different kinds of fragments of FO. Indeed, FO^2, FL and GF seem much more closely related than one might first suspect.

Another advantage of our algebraic approach is that it naturally suggests new fragments, as one can select their favourite operators and consider the resulting algebras. Inspired by the present work, in [17] this idea was put into action and various interesting extensions of ordered and fluted logic were studied. Here we point out yet another natural fragment inspired by our algebraic approach, namely the algebra $\mathrm{GRA}(e,s,\backslash,\dot{\cap},\exists)$. This algebra is interesting since, e.g., it contains the guarded FO^2 and guarded FL on the level of sentences.

▶ **Theorem 6.** *The satisfiability problem of* $\mathrm{GRA}(e,s,\backslash,\dot{\cap},\exists)$ *is* ExpTime*-complete.*

The proof of this theorem is quite involved, but it follows a well-known path: we design a polynomial space alternating Turing machine which tries to construct a tree-like model for the input term. The key point here is that since permutations are heavily restricted in $\mathrm{GRA}(e,s,\backslash,\dot{\cap},\exists)$, the "nodes" of the tree-like model have polynomial size. The proof will be presented in the final full version. Note that it is straightforward to read a first-order syntax for this system from the algebra. It essentially mixes the ideas behind FO^2 and the guarded variant of fluted logic.

5 An algebra for the guarded fragment

In this section we consider $\mathrm{GRA}(e,p,s,\backslash,\dot{\cap},\exists)$ and show that it is sententially equiexpressive with GF. Recall that GF is the logic that has all atoms $R(x_1,\ldots,x_k)$, $x=y$ and $x=x$, is closed under $\neg$ and $\wedge$, but existential quantification is restricted to patterns $\exists x_1 \ldots \exists x_k(\alpha \wedge \psi)$ where α is an atomic formula (a guard) having (at least) all the free variables of $\psi \in \mathrm{GF}$.

We start by defining the notion of a **term guard** of a term $\mathcal{T}$. Term guards are a central concept in our proof. The term guard of a term $\mathcal{T}$ of $\mathrm{GRA}(e,p,s,\backslash,\dot{\cap},\exists)$ is a tuple $(\mathcal{S},(i_1,\ldots,i_k))$, where $\mathcal{S} \in \mathrm{GRA}(e)$ and $k = ar(\mathcal{T}) \leq ar(\mathcal{S})$, with the following properties.

1. The tuple $(i_1,\ldots,i_k)$ consists of pairwise distinct integers i_j such that $1 \leq i_j \leq ar(\mathcal{S})$.
2. For every model $\mathfrak{A}$ and every tuple $(a_1,\ldots,a_k) \in \mathcal{T}^{\mathfrak{A}}$, there exists a tuple $(b_1,\ldots,b_{ar(\mathcal{S})}) \in \mathcal{S}^{\mathfrak{A}}$ such that $(a_1,\ldots,a_k) = (b_{i_1},\ldots,b_{i_k})$.

The intuition is that the term guard $(\mathcal{S},(i_1,\ldots,i_k))$ of $\mathcal{T}$ gives an atomic term $\mathcal{S}$ and a list $(i_1,\ldots,i_k)$ of coordinate positions (of tuples of $\mathcal{S}^{\mathfrak{A}}$) that guard the tuples of $\mathcal{T}^{\mathfrak{A}}$. The remaining $ar(\mathcal{S})-k$ coordinate positions of the tuples of $\mathcal{S}^{\mathfrak{A}}$ are intuitively non-guarding. The following lemma will be used below when translating algebraic terms to GF.

▶ **Lemma 7.** *Every term* $\mathcal{T} \in \mathrm{GRA}(e,p,s,\backslash,\dot{\cap},\exists)$ *has a term guard. Furthermore the term guard can be computed from* $\mathcal{T}$ *in polynomial time.*

Proof. We will define inductively a mapping which maps each term $\mathcal{T}$ of the system $\mathrm{GRA}(e, p, s, \setminus, \dot{\cap}, \exists)$ to a term guard for $\mathcal{T}$. We start by defining that e will be mapped to $(e, (1, 2))$ and that every relational symbol R will be mapped to $(R, (1, \ldots, ar(R)))$.

Suppose then that we have mapped a k-ary term $\mathcal{T}$ to the term guard $(\mathcal{S}, (i_1, \ldots, i_k))$. Using the term guard $(\mathcal{S}, (i_1, \ldots, i_k))$ as a starting point, we will construct term guards for the terms $p\mathcal{T}$ and $s\mathcal{T}$. Firstly, term guard for $p\mathcal{T}$ will be $(\mathcal{S}, (i_k, i_1, \ldots, i_{k-1}))$, where we have simply permuted the tuple $(i_1, \ldots, i_k)$ with p. (Note that if $k \leq 1$, then the permuted tuple is the same as the original tuple, as p leaves tuples of length up to 1 untouched.) Similarly, the term guard for $s\mathcal{T}$ will be $(\mathcal{S}, (i_1, \ldots, i_{k-2}, i_k, i_{k-1}))$, where this time we have permuted the tuple $(i_1, \ldots, i_k)$ with s. (Again if $k \leq 1$, the permuted tuple is the original tuple.)

