Approximating Resultants of Existential Second-Order Quantifier Elimination upon Universal Relational First-Order Formulas

Christoph Wernhard
TU Dresden, Germany

Abstract. We investigate second-order quantifier elimination for a class of formulas characterized by a restriction on the quantifier prefix: existential predicate quantifiers followed by universal individual quantifiers and a relational matrix. For a given second-order formula of this class a possibly infinite sequence of universal first-order formulas that have increasing strength and are all entailed by the second-order formula can be constructed. Any first-order consequence of the second-order formula is a consequence of some member of the sequence. The sequence provides a recursive base for the first-order theory of the second-order formula, in the sense investigated by Craig. The restricted formula class allows to derive further properties, for example that the set of those members of the sequence that are equivalent to the second-order formula, or, more generally, have the same first-order consequences, is co-recursively enumerable. Also the set of first-order formulas that entails the second-order formula is co-recursively enumerable. These properties are proven with formula-based tools used in automated deduction, such as domain closure axioms, eliminating individual quantifiers by ground expansion, predicate quantifier elimination with Ackermann's Lemma, Craig interpolation and decidability of the Bernays-Schönfinkel-Ramsey class.

1 Introduction

The objective of second-order quantifier elimination is to compute from a given second-order formula a so-called *resultant* (from German *Resultante*, used by Schröder [17]), that is, an equivalent first-order formula. Finding such a resultant is not in general possible for arbitrary second-order formulas. The early technical investigations of second-order quantifier elimination on the basis of first-order logic were done jointly with the early investigations of the decision problem: Löwenheim [14], Skolem [18] and Behmann [3] gave decision methods for relational[1] monadic[2] formulas with equality. Their methods are based on existentially quantifying upon all predicates in the formula to decide and computing a resultant by elimination. The obtained resultant of a relational monadic formula is then a truth value constant or a formula that just constrains domain cardinality (see, e.g., [23]).

Whereas much is known today about decidable fragments of first-order logic, see, e.g., [4], knowledge about fragments on which second-order quantifier elim-

[1] No function symbols with exception of individual constants.
[2] No predicate symbols with arity larger than one.

In: P. Koopmann, S. Rudolph, R. Schmidt, C. Wernhard (eds.): *SOQE 2017 – Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017*, published at `http://ceur-ws.org`.

ination succeeds seems still scarce [12]. Ackermann [1] was the first to publish[3] a proof that second-order quantifier elimination on the basis of first-order logic does not succeed in general, by means of a second-order formalization of the induction axiom. Conradie [5] formulated quite specific syntactic conditions under which the modern DLS algorithm [7] for second-order quantifier elimination succeeds. However, it appears that for the case of simultaneous elimination of multiple existential predicate quantifiers only a sufficient condition has been found. In addition, DLS as considered there does not cover the relational monadic case. Van Benthem and Doets [21] consider three classes of relational formulas with existential second-order quantifier prefix, distinguished by the subsequent first-order quantifier prefix: existential first-order quantifiers, universal first-order quantifiers, and first-order quantifier prefixes of the form $\forall x_1 \ldots \forall x_m \exists y_1 \ldots \exists y_n$.[4] As shown in [21], formulas with arbitrary further first-order prefixes can be converted to the latter class by Skolemization. For the second class, it is sketched in [21] with model theoretic arguments that a resultant exists which is an (infinite) disjunction of an (infinite) conjunction of first-order formulas.

Here we will take a closer look on that second class, applying formula-based tools that are used in automated deduction. In particular, our toolkit comprises domain closure axioms as familiar from logical modeling of database semantics, formula normalization and elimination of first-order quantifiers by ground expansion, second-order quantifier elimination by rewriting with an equivalence of a second-order formula of a certain form to a first-order formula (Ackermann's Lemma [1]), a strengthening of Craig interpolation that can be proven with a tableau technique, and decidability of the Bernays-Schönfinkel-Ramsey class.

