Algebraic classifications for fragments of first-order logic and beyond

Reijo Jaakkola

Tampere University
Joint work with Antti Kuusisto

Funding: Theory of computational logics – Academy of Finland grant 438874

Background

Study of decidability of fragments of first-order logic.

Background

Study of decidability of fragments of first-order logic.

Full first-order logic is well-known to be undecidable and the goal is to isolate computationally well-behaved fragments.

Background

Study of decidability of fragments of first-order logic.

Full first-order logic is well-known to be undecidable and the goal is to isolate computationally well-behaved fragments.

Current research also goes beyond first-order logic, e.g. logics with fixed-point operators.

Background

Background

Large number of different decidable fragments of first-order logic, but no *general theory*.

Background

Large number of different decidable fragments of first-order logic, but no *general theory*.

Need for a more *systematic* approach to studying the decidable fragments of first-order logic.

Our approach

Our approach is to give an algebraic characterization of first-order logic based on a *finite* algebraic signature.

Our approach

Our approach is to give an algebraic characterization of first-order logic based on a *finite* algebraic signature. This opens the door for a *systematic* classification for fragments of first-order logic.

General relational algebra

We consider the algebraic signature $(u, p, s, \neg, I, J, \exists)$.

General relational algebra

We consider the algebraic signature $(u, p, s, \neg, I, J, \exists)$.

Given a (relational) vocabulary τ, we define the set of τ-terms GRA as

$$\mathcal{T} ::= u \mid R \mid p\mathcal{T} \mid s\mathcal{T} \mid \neg\mathcal{T} \mid I\mathcal{T} \mid J(\mathcal{T}, \mathcal{T}) \mid \exists\mathcal{T}$$

where $R \in \tau$.

Arity definite relations

An AD-relation over a set A is a pair (R, k), where $R \subseteq A^k$.

Arity definite relations

An AD-relation over a set A is a pair (R, k), where $R \subseteq A^k$.

Why AD-relations?

Arity definite relations

An AD-relation over a set A is a pair (R, k), where $R \subseteq A^k$.

Why AD-relations? Consider the AD-relations $(A, 1)$ and $(A^2, 2)$ over A.

Arity definite relations

An AD-relation over a set A is a pair (R, k), where $R \subseteq A^k$.

Why AD-relations? Consider the AD-relations $(A, 1)$ and $(A^2, 2)$ over A. If we now take their complements, we obtain the AD-relations $(\varnothing, 1)$ and $(\varnothing, 2)$, so we don't lose information.

Arity definite relations

An AD-relation over a set A is a pair (R, k), where $R \subseteq A^k$.

Why AD-relations? Consider the AD-relations $(A, 1)$ and $(A^2, 2)$ over A. If we now take their complements, we obtain the AD-relations $(\varnothing, 1)$ and $(\varnothing, 2)$, so we don't lose information.

AD-relations also allow us to apply projection on an empty relation

Arity definite relations

An AD-relation over a set A is a pair (R, k), where $R \subseteq A^k$.

Why AD-relations? Consider the AD-relations $(A, 1)$ and $(A^2, 2)$ over A. If we now take their complements, we obtain the AD-relations $(\varnothing, 1)$ and $(\varnothing, 2)$, so we don't lose information.

AD-relations also allow us to apply projection on an empty relation, since the projection of $(\varnothing, 2)$ is just $(\varnothing, 1)$.

AD-relations defined by FO formulas

Consider a first-order formula $\varphi(v_{i_1}, ..., v_{i_k})$, where the free variables of φ are exactly $v_{i_1}, ..., v_{i_k}$ and $i_1 < ... < i_k$.

AD-relations defined by FO formulas

Consider a first-order formula $\varphi(v_{i_1}, \ldots, v_{i_k})$, where the free variables of φ are exactly $v_{i_1}, \ldots, v_{i_k}$ and $i_1 < \ldots < i_k$.