The other cases are similar. Recall the assumption that we have mapped a k-ary term $\mathcal{T}$ to the term guard $(\mathcal{S}, (i_1, \ldots, i_k))$, and suppose further that an ℓ-ary term $\mathcal{P}$ has been mapped to the term guard $(\mathcal{S}', (j_1, \ldots, j_\ell))$. If $k \geq \ell$, then $\mathcal{T} \dot{\cap} \mathcal{P}$ will be mapped to $(\mathcal{S}, (i_1, \ldots, i_k))$, and if $k < \ell$, then $\mathcal{T} \dot{\cap} \mathcal{P}$ will be mapped to $(\mathcal{S}', (j_1, \ldots, j_\ell))$. Independently of how the arities k and ℓ are related, $\mathcal{T} \setminus \mathcal{P}$ will always be mapped to $(\mathcal{S}, (i_1, \ldots, i_k))$. (Recall that if the arities of the terms $\mathcal{Q}$ and $\mathcal{R}$ differ, then by definition $\mathcal{Q} \setminus \mathcal{R}$ is equivalent to $\mathcal{Q}$). If $k \geq 1$, the term $\exists\mathcal{T}$ will be mapped to $(\mathcal{S}, (i_1, \ldots, i_{k-1}))$. If $k = 0$, $\exists\mathcal{T}$ maps to the same term guard as $\mathcal{T}$.

Since this mapping is clearly computable in polynomial time, the claim follows. ◀

▶ **Theorem 8.** $\mathrm{GRA}(e, p, s, \setminus, \dot{\cap}, \exists)$ *and* GF *are sententially equiexpressive.*

Proof. We will first show that for every formula $\exists x_1 \ldots \exists x_k \, \psi$ of GF, there is an equivalent term of $\mathrm{GRA}(e, p, s, \setminus, \dot{\cap}, \exists)$. We begin our (ultimately inductive) argument by first showing this for a formula $\varphi := \exists x_1 \ldots \exists x_k \, \psi$ where ψ is quantifier-free. We assume $\varphi = \exists x_1 \ldots \exists x_k (\alpha(y_1, \ldots, y_n) \wedge \beta(z_1, \ldots, z_m))$ where $\alpha(y_1, \ldots, y_n)$ is an atom and $\{z_1, \ldots, z_m\} \subseteq \{y_1, \ldots, y_n\}$ and $\{x_1, \ldots, x_k\} \subseteq \{y_1, \ldots, y_n\}$.

Now consider a conjunction $\alpha \wedge \rho$ where $\alpha = \alpha(y_1, \ldots, y_n)$ is our guard atom and ρ an arbitrary atom whose set of variables is a subset of $\{y_1, \ldots, y_n\}$. We call such a conjunction an *α-guarded atom*. For each α-guarded atom, we can find an equivalent term as follows. First, by Corollary 2, we can write a term equivalent to any atomic FO-formula using e, p, s, I; note that we can use I in $\mathrm{GRA}(e, p, s, \setminus, \dot{\cap}, \exists)$, since if $ar(\mathcal{T}) > 1$, then $I\mathcal{T}$ is equivalent to $\exists(\mathcal{T} \dot{\cap} e)$, and if $ar(\mathcal{T}) \leq 1$, then $I\mathcal{T}$ is equivalent to $\mathcal{T}$. Therefore we can find terms $\mathcal{T}_\alpha$ and $\mathcal{T}_\rho$ equivalent to α and ρ, respectively. Now, the term $\mathcal{T}_\alpha \dot{\cap} \mathcal{T}_\rho$ is not likely to be equivalent to $\alpha \wedge \rho$, as the variables in $\alpha \wedge \rho$ can be unfavourably ordered instead of matching each other nicely. However – recalling that p and s can be composed to produce arbitrary permutations – we first permute $\mathcal{T}_\alpha$ to match $\mathcal{T}_\rho$ at the last coordinates of tuples, then we combine the terms with $\dot{\cap}$, and finally we permute the obtained term to the final desired form. In this fashion we obtain a term for an arbitrary α-guarded atom.

Now recall the formula $\alpha(y_1, \ldots, y_n) \wedge \beta(z_1, \ldots, z_m)$ from above. For each atom γ in β, let $\mathcal{T}_\gamma^\alpha$ denote the term equivalent to the α-guarded atom $\alpha \wedge \gamma$ formed from γ. The formula β is a Boolean combination composed from atoms by using $\neg$ and $\wedge$. We let $\mathcal{T}_\beta$ denote the term obtained from β by replacing each atom γ by the term $\mathcal{T}_\gamma^\alpha$, each $\wedge$ by $\dot{\cap}$ and each $\neg$ by relative complementation with respect to $\mathcal{T}_\alpha$ (i.e., formulas $\neg\eta$ become replaced by $\mathcal{T}_\alpha \setminus \eta^*$ where η^* is the translation of the formula η). It is easy to show that $\mathcal{T}_\beta$ is equivalent to $\alpha(y_1, \ldots, y_n) \wedge \beta(z_1, \ldots, z_m)$. Thus we can clearly apply p and $\exists$ in a suitable way to the term $\mathcal{T}_\beta$ to get a term equivalent to the formula $\varphi = \exists x_1 \ldots \exists x_k (\alpha(y_1, \ldots, y_n) \wedge \beta(z_1, \ldots, z_m))$.