The main original motivation was to explore possible foundations for applying instance-based [2] techniques as known for theorem proving also to solve elimination tasks. Their underlying principle is Herbrand's theorem, which justifies reducing unsatisfiability of a first-order formula to unsatisfiability of a propositional formula obtained by eliminating first-order quantifiers through ground expansion with terms constructed from the input vocabulary. The general idea of instance-based methods for theorem proving is to successively generate conjunctions of instances of the universally quantified input formulas and test these for propositional unsatisfiability. In presence of various techniques to avoid naive explicit expansion and the ability of recent SAT solvers to handle quite large propositional formulas this a practically feasible approach to first-order theorem proving. Thus, the question came up, whether there are quantifier-free formula expansions that are large enough to ensure that elimination can be performed on these instead of the original formula, in analogy to detecting unsatisfiability.

With the expectation of considering an important but somehow easier special case we focus on relational formulas, which underlie most formalizations of databases. Of course, functions can be represented by predicates, such that

[3] [15, p. 336] and a letter by Ackermann dated 1 November 1928 [22] suggest that Löwenheim earlier obtained similar results.

[4] That these classes are restricted to *relational* formulas can be guessed from the symbolic notation in [21] (actually only the case with a single unquantifed predicate seems considered there) and the observation that if function symbols would be permitted, then the second class would be as expressive as the third one.

without restriction on quantification, the *relational* property is not a limitation of expressive power. We focus on the restriction to *universal* relational formulas. In Skolemized form, formulas of the Bernays-Schönfinkel-Ramsey class are such formulas. Typically, in contrast to many resolution-based methods [10], instance-based methods decide universal relational formulas. An intuitive argument is that these formulas trivially have a largest expansion that needs to be considered to determine satisfiability. Do they also have in some sense a largest expansion that needs to be considered for elimination?

The obtained results are not as positive as desired, but at least shed some light on the properties of elimination problems with certain quantification restrictions. The considered formulas have an existential second-order quantifier prefix, followed by a universal first-order prefix and a quantifier-free formula. It is shown that for such a second-order formula F a sequence $\{G_0, G_1, G_2, \ldots\}$ of universal relational first-order formulas that have (not necessarily strictly) increasing strength and are all entailed by F can be constructed. Any first-order consequence of F is a consequence of some G_i. Formula F has a resultant if and only if it is equivalent to some G_i. The set of the formulas G_i that constitute a resultant of F or, more generally, have the same first-order consequences as F is co-recursively enumerable. Also the set of first-order formulas that entail F is co-recursively enumerable.

Craig [6] considered the question of constructing a so-called *base* for a given formula with existential second-order prefix, that is, a recursive set of first-order formulas whose set of consequences is identical to the set of first-order consequences of the given second-order formula. He notes that this corresponds to a weakened version of Ackermann's [1] generalized notion of *resultant*. Our construction of $\{G_0, G_1, G_2, \ldots\}$ can be viewed as construction of a monotonic base, actually with similar techniques as in [6], but where the restriction to universal relational formulas allows to derive additional properties.

The rest of the paper is structured as follows: Notation and definitions of the considered formula classes are introduced in Sect. 2 and the toolkit of the used techniques is specified in Sect. 3. In Sect. 4 then the main results are stated, proven and informally described, followed by a discussion of related work, potential applications and open issues in Sect. 5. Section 6 concludes the paper.

2 Notation and Preliminaries

We consider second-order formulas where second-order quantification is just upon predicates (in contrast to functions), or, in other words, first-order formulas extended by second-order quantification upon predicates. They are constructed from atoms (including equality atoms), constant operators $\top$, $\bot$, the unary operator $\neg$, binary operators $\wedge, \vee$ and quantifiers $\forall, \exists$ with their usual meaning. Further binary operators $\rightarrow$, $\leftarrow$, $\leftrightarrow$, as well as n-ary versions of $\wedge$ and $\vee$ can be understood as meta-level notation. The operators $\wedge$ and $\vee$ bind stronger than $\rightarrow$, $\leftarrow$ and $\leftrightarrow$. The scope of $\neg$, the quantifiers, and the n-ary connectives is the immediate subformula to the right.