The formula $\varphi(v_{i_1}, \ldots, v_{i_k})$ defines an AD-relation on every model $\mathfrak{A}$

$$(\{(a_1, \ldots, a_k) \in A^k \mid \mathfrak{A} \models \varphi(a_1, \ldots, a_k)\}, k)$$

AD-relations defined by FO formulas

Consider a first-order formula $\varphi(v_{i_1}, ..., v_{i_k})$, where the free variables of φ are exactly $v_{i_1}, ..., v_{i_k}$ and $i_1 < ... < i_k$.

The formula $\varphi(v_{i_1}, ..., v_{i_k})$ defines an AD-relation on every model $\mathfrak{A}$

$$(\{(a_1, ..., a_k) \in A^k \mid \mathfrak{A} \models \varphi(a_1, ..., a_k)\}, k)$$

For example $\varphi(x_1, x_2)$ and $\varphi(x_7, x_9)$ define the same AD-relations.

AD-relations defined by FO formulas

Consider a first-order formula $\varphi(v_{i_1}, ..., v_{i_k})$, where the free variables of φ are exactly $v_{i_1}, ..., v_{i_k}$ and $i_1 < ... < i_k$.

The formula $\varphi(v_{i_1}, ..., v_{i_k})$ defines an AD-relation on every model $\mathfrak{A}$

$$(\{(a_1, ..., a_k) \in A^k \mid \mathfrak{A} \models \varphi(a_1, ..., a_k)\}, k)$$

For example $\varphi(x_1, x_2)$ and $\varphi(x_7, x_9)$ define the same AD-relations. Also note that $R(v_1, v_3, v_3)$ defines a binary AD-relation.

Semantics of GRA

Let $\mathfrak{A}$ be a τ-model. Every term $\mathcal{T}$ in GRA defines an AD-relation $\mathcal{T}^{\mathfrak{A}}$ over A as follows.

Semantics of GRA

Let $\mathfrak{A}$ be a τ-model. Every term $\mathcal{T}$ in GRA defines an AD-relation $\mathcal{T}^{\mathfrak{A}}$ over A as follows.

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

Semantics of GRA

Let $\mathfrak{A}$ be a τ-model. Every term $\mathcal{T}$ in GRA defines an AD-relation $\mathcal{T}^{\mathfrak{A}}$ over A as follows.

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

u) We define $u^{\mathfrak{A}} = (A, 1)$. The constant u can be called the **universe** constant or the **universal unary relation** constant.

Semantics of GRA

p) If $ar(\mathcal{T}) = k \geq 2$, we define

$$(p(\mathcal{T}))^{\mathfrak{A}} = \big(\{(a_2, \ldots, a_k, a_1) \,|\, (a_1, \ldots, a_k) \in \mathcal{T}^{\mathfrak{A}}\}, k\big).$$

We call p the **permutation** operator, or **cyclic permutation** operator.

Semantics of GRA

p) If $ar(\mathcal{T}) = k \geq 2$, we define

$$(p(\mathcal{T}))^{\mathfrak{A}} = \left(\{(a_2, \ldots, a_k, a_1) \,|\, (a_1, \ldots, a_k) \in \mathcal{T}^{\mathfrak{A}}\}, k\right).$$

We call p the **permutation** operator, or **cyclic permutation** operator.

s) If $ar(\mathcal{T}) = k \geq 2$, we define

$$(s(\mathcal{T}))^{\mathfrak{A}} = \left(\{(a_2, a_1, a_3, \ldots, a_k) \,|\, (a_1, \ldots, a_k) \in \mathcal{T}^{\mathfrak{A}}\}, k\right).$$

We refer to s as the **swap** operator.

Semantics of GRA

I) If $ar(\mathcal{T}) = k \geq 2$, we let

$$(I(\mathcal{T}))^{\mathfrak{A}} = \big(\{(a_1, \ldots, a_k) \,|\, (a_1, \ldots, a_k) \in \mathcal{T}^{\mathfrak{A}} \text{ and } a_1 = a_2\}, k\big).$$

We refer to I as the **identity** operator, or **equality** operator.