Thus we managed to translate φ. To get the full translation, we mainly just keep repeating the procedure just described. The only difference is that above the formula $\beta(z_1, \ldots, z_m)$ was a Boolean combination of atoms, while now β will also contain formulas of the form

$\exists x_1 \dots \exists x_r(\delta \wedge \eta)$ in addition to atoms. Proceeding by induction, we get a term equivalent to $\exists x_1 \dots \exists x_r(\delta \wedge \eta)$ by the induction hypothesis, and otherwise we proceed exactly as described above. This concludes the argument for translating formulas to terms.

Let us then translate terms into equivalent formulas of GF. We proceed by induction. As GF is closed under Boolean operators, the only non-trivial case is the translation of $\exists$. The hard part in this case is to ensure that we can translate $\exists$ so that the resulting formula has a suitable guarding pattern (with a suitable guard atom) and thereby belongs to GF.

So suppose that we have translated $\mathcal{T}$ to $\psi(v_1, \dots, v_k)$. By Lemma 7, we can find a term guard $(\mathcal{S}, (i_1, \dots, i_k))$ for $\mathcal{T}$. By the definition of term guards, $\mathcal{S}$ is e or some relation symbol R. We let m denote the arity of $\mathcal{S}$, and we let $\alpha(v_1, \dots, v_m)$ denote $R(v_1, \dots, v_m)$, if $\mathcal{S}$ is a relation symbol, and $v_1 = v_2$ if $\mathcal{S}$ is e. Notice that $\{i_1, \dots, i_k\} \subseteq \{1, \dots, m\}$ by the definition of term guards. Now define $\chi(v_{i_1}, \dots, v_{i_k}) := \exists \overline{v}(\alpha(v_1, \dots, v_m) \wedge \psi(v_{i_1}, \dots, v_{i_k}))$, where $\overline{v}$ lists those variables from $\{v_1, \dots, v_m\}$ that are *not* included in $(v_{i_1}, \dots, v_{i_k})$. Modifying $\chi(v_{i_1}, \dots, v_{i_k})$ to the formula $\chi(v_1, \dots, v_k)$ and recalling the definition of term guards, we now observe that $\exists \mathcal{T}$ is equivalent to $\exists v_k \chi(v_1, \dots, v_k)$ which is a GF-formula. ◀

As GF is 2ExpTime-complete and the above translations polynomial, the following holds.

▶ **Corollary 9.** *The satisfiability problem for* $\mathrm{GRA}(e, p, s, \setminus, \dot{\cap}, \exists)$ *is 2ExpTime-complete.*

6 Decidable fragments of GRA

In this section we identify subsystems of $\mathrm{GRA} = \mathrm{GRA}(e, p, s, I, \neg, J, \exists)$ with a decidable satisfiability problem. We concentrate on systems obtained by limiting to a subset of the operators involved. We show that removing any of the operators $\neg, \exists, J, I$ leads to decidability, and we also pinpoint the exact complexity of each system. As a by-product, we make observations about conjunctive queries (CQs) and show NP-completeness of, e.g., $\mathrm{GRA}(\neg, J, \exists)$ and $\mathrm{GRA}(I, \neg, J)$. We also give a characterization for quantifier-free FO.

Our first result concerns GRA with the complementation operation $\neg$ removed. All negation-free fragments of FO are trivially decidable – every formula being satisfiable – and thus so is $\mathrm{GRA}(e, p, s, I, J, \exists)$. Nevertheless, this system has the following very interesting property concerning conjunctive queries with equality, or CQEs.

▶ **Proposition 10.** $\mathrm{GRA}(e, p, s, I, J, \exists)$ *is equiexpressive with the set of* CQE*s. Also, the system* $\mathrm{GRA}(p, s, I, J, \exists)$ *is equiexpressive with the set of conjunctive queries* (CQ*s*).

Proof. Analyzing the proof that $\mathrm{GRA}(e, p, s, I, \neg, J, \exists)$ and FO are equiexpressive, we see that $\mathrm{GRA}(e, p, s, I, J, \exists)$ can express every formula built from relational atoms and equality atoms with conjunctions and existential quantification. Conversely, an easy induction on term structure establishes that every term of the system $\mathrm{GRA}(e, p, s, I, J, \exists)$ is expressible by a CQE. The claim for $\mathrm{GRA}(p, s, I, J, \exists)$ follows similarly, noting that e is used only to express the atoms $x = x$ and $x = y$ in the proof of Theorem 1. ◀

We then consider GRA without $\exists$.