A subformula occurrence has in a given formula *positive (negative) polarity* if it is in the scope of an even (odd) number of negations. A *vocabulary* is a set of *symbols*, that is, predicate symbols (briefly *predicates*), function symbols (briefly *functions*) and *individual symbols*. (Function symbols are assumed to have an arity ≥ 1. Individual symbols are not partitioned into variables and constants. Thus, an individual symbol is – like a predicate in second-order logic – considered as variable if and only if it is bound by a quantifier.) The set of symbols that occur *free* in a formula F is denoted by $\mathcal{V}(F)$, the set of predicate symbols that occur free in F by $\mathcal{V}_P(F)$, and the set of individual symbols that occur free in F (commonly termed the set of *constants* occurring in F) by $\mathcal{V}_C(F)$. The arity of a predicate or function symbol s is denoted by $\mathsf{arity}(s)$.

We use straightforward shorthands based on sequences of terms for expressing application of predicates and functions to arguments, simultaneous comparison of multiple terms and quantifying upon multiple variables: If $\boldsymbol{t} = t_1, \ldots, t_n$ is an n-ary sequence of terms, and s is an n-ary predicate or function symbol, we write $s\boldsymbol{t}$ for $s(t_1, \ldots, t_n)$, or, in case $n = 0$, for s, respectively. If $\boldsymbol{t} = t_1, \ldots, t_n$ and $\boldsymbol{u} = u_1, \ldots, u_n$ are both n-ary sequences of terms, we write $\boldsymbol{t} = \boldsymbol{u}$ for $t_1 = u_1 \wedge \ldots \wedge t_n = u_n$ and $\boldsymbol{t} \neq \boldsymbol{u}$ for $\neg(\boldsymbol{t} = \boldsymbol{u})$. If $\boldsymbol{x} = x_1, \ldots, x_n$ is a sequence of individual symbols or of predicates, we write $\exists \boldsymbol{s}$ ($\forall \boldsymbol{s}$, resp.) for $\exists s_1 \ldots \exists s_n$ ($\forall s_1 \ldots \forall s_n$, resp.).

We consider three specific formula classes whose members are constructed as a second-order quantifier prefix followed by a first-order quantifier prefix and a quantifier-free relational formula of first-order logic with equality. These classes are characterized by restrictions on the second-order and first-order prefix as shown in the following table, where the R in the class names points out the restriction to relational formulas:

Formula class	Second-order prefix	First-order prefix
$\forall$R-formulas	empty	$\forall x_1 \ldots \forall x_n$
$\exists\forall$R-formulas	empty	$\exists x_1 \ldots \exists x_m \forall y_1 \ldots \forall y_n$
$\exists\forall$R-formulas	$\exists p_1 \ldots \exists p_m$	$\forall x_1 \ldots \forall x_n$

The class $\forall$R-*formulas* is the class of universal relational first-order formulas. The class $\exists\forall$R-*formulas* is also known as Bernays-Schönfinkel-Ramsey class.

If F, G are formulas, we write $F \models G$ for F *entails* G; $\models F$ for F *is valid*; and $F \equiv G$ for F *is equivalent to* G, that is, $F \models G$ *and* $G \models F$. If $\mathfrak{I}$ is an interpretation and F is a formula, we write $\mathfrak{I} \models F$ for $\mathfrak{I}$ *is a model of* F.

3 Underlying Toolkit

We now specify the technical background underlying the proofs of the main results. It is essentially a small toolkit of formula-based concepts and construction techniques used in automated deduction.

3.1 Second-Order Quantifier Elimination

The objective of *second-order quantifier elimination* is to compute for a given second-order formula a first-order *resultant*, which is characterized as follows:

Definition 1 (Resultant). A *resultant* of a second-order formula F is a formula F' such that

1. F' is first-order,
2. $F' \equiv F$,
3. $\mathcal{V}(F') \subseteq \mathcal{V}(F)$.

It follows immediately from condition 2. of Def. 1 that all resultants of a given formula are equivalent. Hence, we also speak of *the* resultant of a second-order formula. From conditions 1. and 3. it follows that none of the quantified predicates possibly occurring in F do occur in F'. They are, so-to-speak, "forgotten" in F'.