Semantics of GRA

I) If $ar(\mathcal{T}) = k \geq 2$, we let

$$(I(\mathcal{T}))^{\mathfrak{A}} = \big(\{(a_1, \ldots, a_k) \,|\, (a_1, \ldots, a_k) \in \mathcal{T}^{\mathfrak{A}} \text{ and } a_1 = a_2\}, k\big).$$

We refer to I as the **identity** operator, or **equality** operator.

$\exists$) If $ar(\mathcal{T}) = k \geq 1$, we let

$$(\exists(\mathcal{T}))^{\mathfrak{A}} = \big(\{(a_2, \ldots, a_k) \,|\, (a_1, \ldots, a_k) \in \mathcal{T}^{\mathfrak{A}} \text{ for some } a_1 \in A\}, k-1\big).$$

We call $\exists$ the **existence** operator, or **projection** operator.

Semantics of GRA

J) Let $ar(\mathcal{T}) = k$ and $ar(\mathcal{S}) = \ell$. We define

$$(J(\mathcal{T}, \mathcal{S}))^{\mathfrak{A}} = (\mathcal{T}^{\mathfrak{A}} \times \mathcal{S}^{\mathfrak{A}}, k + \ell).$$

We refer to J as the **join** operator.

Semantics of GRA

J) Let $ar(\mathcal{T}) = k$ and $ar(\mathcal{S}) = \ell$. We define

$$(J(\mathcal{T}, \mathcal{S}))^{\mathfrak{A}} = (\mathcal{T}^{\mathfrak{A}} \times \mathcal{S}^{\mathfrak{A}}, k + \ell).$$

We refer to J as the **join** operator.

$\neg$) Let $ar(\mathcal{T}) = k$. We define

$$(\neg(\mathcal{T}))^{\mathfrak{A}} = \left(A^k \backslash T^{\mathfrak{A}}, k\right).$$

We refer to $\neg$ as the **negation** or **complementation** operator.

GRA captures FO

Theorem

FO and GRA are equiexpressive.

GRA captures FO

Theorem

FO and GRA are equiexpressive.

Direction from GRA to FO is straightforward.

GRA captures FO

Theorem
FO and GRA are equiexpressive.

Direction from GRA to FO is straightforward.We will focus on pointing out the main ideas for the other direction.

FO is contained in GRA

Identities $x = x$ and $x = y$ can be translated to u and $IJ(u, u)$ respectively.

FO is contained in GRA

Identities $x = x$ and $x = y$ can be translated to u and $IJ(u, u)$ respectively.

The case of formulas $R(v_{i_1}, ..., v_{i_k})$ is more involved. First note that if no variable occurs twice in the tuple $(v_{i_1}, ..., v_{i_k})$ and $i_1 < ... < i_k$, then we can translate $R(v_{i_1}, ..., v_{i_k})$ simply to R.

FO is contained in GRA

Identities $x = x$ and $x = y$ can be translated to u and $IJ(u, u)$ respectively.

The case of formulas $R(v_{i_1}, ..., v_{i_k})$ is more involved. First note that if no variable occurs twice in the tuple $(v_{i_1}, ..., v_{i_k})$ and $i_1 < ... < i_k$, then we can translate $R(v_{i_1}, ..., v_{i_k})$ simply to R.

In the other case start with R and then use p, s, I and $\exists$ to express what elements are the same, after which we use to p and s to order the remaining elements in the desired order.

FO is contained in GRA

As an example consider the formula $R(v_2, v_1, v_2)$.

FO is contained in GRA

As an example consider the formula $R(v_2, v_1, v_2)$.

1. We start with the term R, which is equivalent to $R(v_1, v_2, v_3)$.

FO is contained in GRA

As an example consider the formula $R(v_2, v_1, v_2)$.