▶ **Proposition 11.** $\mathrm{GRA}(e, p, s, I, \neg, J)$ *is equiexpressive with quantifier-free FO, and the satisfiability problem for* $\mathrm{GRA}(e, p, s, I, \neg, J)$ *is NP-complete.*

(We can sharpen the lower bound by showing that already $\mathrm{GRA}(I, \neg, J)$ is NP-complete.) We then consider the join-free fragment of GRA. It turns out to be interestingly tame, with a very low complexity:

▶ **Theorem 12.** *Satisfiability of* $\mathrm{GRA}(e, p, s, I, \neg, \exists)$ *can be checked by a finite automaton.*

We then study GRA without I and show it NP-complete. We begin by identifying a new decidable fragment $\mathcal{F}$ of FO. The new logic $\mathcal{F}$ turns out to be an interesting, low-complexity fragment of FO, as we will prove it NP-complete. The fragment $\mathcal{F}$ is defined as the set of formulas φ of FO which satisfy the following condition: if $(\psi \wedge \chi)$ is a subformula of φ, then $\mathrm{Free}(\psi) \cap \mathrm{Free}(\chi) = \varnothing$ (note here that disjunction is not treated as a primitive operator; the primitive operators of $\mathcal{F}$ are $\neg$, $\wedge$, $\exists$).

We will then establish NP-completeness of $\mathrm{GRA}\backslash I$. We will first show that the satisfiability problem of $\mathcal{F}$ is complete for NP. The upper bound is based on a non-deterministic reduction to the satisfiability problem of the set of *relational Herbrand sentences*. We note that checking satisfiability of equality-free relational Herbrand sentences is known to be PTime-complete, see Theorem 8.2.6 in [7]. However, there seems to be *no explicit proof* of the PTime-completeness of case with equality in the literature, so we provide it in [19].

▶ **Theorem 13.** *The satisfiability problem of* $\mathcal{F}$ *is NP-complete.*

Proof. For the lower bound, suppose φ is a formula of propositional logic. We obtain an equisatisfiable formula of $\mathcal{F}$ by replacing each proposition symbol p_i by the sentence $\forall x P_i(x)$.

We thus consider the upper bound. Let $\chi \in \mathcal{F}$ be a formula. First we transform χ into negation normal form, thus obtaining a formula χ'. Now note that in $\mathcal{F}$, the formula $\forall x(\varphi \vee \psi)$ is equivalent to either $(\varphi \vee \psi)$, $(\forall x\varphi \vee \psi)$ or $(\varphi \vee \forall x\psi)$ since $Free(\varphi) \cap Free(\psi) = \varnothing$. Similarly, $\exists x(\varphi \wedge \psi)$ is equivalent to $(\varphi \wedge \psi)$, $(\exists x\varphi \wedge \psi)$ or $(\varphi \wedge \exists x\psi)$. Thus we can push all quantifiers past all connectives in the formula χ' in polynomial time, getting a formula χ''.

Let C be the set of all conjunctions obtained from χ'' as follows: begin from the syntax tree of χ'' and keep eliminating disjunctions $\vee$, always keeping one of the two disjuncts. Now χ'' is satisfiable iff some $\beta \in C$ is satisfiable. Starting from χ'', we nondeterministically guess some $\beta \in C$ (without constructing C). Now, β is a conjunction of formulas $Q_1 x_1 \dots Q_k x_k\, \eta$ where $Q_i \in \{\forall, \exists\}$ for each i and η is a literal. Putting β in prenex normal form, we get a relational Herbrand sentence. In [19] we show that satisfiability of relational Herbrand sentences is PTime-complete. ◀

Using Theorem 13, we then determine the exact complexity of $\mathrm{GRA}\backslash I$.

▶ **Theorem 14.** *The satisfiability problem of* $\mathrm{GRA}(e, p, s, \neg, J, \exists)$ *is NP-complete.*

Proof. We shall first show that $\mathrm{GRA}(e, p, s, \neg, J, \exists)$-terms translate to equisatisfiable formulas of $\mathcal{F}$ in polytime: the upper bound of the current theorem then follows due to Theorem 13. We use induction on the structure of terms $\mathcal{T}$ of $\mathrm{GRA}(e, p, s, \neg, J, \exists)$. For the base case of the induction we note that e is equivalent to $v_1 = v_2$ and R to $R(v_1, \dots, v_k)$. Suppose then that $\mathcal{T}$ is equivalent to $\varphi(v_1, \dots, v_k)$. Then $\neg\mathcal{T}$ is equivalent to $\neg\varphi(v_1, \dots, v_k)$ and $\exists\mathcal{T}$ to $\exists v_k\varphi(v_1, \dots, v_k)$. We translate $s\mathcal{T}$ to the variant of $\varphi(v_1, \dots, v_k)$ that swaps v_{k-1} and v_k and $p\mathcal{T}$ to $\varphi(v_2, \dots, v_k, v_1)$. Finally, suppose that $\mathcal{T}$ translates to $\varphi(v_1, \dots, v_k)$ and $\mathcal{P}$ to $\psi(v_1, \dots, v_\ell)$. Now $J(\mathcal{T}, \mathcal{P})$ is translated to $\varphi(v_1, \dots, v_k) \wedge \psi(v_{k+1}, \dots, v_{k+\ell})$. This concludes the proof of the upper bound.