The following proposition shows an equivalence of a second-order formula with a certain shape and a first-order formula, due to Ackermann [1]. It can be applied to compute the resultant of a second-order formula that matches its left side by rewriting with the right side. The DLS algorithm [7,5] for second-order quantifier elimination surrounds such rewriting steps with pre- and postprocessing operations.

Proposition 2 (Ackermann's Lemma [1]). *Let p be an n-ary predicate, let G and H be first-order formulas such that p does not occur in G, let $\boldsymbol{x} = x_1, \ldots, x_n$ be a sequence of distinct individual symbols that do not occur bound in G and do not occur in H. Let $H[p \mapsto G]$ stand for H with each occurrence of atoms $p(t_1, \ldots, t_n)$ whose predicate is p replaced by $G\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$, that is, G under the substitution that maps x_i to t_i, for $i \in \{1, \ldots, n\}$. If p does not occur with positive polarity in H, then*

$$\exists p\,(\forall \boldsymbol{x}(p\boldsymbol{x} \vee G) \wedge H) \;\equiv\; H[p \mapsto G].$$

Ackermann's lemma also holds in a dual variant: If p does not occur with negative polarity in H, then $\exists p\,(\forall \boldsymbol{x}(\neg p\boldsymbol{x} \vee G) \wedge H) \;\equiv\; H[p \mapsto G]$. For quantifier-free formulas (also with function symbols) with an existential second-order prefix, it is always possible to compute a resultant on the basis of Ackermann's lemma, as demonstrated with the following algorithm:

Algorithm 3 (Second-Order Quantifier Elimination upon Quantifier-Free Formulas).
INPUT: An formula that consists of a prefix of existential predicate quantifiers followed by a quantifier-free formula of first-order logic with equality.
METHOD: Starting with the input formula, repeatedly eliminate the innermost existential second-order quantifier with the following method: Given is a formula $\exists p\, F$, where F is quantifier-free. Convert F to disjunctive normal form $K_1 \vee \ldots \vee K_n$, where each disjunct is a conjunction of literals. Propagate the second-order quantifier inwards to obtain the formula $\exists p\, K_1 \vee \ldots \vee \exists p\, K_n$, which is equivalent to $\exists p\, F$. Eliminate $\exists p$ in each disjunct individually: Arrange the disjunct in the form

$$\exists p\,(p\boldsymbol{t}_1 \wedge \ldots \wedge p\boldsymbol{t}_k \wedge \neg p\boldsymbol{u}_1 \wedge \ldots \wedge \neg p\boldsymbol{u}_l) \wedge K',$$

where p does not occur in K' and $k, l \geq 0$. Rewrite this formula to the equivalent formula

$$\exists p\,(\forall \boldsymbol{x}\,(p\boldsymbol{x} \vee \boldsymbol{x} \neq \boldsymbol{t}_1 \wedge \ldots \wedge \boldsymbol{x} \neq \boldsymbol{t}_k) \wedge \neg p\boldsymbol{u}_1 \wedge \ldots \wedge \neg p\boldsymbol{u}_l) \wedge K',$$

where $\boldsymbol{x}$ is a sequence of fresh individual symbols whose length is the arity of p. Apply Ackermann's Lemma (Prop. 2) to rewrite this formula to its resultant

$$(\boldsymbol{u}_1 \neq \boldsymbol{t}_1 \wedge \ldots \wedge \boldsymbol{u}_1 \neq \boldsymbol{t}_k) \wedge \ldots \wedge (\boldsymbol{u}_l \neq \boldsymbol{t}_1 \wedge \ldots \wedge \boldsymbol{u}_l \neq \boldsymbol{t}_k) \wedge K'.$$

Combine these individual resultants disjunctively to obtain a resultant of $\exists p\, F$. OUTPUT: A resultant of the input formula.