1. We start with the term R, which is equivalent to $R(v_1, v_2, v_3)$.
2. We first express that in every tuple the first and the third element are the same.

FO is contained in GRA

As an example consider the formula $R(v_2, v_1, v_2)$.

1. We start with the term R, which is equivalent to $R(v_1, v_2, v_3)$.
2. We first express that in every tuple the first and the third element are the same. This can be done with the term $IppR$, which is equivalent to $v_1 = v_2 \wedge R(v_2, v_3, v_1)$.

FO is contained in GRA

As an example consider the formula $R(v_2, v_1, v_2)$.

1. We start with the term R, which is equivalent to $R(v_1, v_2, v_3)$.
2. We first express that in every tuple the first and the third element are the same. This can be done with the term $IppR$, which is equivalent to $v_1 = v_2 \wedge R(v_2, v_3, v_1)$.
3. Next we reduce the arity of the term by using projection $\exists$.

FO is contained in GRA

As an example consider the formula $R(v_2, v_1, v_2)$.

1. We start with the term R, which is equivalent to $R(v_1, v_2, v_3)$.
2. We first express that in every tuple the first and the third element are the same. This can be done with the term $IppR$, which is equivalent to $v_1 = v_2 \wedge R(v_2, v_3, v_1)$.
3. Next we reduce the arity of the term by using projection $\exists$. This results in the term $\exists IppR$, which is equivalent to $R(v_1, v_2, v_1)$.

FO is contained in GRA

As an example consider the formula $R(v_2, v_1, v_2)$.

1. We start with the term R, which is equivalent to $R(v_1, v_2, v_3)$.
2. We first express that in every tuple the first and the third element are the same. This can be done with the term $IppR$, which is equivalent to $v_1 = v_2 \wedge R(v_2, v_3, v_1)$.
3. Next we reduce the arity of the term by using projection $\exists$. This results in the term $\exists IppR$, which is equivalent to $R(v_1, v_2, v_1)$.
4. We use s to swap the places of v_1 and v_2.

FO is contained in GRA

As an example consider the formula $R(v_2, v_1, v_2)$.

1. We start with the term R, which is equivalent to $R(v_1, v_2, v_3)$.
2. We first express that in every tuple the first and the third element are the same. This can be done with the term $IppR$, which is equivalent to $v_1 = v_2 \wedge R(v_2, v_3, v_1)$.
3. Next we reduce the arity of the term by using projection $\exists$. This results in the term $\exists IppR$, which is equivalent to $R(v_1, v_2, v_1)$.
4. We use s to swap the places of v_1 and v_2. The resulting term $s\exists IppR$ is then equivalent to $R(v_2, v_1, v_2)$.

FO is contained in GRA

How to translate $(\varphi \wedge \psi)$?

FO is contained in GRA

How to translate $(\varphi \wedge \psi)$? Let $\mathcal{T}$ be equivalent to φ and $\mathcal{S}$ be equivalent to ψ.

FO is contained in GRA

How to translate $(\varphi \wedge \psi)$? Let $\mathcal{T}$ be equivalent to φ and $\mathcal{S}$ be equivalent to ψ. We start by considering the term $J(\mathcal{T}, \mathcal{S})$.

FO is contained in GRA

How to translate $(\varphi \wedge \psi)$? Let $\mathcal{T}$ be equivalent to φ and $\mathcal{S}$ be equivalent to ψ. We start by considering the term $J(\mathcal{T}, \mathcal{S})$.

Similar to the case of atomic formulas, if the formulas φ and ψ share free variables, then we must express this using p, s, l and $\exists$.

FO is contained in GRA

How to translate $(\varphi \wedge \psi)$? Let $\mathcal{T}$ be equivalent to φ and $\mathcal{S}$ be equivalent to ψ. We start by considering the term $J(\mathcal{T}, \mathcal{S})$.

Similar to the case of atomic formulas, if the formulas φ and ψ share free variables, then we must express this using p, s, l and $\exists$.