For the lower bound, we shall prove the sharper result that the satisfiability problem of $\mathrm{GRA}(\neg, J, \exists)$ is NP-hard. We give a reduction from SAT. Let φ be a formula of propositional logic. Let $\{p_1, \dots, p_n\}$ be the set of proposition symbols in φ, and let $\{P_1, \dots, P_n\}$ be a set of unary relation symbols. Let φ^* be the formula obtained from φ by replacing each symbol p_i with $\forall x P_i(x)$. It is easy to see that φ and φ^* are equisatisfiable. Finally, since $\forall x P_i(x)$ is equivalent to $\neg\exists\neg P_i$, the sentence φ^* can be expressed in $\mathrm{GRA}(\neg, J, \exists)$. ◀

7 Undecidable fragments of GRA

In this section we identify undecidable subsystems of GRA. We show that GRA is undecidable without s and also without e. The principal technical result of this section is the following theorem, dealing with GRA without both e and s. Recall that Π^0_1 denotes the set of languages with recursively enumerable complements.

▶ **Theorem 15.** *The satisfiability problem of* $\mathrm{GRA}(p, I, \neg, J, \exists)$ *is* Π^0_1*-hard.*

As the satisfiability problem of FO is Π^0_1-complete, Theorem 15 implies the following corollary.

▶ **Corollary 16.** *Satisfiability of* $\mathrm{GRA}(p, s, I, \neg, J, \exists)$ *and* $\mathrm{GRA}(e, p, I, \neg, J, \exists)$ *are* Π^0_1*-complete.*

A standard method of proving undecidability is via a reduction from the tiling problem of the grid $\mathbb{N} \times \mathbb{N}$, which is well-known to be Π^0_1-complete [7]. While we also follow this approach in our proof of Theorem 15, we have to deal with the fact that in $\mathrm{GRA}(p, I, \neg, J, \exists)$ we can only permute variables in a cyclic manner.

We begin by recalling the **tiling problem** for $\mathbb{N}\times\mathbb{N}$. A **tile** is a function $t : \{R, L, T, B\} \to C$ where C is a countably infinite set of colors. We let t_X denote $t(X)$. Intuitively, t_R, t_L, t_T and t_B are the colors of the right, left, top and bottom edges of a tile. Now, let $\mathbb{T}$ be a finite set of tiles. A **$\mathbb{T}$-tiling** of $\mathbb{N} \times \mathbb{N}$ is a function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{T}$ such that for all $i, j \in \mathbb{N}$, we have $t_R = t'_L$ when $f(i, j) = t$ and $f(i + 1, j) = t'$, and similarly, $t_T = t'_B$ when $f(i, j) = t$ and $f(i, j + 1) = t'$. Intuitively, the right color of each tile equals the left color of its right neighbour, and analogously for top and bottom colors. The tiling problem for the grid $\mathbb{N} \times \mathbb{N}$ asks, with the input of a finite set $\mathbb{T}$ of tiles, if there exists a $\mathbb{T}$-tiling of $\mathbb{N} \times \mathbb{N}$. It is well known that this problem is Π^0_1-complete. We will show that satisfiability of $\mathrm{GRA}(p, I, \neg, J, \exists)$ is undecidable by reducing the tiling problem to it.

Define the **standard grid** $\mathfrak{G}_{\mathbb{N}} := (\mathbb{N} \times \mathbb{N}, R, U)$ where $R = \{((i, j), (i + 1, j)) \mid i, j \in \mathbb{N}\}$ and $U = \{((i, j), (i, j+1)) \mid i, j \in \mathbb{N}\}$. If $\mathfrak{G}$ is a structure of vocabulary $\{R, U\}$ with the binary relation symbols R and U, then $\mathfrak{G}$ is **grid-like** if there is a homomorphism $\tau : \mathfrak{G}_{\mathbb{N}} \to \mathfrak{G}$. Consider then the extended vocabulary $\{R, U, L, D\}$ where L and D are binary. Define

$$\varphi_{inverses} := \forall x \forall y (R(x, y) \leftrightarrow L(y, x)) \ \wedge \ \forall x \forall y (U(x, y) \leftrightarrow D(y, x))$$
$$\varphi_{successor} := \forall x (\exists y R(x, y) \wedge \exists y U(x, y))$$
$$\varphi_{cycle} := \ \forall x \forall y \forall z \forall u [(L(y, x) \wedge U(x, z) \wedge R(z, u)) \to D(u, y)].$$

Then define $\Gamma := \varphi_{inverses} \wedge \varphi_{successor} \wedge \varphi_{cycle}$. The intended model of Γ is the standard grid $\mathfrak{G}_{\mathbb{N}}$ extended with two binary relations, L pointing left and D pointing down.