Algorithm 3 also justifies the computation of resultants of formulas of the form

$$\exists p_1 \ldots \exists p_m \exists x_1 \ldots \exists x_n\, F, \tag{i}$$

where F is quantifier-free. These are the formulas of the first of the three classes with existential second-order quantification explicitly considered in [21] as mentioned in the introduction, now generalized by allowing function symbols. Formula (i) is equivalent to $\exists x_1 \ldots \exists x_n \exists p_1 \ldots \exists p_n\, F$, obtained by switching the first- and second-order quantifier prefix. If F' is a resultant of $\exists p_1 \ldots \exists p_n\, F$, obtained for example with Algorithm 3, then $\exists x_1 \ldots \exists x_n F'$ is a resultant of (i).

Algorithm 3 can be considered as a specialized variant of DLS [7], with a simpler pre- and postprocessing that just suffices to handle predicate quantification applied to quantifier-free first-order formulas. In [6] a similar technique is used and some refinements are shown. In [9] a further variant of DLS is introduced that computes resultants of quantifier-free formulas under existential second-order quantification, however constrained such that all occurrences of a quantified predicate are replaced by the same "witness" formula, whereas in Algorithm 3 it is allowed that in the processing of each disjunct K_i a different formula $(\boldsymbol{x} \neq \boldsymbol{t}_1 \wedge \ldots \wedge \boldsymbol{x} \neq \boldsymbol{t}_k)$ is used to replace p. In fact, the algorithm from [9] can be considered as based on Algorithm 3, but followed by a second step in which the different replacement formulas in each disjunct are combined to a single witness that is suitable as replacement in all disjuncts. The necessity of only a single replacement formula is imposed by the restricted form of second-order quantifier elimination through finding a witnesses that is considered in [9], which can be also viewed as finding a Boolean unifier and as finding a particular solution of the Boolean solution problem [24].

3.2 Domain Closure Axioms and Restricted Prefixes

The *domain closure axiom*, which restricts the universe of discourse to just those individuals denoted by constants in the given formula, emerged as a tool for the logical modeling of relational databases [16]. Following [8], we consider a generalized variant that in addition permits a fixed number of further objects.

Definition 4 (Domain Closure Axiom). Let $\mathcal{C} = \{c_1, \ldots, c_n\}$ be a nonempty set of individual symbols, let $k \in \mathbb{N}_0$, and assume that $x, y_1, \ldots, y_k$ are individual symbols that are not in $\mathcal{C}$. Define

$$\mathsf{DCA}^k_{\mathcal{C}} \stackrel{\text{def}}{=} \exists y_1 \ldots \exists y_k \forall x\, (x = y_1 \vee \ldots \vee x = y_k \vee x = c_1 \vee \ldots \vee x = c_n).$$

It is possible, to specify *domain closure axiom* with a formula as parameter whose free individual symbols are taken as considered set $\mathcal{C}$. However, considered as a function of formulas, *domain closure axiom* is then not "semantic",

that is, it might have semantically different values for semantically equivalent argument formulas, which, despite being equivalent, might have different sets of free individual symbols. To avoid "non-semantic" functions of formulas, we use here a version of *domain closure axiom* that is explicitly parameterized with a set $\mathcal{C}$ of individual symbols.

Domain closure axioms play a role in *general domain circumscription* [8]. As shown with [8, Thm. 6.1], the domain circumscription of an $\forall$R-formula F such that $\mathcal{V}_C(F) \neq \emptyset$ is equivalent to $\mathsf{DCA}^0_{\mathcal{C}} \wedge F$, where $\mathcal{C} = \mathcal{V}_C(F)$.