Also we might have to use p and s to reorder elements in the tuples.

FO is contained in GRA

How to translate $(\varphi \wedge \psi)$? Let $\mathcal{T}$ be equivalent to φ and $\mathcal{S}$ be equivalent to ψ. We start by considering the term $J(\mathcal{T}, \mathcal{S})$.

Similar to the case of atomic formulas, if the formulas φ and ψ share free variables, then we must express this using p, s, l and $\exists$.

Also we might have to use p and s to reorder elements in the tuples. For example we could have a case like $\varphi(v_1, v_3) \wedge \psi(v_2)$.

FO is contained in GRA

Negation is easy, but what about $\exists v_i \varphi$?

FO is contained in GRA

Negation is easy, but what about $\exists v_i \varphi$?

Use p together with $\exists$ to project away the correct element.

FO is contained in GRA

Negation is easy, but what about $\exists v_i \varphi$?

Use p together with $\exists$ to project away the correct element. For example $\exists v_2 R(v_1, v_2, v_3)$ is equivalent to $p \exists p R$.

Fragments of GRA

What happens to the complexity of satisfiability problem if we remove some of the relation operators from GRA?

Fragments of GRA

What happens to the complexity of satisfiability problem if we remove some of the relation operators from GRA?

1. Every term in $\mathrm{GRA}\setminus\neg$ is satisfiable.

Fragments of GRA

What happens to the complexity of satisfiability problem if we remove some of the relation operators from GRA?

1. Every term in $\mathrm{GRA}\backslash\neg$ is satisfiable.
2. The set of satisfiable terms of $\mathrm{GRA}\backslash J$ is a regular language.

Fragments of GRA

What happens to the complexity of satisfiability problem if we remove some of the relation operators from GRA?

1. Every term in $\mathrm{GRA}\backslash\neg$ is satisfiable.
2. The set of satisfiable terms of $\mathrm{GRA}\backslash J$ is a regular language.
3. $\mathrm{GRA}\backslash\exists$ and $\mathrm{GRA}\backslash I$ are both NP-complete.

Fragments of GRA

What happens to the complexity of satisfiability problem if we remove some of the relation operators from GRA?

1. Every term in $\mathrm{GRA}\backslash\neg$ is satisfiable.
2. The set of satisfiable terms of $\mathrm{GRA}\backslash J$ is a regular language.
3. $\mathrm{GRA}\backslash\exists$ and $\mathrm{GRA}\backslash I$ are both NP-complete.
4. $\mathrm{GRA}(p, I, \neg, J, \exists)$ is Π^0_1-complete and thus removing u or s does not lead to a decidable logic.

Fragments of GRA

What happens to the complexity of satisfiability problem if we remove some of the relation operators from GRA?

1. Every term in $\mathrm{GRA} \backslash \neg$ is satisfiable.
2. The set of satisfiable terms of $\mathrm{GRA} \backslash J$ is a regular language.
3. $\mathrm{GRA} \backslash \exists$ and $\mathrm{GRA} \backslash I$ are both NP-complete.
4. $\mathrm{GRA}(p, I, \neg, J, \exists)$ is Π^0_1-complete and thus removing u or s does not lead to a decidable logic.

Complexity of $\mathrm{GRA} \backslash p$ remains as an open problem, but we conjecture that it is decidable.

Going beyond GRA

The system GRA is only of the many interesting systems that are equivalent to first-order logic.

Going beyond GRA

The system GRA is only of the many interesting systems that are equivalent to first-order logic.

One can also study weaker, stronger as well as orthogonal systems.

Going beyond GRA

The system GRA is only of the many interesting systems that are equivalent to first-order logic.

One can also study weaker, stronger as well as orthogonal systems. For this purpose we provide a definition for a general relation operator.

Generalized relation operator

Let AD_A denote the set of all AD-relations over A. An AD-structure is a tuple $(A, T_1, \ldots, T_k)$, where $T_1, \ldots, T_k \in \mathrm{AD}_A$.