▶ **Lemma 17.** *Let* $\mathfrak{G}$ *be a structure of vocabulary* $\{R, U, L, D\}$*. If* $\mathfrak{G} \models \Gamma$*, then there is a homomorphism from* $\mathfrak{G}_{\mathbb{N}}$ *to* $\mathfrak{G} \restriction \{R, U\}$*, i.e,. to the restriction of* $\mathfrak{G}$ *to the vocabulary* $\{R, U\}$*.*

Proof. As $\mathfrak{G}$ satisfies $\varphi_{inverses}$ and φ_{cycle}, it is easy to see that $\mathfrak{G}$ satisfies the sentence $\varphi_{grid-like} := \forall x \forall y \forall z \forall u [(R(x, y) \wedge U(x, z) \wedge R(z, u)) \to U(y, u)]$. Using this sentence and $\varphi_{successor}$, it is easy to inductively construct a homomorphism from $\mathfrak{G}_{\mathbb{N}}$ to $\mathfrak{G} \restriction \{R, U\}$. ◀

Fix a set of tiles $\mathbb{T}$. We simulate the tiles $t \in \mathbb{T}$ by unary relation symbols P_t. Let $\varphi_{\mathbb{T}}$ be the conjunction of the following sentences (the 2nd one on the first row could be dropped):

$$\forall x \bigvee_{t \in \mathbb{T}} P_t(x) \qquad \bigwedge_{t \neq t'} \forall x \neg (P_t(x) \wedge P_{t'}(x))$$
$$\bigwedge_{t_R \neq t'_L} \forall x \forall y \neg (P_t(x) \wedge R(x, y) \wedge P_{t'}(y)) \qquad \bigwedge_{t_T \neq t'_B} \forall x \forall y \neg (P_t(x) \wedge U(x, y) \wedge P_{t'}(y))$$

The following claim shows, using Lemma 17, that $\mathbb{N} \times \mathbb{N}$ is $\mathbb{T}$-tilable iff $\varphi_{\mathbb{T}} \wedge \Gamma$ is satisfiable.

▷ Claim 18. The grid $\mathbb{N} \times \mathbb{N}$ is $\mathbb{T}$-tilable iff $\varphi_{\mathbb{T}} \wedge \Gamma$ is satisfiable.

Proof. Suppose there is a model $\mathfrak{G}$ so that $\mathfrak{G} \models \varphi_{\mathbb{T}} \wedge \Gamma$. Therefore, by Lemma 17, there exists a homomorphism $\tau : \mathfrak{G}_{\mathbb{N}} \to \mathfrak{G} \restriction \{R, U\}$. Define a tiling T of $\mathbb{N} \times \mathbb{N}$ by setting $T((i,j)) = t$ if $\tau((i,j)) \in P_t$. Since $\mathfrak{G} \models \varphi_{\mathbb{T}}$ and τ is homomorphism, the tiling is well-defined and correct.

Now suppose that there is a tiling T of $\mathbb{N} \times \mathbb{N}$ using $\mathbb{T}$. Thus we can expand $\mathfrak{G}_{\mathbb{N}} = (\mathbb{N} \times \mathbb{N}, R, U)$ to $\mathfrak{G}'_{\mathbb{N}} = (\mathbb{N} \times \mathbb{N}, R, U, L, D, (P_t)_{t \in \mathbb{T}})$ in the obvious way. Clearly $\mathfrak{G}'_{\mathbb{N}} \models \varphi_{\mathbb{T}} \wedge \Gamma$. ◁

We can now prove Theorem 15; it suffices to show that $\varphi_{\mathbb{T}}$ and each sentence in Γ is expressible in $\mathrm{GRA}(p, I, \neg, J, \exists)$. Now, the sentence $\varphi_{grid-like}$ in the proof of Lemma 17 reveals the key trick in our argument. The sentence $\varphi_{grid-like}$ would be the natural choice for our argument rather than φ_{cycle}. Indeed, we could replace $\Gamma = \varphi_{inverses} \wedge \varphi_{successor} \wedge \varphi_{cycle}$ in the statement of Lemma 17 by $\varphi_{successor} \wedge \varphi_{grid-like}$, as the proof of the lemma shows. But translating $\varphi_{grid-like}$ to $\mathrm{GRA}(p, I, \neg, J, \exists)$ would become an obstacle due to the arrangement of the variables and the lack of s in the algebra. We solve this issue by using φ_{cycle} instead of $\varphi_{grid-like}$. By extending the vocabulary, we were able to formulate φ_{cycle} so that the variables in it occur in a cyclic order. The steps below will demonstrate that by using this cyclicity, we can express φ_{cycle} in $\mathrm{GRA}(p, I, \neg, J, \exists)$ even though it lacks the swap operator s.

Let us first express $\varphi_{inverses}$. Note that $\varphi_{inverses}$ is equivalent to the conjunction

$$\forall x \forall y (R(x,y) \to L(y,x)) \wedge \forall x \forall y (L(y,x) \to R(x,y)) \\ \wedge \forall x \forall y (U(x,y) \to D(y,x)) \wedge \forall x \forall y (D(y,x) \to U(x,y)).$$

Let us show how to express $\forall x \forall y (R(x,y) \to L(y,x))$ in $\mathrm{GRA}(p, I, \neg, J, \exists)$; the other conjuncts are treated similarly. Consider the formula $R(x,y) \to L(y,x)$. To express this, consider first the formula $\psi := R(x,y) \to L(z,u)$ which can be expressed by the term $\mathcal{T} = \neg J(R, \neg L)$. Now, to make ψ equivalent to $R(x,y) \to L(y,x)$, we could first write $y = z \wedge x = u \wedge \psi$ and then existentially quantify z and u away. On the algebraic side, an essentially corresponding trick is done by transitioning from $\mathcal{T}$ first to $Ip(\mathcal{T})$ and then to $IpIp(\mathcal{T})$ and finally reordering this by p, i.e., going to $pIpIp(\mathcal{T})$. This term is equivalent to $R(x,y) \to L(y,x)$. Therefore the sentence $\forall x \forall y (R(x,y) \to L(y,x))$ is equivalent to $\forall\forall pIpIp\mathcal{T}$ where $\forall = \neg\exists\neg$.

Consider then the formula $\varphi_{cycle} = \forall x \forall y \forall z \forall u [(L(y,x) \wedge U(x,z) \wedge R(z,u)) \to D(u,y)]$. In the quantifier-free part, the variables occur in a cyclic fashion, but with repetitions. We first translate the repetition-free variant $(L(v_1,v_2) \wedge U(v_3,v_4) \wedge R(v_5,v_6)) \to D(v_7,v_8)$ by using $\neg$ and J, letting $\mathcal{T}$ be the resulting term. Now we would need to modify $\mathcal{T}$ so that the repetitions are taken into account. To introduce one repetition, first use p on $\mathcal{T}$ repeatedly to bring the involved coordinates to the right end of tuples, and then use I. Here p suffices (and s is not needed) because φ_{cycle} was designed so that the repeated variable occurrences are cyclically adjacent to each other in the variable ordering. Thus it is now easy to see that we can form a term $\mathcal{T}'$ equivalent to $(L(v_1,v_2) \wedge U(v_2,v_3) \wedge R(v_3,v_4)) \to D(v_4,v_1)$, and $\mathcal{T}'$ can easily be modified to a term for φ_{cycle}.

From subformulas of $\varphi_{\mathbb{T}}$, consider the formula $\neg(P_t(x) \wedge R(x,y) \wedge P_{t'}(y))$. Here $\psi(x,y) := R(x,y) \wedge P_{t'}(y)$ is equivalent to $\mathcal{T} := IJ(R, P_{t'})$ and $P_t(x) \wedge \psi(x,y)$ thus to $pIJ(p\mathcal{T}, P_t)$. It is now easy to see how to translate the rest of $\varphi_{\mathbb{T}}$ and also $\varphi_{successor}$. ◀

8 Conclusions

The principal aim of the article has been to *introduce* an approach for systematically studying logics via algebras based on finite signatures. The technical results obtained demonstrated how the setting works.

Our work can be continued into many directions; the key is to identify relevant collections of *relation operators* and provide classifications for the thereby generated systems. This work can naturally involve systems that capture FO, but also stronger, weaker and orthogonal ones. In addition to decidability, complexity and expressive power, also completeness of equational theories (including the one for GRA) is an interesting research direction.

References

1 Hajnal Andréka, István Németi, and Johan van Benthem. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.

2 Franz Baader, Ralf Küsters, and Frank Wolter. Extensions to description logics. In Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 219–261. Cambridge University Press, 2003.

3 John Bacon. The completeness of a predicate-functor logic. *Journal of Symbolic Logic*, 50:903–921, 1985.

4 Vince Bárány, Balder ten Cate, and Luc Segoufin. Guarded negation. *Journal of the ACM*, 62(3):22:1–22:26, 2015.

5 Saguy Benaim, Michael Benedikt, Witold Charatonik, Emanuel Kieronski, Rastislav Lenhardt, Filip Mazowiecki, and James Worrell. Complexity of two-variable logic on finite trees. *ACM Transactions on Computational Logic*, 17(4):32:1–32:38, 2016.

6 Johan Van Benthem. Dynamic bits and pieces. ILLC research report, University of Amsterdam, 1997.

7 Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer, 1997.

8 Witold Charatonik and Piotr Witkowski. Two-variable logic with counting and trees. *ACM Transactions on Computational Logic*, 17(4):31:1–31:27, 2016.

9 Edgar F. Codd. Relational completeness of data base sublanguages. *Research Report / RJ / IBM / San Jose, California*, RJ987, 1972.

10 Erich Grädel. Invited talk: Decision procedures for guarded logics. In Harald Ganzinger, editor, *Automated Deduction – CADE-16, 16th International Conference on Automated Deduction, Proceedings*, volume 1632 of *Lecture Notes in Computer Science*, pages 31–51. Springer, 1999.

11 Erich Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 64(4):1719–1742, 1999.

12 Erich Grädel, Phokion Kolaitis, and Moshe Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3(1):53–69, 1997.

13 Erich Grädel, Martin Otto, and Eric Rosen. Two-variable logic with counting is decidable. In *Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science LICS*, pages 306–317. IEEE, 1997.

14 Lauri Hella and Antti Kuusisto. One-dimensional fragment of first-order logic. In Rajeev Goré, Barteld P. Kooi, and Agi Kurucz, editors, *Invited and contributed papers from the tenth conference on "Advances in Modal Logic" AiML*, pages 274–293. College Publications, 2014.

15 Jelle Hellings, Catherine L. Pilachowski, Dirk Van Gucht, Marc Gyssens, and Yuqing Wu. From relation algebra to semi-join algebra: An approach to graph query optimization. *The Computer Journal*, 64(5):789–811, 2021.