Generalized relation operator

Let AD_A denote the set of all AD-relations over A. An AD-structure is a tuple $(A, T_1, ..., T_k)$, where $T_1, ..., T_k \in \mathrm{AD}_A$.

A bijection $g : A \to B$ is an isomorphism between AD-structures $(A, T_1, ..., T_k)$ and $(B, S_1, ..., S_k)$, if $ar(T_i) = ar(S_i)$, for every i, and g is an ordinary isomorphism between $(A, rel(T_1), ..., rel(T_k))$ and $(B, rel(S_1), ..., rel(S_k))$.

Generalized relation operator

A k-ary relation operator f is a map that outputs for any given set A, a k-ary function $f^A : (\mathrm{AD}_A)^k \to \mathrm{AD}_A$.

Generalized relation operator

A k-ary relation operator f is a map that outputs for any given set A, a k-ary function $f^A : (\mathrm{AD}_A)^k \to \mathrm{AD}_A$.

We also require that f^A is isomorphism invariant: if $(A, T_1, ..., T_k)$ and $(B, S_1, ..., S_k)$ are isomorphic via g, then also $(A, f^A(T_1, ..., T_k))$ and $(B, f^A(S_1, ..., S_k))$ are, likewise, isomorphic via g.

Generalized relation operator

A k-ary relation operator f is a map that outputs for any given set A, a k-ary function $f^A : (\mathrm{AD}_A)^k \to \mathrm{AD}_A$.

We also require that f^A is isomorphism invariant: if $(A, T_1, ..., T_k)$ and $(B, S_1, ..., S_k)$ are isomorphic via g, then also $(A, f^A(T_1, ..., T_k))$ and $(B, f^A(S_1, ..., S_k))$ are, likewise, isomorphic via g.

Generalized quantifiers can be seen as a relation operators that always output either $(\{\varnothing\}, 0) = \top_0$ or $(\varnothing, 0) = \bot_0$.

Algebraic characterizations for $\mathrm{FO}^2, \mathrm{GF}$ and FL

Providing algebraic characterizations for decidable fragments can be also used to compare different decidable fragments.

Algebraic characterizations for $\mathrm{FO}^2, \mathrm{GF}$ and FL

Providing algebraic characterizations for decidable fragments can be also used to compare different decidable fragments.

Using the suffix intersection operator $\dot{\cap}$, we were able to give very similar algebraic characterizations for the two-variable logic FO^2, guarded fragment GF and fluted logic FL.

Algebraic characterizations for $\mathrm{FO}^2, \mathrm{GF}$ and FL

Providing algebraic characterizations for decidable fragments can be also used to compare different decidable fragments.

Using the suffix intersection operator $\dot{\cap}$, we were able to give very similar algebraic characterizations for the two-variable logic FO^2, guarded fragment GF and fluted logic FL.

The suffix intersection is a generalization of intersection which can operate on relations of different arity.

Algebraic characterizations for $\mathrm{FO}^2, \mathrm{GF}$ and FL

Providing algebraic characterizations for decidable fragments can be also used to compare different decidable fragments.

Using the suffix intersection operator $\dot{\cap}$, we were able to give very similar algebraic characterizations for the two-variable logic FO^2, guarded fragment GF and fluted logic FL.

The suffix intersection is a generalization of intersection which can operate on relations of different arity. For example $R(x, y) \wedge P(y)$ is equivalent to $R \dot{\cap} P$.

Algebraic characterizations for $\mathrm{FO}^2, \mathrm{GF}$ and FL

Theorem

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

Algebraic characterizations for $\mathrm{FO}^2, \mathrm{GF}$ and FL

Theorem

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

Theorem

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

Algebraic characterizations for $\mathrm{FO}^2, \mathrm{GF}$ and FL

Theorem

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

Theorem

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

Theorem

FL and $\mathrm{GRA}(\neg, \dot{\cap}, \acute{\exists})$ are equiexpressive.