16 Robin Hirsch and Ian Hodkinson. *Relation algebras by games*. North Holland, 2002.

17 Reijo Jaakkola. Ordered Fragments of First-Order Logic. In *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*, volume 202 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 62:1–62:14, 2021.

18 Reijo Jaakkola and Antti Kuusisto. Algebraic classifications for fragments of first-order logic and beyond. *CoRR*, abs/2005.01184v1, 2020.

19 Reijo Jaakkola and Antti Kuusisto. Algebraic classifications for fragments of first-order logic and beyond. *arXiv Preprint*, arXiv:2005.01184v2, 2021.

20 Emanuel Kieronski and Antti Kuusisto. Complexity and expressivity of uniform one-dimensional fragment with equality. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014 – 39th International Symposium, MFCS, Proceedings, Part I*, volume 8634 of *LNCS*, pages 365–376. Springer, 2014.

21 Emanuel Kieronski, Jakub Michaliszyn, Ian Pratt-Hartmann, and Lidia Tendera. Two-variable first-order logic with equivalence closure. *SIAM Journal on Computing*, 43(3):1012–1063, 2014.

22 Emanuel Kieronski, Ian Pratt-Hartmann, and Lidia Tendera. Equivalence closure in the two-variable guarded fragment. *J. Log. Comput.*, 27(4):999–1021, 2017.

23 Steven T. Kuhn. An axiomatization of predicate functor logic. *Notre Dame Journal of Formal Logic*, 24:233–241, 1983.

24 Antti Kuusisto. On games and computation. *CoRR*, abs/1910.14603, 2019.

25 Dirk Leinders, Maarten Marx, Jerzy Tyszkiewicz, and Jan Van den Bussche. The semijoin algebra and the guarded fragment. *Journal of Logic, Language and Information*, 14(3):331–343, 2005.

26 Per Lindström. First order predicate logic with generalized quantifiers. *Theoria*, 32(3):186–195, 1966.

27 Leopold Löwenheim. Über möglichkeiten im relativkalkül. *Matematische Annalen*, 76(4):447–470, 1915.

28 Carsten Lutz, Ulrike Sattler, and Frank Wolter. Modal logic and the two-variable fragment. In Laurent Fribourg, editor, *Computer Science Logic, 15th International Workshop, CSL*, 2001.

29 Amaldev Manuel and Thomas Zeume. Two-variable logic on 2-dimensional structures. In Simona Ronchi Della Rocca, editor, *Computer Science Logic 2013 (CSL 2013)*, volume 23 of *LIPIcs*, pages 484–499. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2013.

30 Maarten Marx. Tolerance logic. *Journal of Logic, Language and Information*, 10(3):353–374, 2001.

31 S. Ju. Maslov. The inverse method for establishing deducibility for logical calculi. In V. P. Orevkov, editor, *Logical and logical-mathematical calculus. Part I, Trudy Mat. Inst. Steklov*, volume 98, pages 26–87. Public, 1968.

32 Fabio Mogavero and Giuseppe Perelli. Binding forms in first-order logic. In Stephan Kreutzer, editor, *24th EACSL Annual Conference on Computer Science Logic, CSL*, volume 41 of *LIPIcs*, pages 648–665, 2015.

33 Michael Mortimer. Reasoning about strategies: On the model-checking problem. *Mathematical Logic Quarterly*, 21(1), 1975.

34 Andrdzej Mostowski. On a generalization of quantifiers. *Fundamenta Mathematicae*, 44(1):12–36, 1957.

35 Leszek Pacholski, Wieszlaw Szwast, and Lidia Tendera. Complexity of two-variable logic with counting. In *Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science LICS*, pages 318–327. IEEE, 1997.

36 Ian Pratt-Hartmann. Complexity of the two-variable fragment with counting quantifiers. *Journal of Logic, Language and Information*, 14(3):369–395, 2005.

37 Ian Pratt-Hartmann, Wieslaw Szwast, and Lidia Tendera. The fluted fragment revisited. *Journal of Symbolic Logic*, 84(3):1020–1048, 2019.

38 Willard Van Quine. Toward a calculus of concepts. *The Journal of Symbolic Logic*, I:2–25, 1936.

39 Willard Van Quine. Variables explained away. In *Proceedings of the American Philosophical Society*, 1960.

40 Willard Van Quine. On the limits of decision. In *Proceedings of the 14th International Congress of Philosophy*, volume III, pages 57–62. University of Vienna, 1969.

41 Willard Van Quine. Algebraic logic and predicate functors. In *Logic and Art*, pages 214–238. Bobbs-Merrill, Indianapolis, Indiana, 1972.

42 Luc Segoufin and Balder ten Cate. Unary negation. *Logical Methods in Computer Science*, 9(3), 2013.

43 Szymon Torunczyk and Thomas Zeume. Register automata with extrema constraints, and an application to two-variable logic. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 873–885. ACM, 2020.

44 Marco Voigt. A fine-grained hierarchy of hard problems in the separated fragment. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS*, pages 1–12. IEEE, 2017.