The remaining propositions in this section show properties of first-order formulas with restricted quantifier prefixes conjoined with domain closure axioms. Roughly, the intuition there is that for these formulas the addition of domain closure axioms does not essentially alter the semantics, but allows to eliminate first-order quantification by ground expansion with respect to the finite set of symbols presupposed in the domain closure axioms. The underlying core property is that from a model of an $\exists\forall$R-formula a further model can be derived that in addition also satisfies the domain closure axiom with respect to length of the existential quantifier prefix and the free individual symbols:

Proposition 5 (Domain Closure Extension for $\exists\forall$R-Formulas). *Let F be an $\exists\forall$R-formula and let $\mathfrak{I}$ be an interpretation such that $\mathfrak{I} \models F$. Then for all non-empty sets $\mathcal{C} \supseteq \mathcal{V}_C(F)$ of individual symbols and for all natural numbers k larger than or equal to the length of the existential quantifier prefix of F there exists an interpretation $\mathfrak{I}'$ such that*

1. *$\mathfrak{I}' \models \mathsf{DCA}^k_{\mathcal{C}} \wedge F$.*
2. *For all ground atoms A constructed from predicates in F and individual symbols in $\mathcal{C}$ it holds that $\mathfrak{I} \models A$ iff $\mathfrak{I}' \models A$.*

This proposition is not hard to prove by considering the submodel of $\mathfrak{I}$ obtained by restricting the domain to the union of the $\leq k$ individuals whose existence is presupposed by the existential quantifier prefix of F and the set of the values of the members of $\mathcal{V}_C(F)$ in $\mathfrak{I}$. From this proposition it follows that if an $\exists\forall$R-formula conjoined with a suitable domain closure axiom entails an $\forall\exists$R-formula, then also the $\exists\forall$R-formula alone entails that formula:

Proposition 6 (Redundancy of Domain Closure for $\exists\forall$R-$\forall\exists$R Entailments). *Let F be an $\exists\forall$R-formula and let G be an $\forall\exists$R-formula, let $\mathcal{C} \supseteq \mathcal{V}_C(F) \cup \mathcal{V}_C(G)$ be a non-empty set of individual symbols, and let k be a natural number that is larger than or equal to the sum of the length of the existential quantifier prefix of F and the length of universal quantifier prefix of G. It then holds that*

$$\text{if } \mathsf{DCA}^k_{\mathcal{C}} \wedge F \models G, \quad \text{then } F \models G.$$

Proof. Assume the left side of the proposition, that is, $\mathsf{DCA}^k_{\mathcal{C}} \wedge F \models G$. Assume further that the right side does not hold. Then there exists an interpretation $\mathfrak{I}$ such that $\mathfrak{I} \models F \wedge \neg G$. Observe that $F \wedge \neg G$ is equivalent to an $\exists\forall$R-formula with k existential quantifiers. Thus, by Prop. 5 there exists an interpretation $\mathfrak{I}'$ such that $\mathfrak{I}' \models \mathsf{DCA}^k_{\mathcal{C}} \wedge F \wedge \neg G$. With the assumption $\mathsf{DCA}^k_{\mathcal{C}} \wedge F \models G$ it then follows that $\mathfrak{I}' \models G \wedge \neg G$, which contradicts with $\mathfrak{I}'$ being an interpretation. □

From this proposition it follows that the semantics of $\forall$R-formulas F is preserved under conjunction with a suitable domain closure axiom:

Proposition 7 (Semanticity of Domain Closure for $\forall$R-formulas). *Let F, G be $\forall$R-formulas, let $\mathcal{C} \supseteq \mathcal{V}_C(F) \cup \mathcal{V}_C(G)$ $\mathcal{C}$ be a non-empty set of individual symbols, and let k be a natural number that is larger than or equal to the maximum of the quantifier prefix lengths of F and G. It then holds that*

$$\textit{if } \mathsf{DCA}^k_{\mathcal{C}} \wedge F \equiv \mathsf{DCA}^k_{\mathcal{C}} \wedge G, \quad \textit{then } F \equiv G.$$

Proof. Follows from Prop. 6. □

3.3 Resultants of $\exists\forall$R-formulas are Universal

Based on a strengthening of Craig's interpolation theorem it can be shown that whenever an $\exists\forall$R-formula has a resultant, then the resultant is equivalent to an $\forall$R-formula, and, moreover, that there is an effective method to compute for any given resultant of an $\exists\forall$R-formula such an equivalent $\forall$R-formula. Before we state this as a proposition, we show the underlying interpolation property:

Proposition 8 (Interpolants with $\exists\forall$R-Formula on the Left). *Let F be an $\exists\forall$R-formula and let G be a first-order formula such that $F \models G$. Then there is an effective method to compute from given F and G a formula H such that*

1. *H is an $\forall$R-formula.*
2. *$F \models H \models G$.*
3. *$\mathcal{V}_P(H) \subseteq \mathcal{V}_P(F) \cap \mathcal{V}_P(G)$.*
4. *$\mathcal{V}_C(H) \subseteq \mathcal{V}_C(F)$.*

Proof. Let G' be G conjoined with tautologies such that $\mathcal{V}_C(G') \supseteq \mathcal{V}_C(F)$, whereas $\mathcal{V}_P(G') = \mathcal{V}_P(G)$ and $G' \equiv G$. Let F' be the $\forall$R-formula obtained from F by renaming the quantified predicates with fresh symbols and dropping the second-order prefix. Compute H as Craig interpolant of F' and G' with the tableau-based interpolant construction method described in [19] and [11]. Conditions 2.–4. of the proposition follow from the properties of Craig interpolants: $F' \models H \models G'$ implies $F \models H \models G$; from $\mathcal{V}(H) \subseteq \mathcal{V}(F') \cap \mathcal{V}(G')$ it follows, since $\mathcal{V}_P(F') = \mathcal{V}_P(F)$ and $\mathcal{V}_P(G') = \mathcal{V}_P(G)$, that $\mathcal{V}_P(H) \subseteq \mathcal{V}_P(F) \cap \mathcal{V}_P(G)$, and, since $\mathcal{V}_C(F') = \mathcal{V}_C(F) \subseteq \mathcal{V}_C(G')$, that $\mathcal{V}_C(H) \subseteq \mathcal{V}_C(F)$. Condition 1. follows from particular features of the interpolant construction method of [19,11], where an existential quantifier in the interpolant would not be introduced if the left formula of the interpolation is universal and all individual symbols that occur free in the left formula also occur free in the right formula. □

Proposition 9 (Resultants of $\exists\forall$R-Formulas are Universal). *A resultant of an $\exists\forall$R formula F is equivalent to an $\forall$R-formula. Moreover, there is an effective method to compute from F and any given resultant a resultant that is an $\forall$R-formula.*

Proof. Assume that G is an arbitrary resultant of F. Thus $F \equiv G$. Let H be computed from F and G according to Prop. 8. It is easy to verify that H an $\forall$R-formula and is a resultant of F. □

4 Approximating Resultants of $\exists\forall_{\mathrm{R}}$-formulas

The main results of the paper are now shown: For a given $\exists\forall_{\mathrm{R}}$-formula F a sequence $\{G_0, G_1, G_2, \ldots\}$ of universal first-order formulas that have (not necessarily strictly) increasing strength and are all entailed by F can be constructed. Any first-order consequence of F is a consequence of some G_i. Formula F has a resultant if and only if it is equivalent to some G_i. However, only methods are provided to detect that the consequences of a given G_i do *not* include all first-order consequences of F, or, respectively, G_i is *not* equivalent to F. This leads to co-recursive enumerability of the first-order formulas that are equivalent to F or entail F, directly in case F has a resultant, or with respect the first-order consequences of F in case it has no resultant. Theorem 11 below makes this precise and shows further properties of the formulas G_i, some of which are underlying the proofs of the mentioned results. First the theorem is stated and proven formally, then the individual claimed properties are described informally.

The following definition is used in the statement of Theorem 11. It specifies notions of entailment and equivalence of second-order formulas modulo the sets of entailed first-order formulas.

Definition 10 (Entailment and Equivalence Modulo First-Order Consequences). For second-order formulas F, G define
(i) $F \models_{\mathsf{FO}} G$ if and only if for all first-order formulas H it holds that if $G \models H$, then $F \models H$.
(ii) $F \equiv_{\mathsf{FO}} G$ if and only if $F \models_{\mathsf{FO}} G$ and $G \models_{\mathsf{FO}} F$.

These notions are helpful to express properties of second-order formulas that possibly have no resultant. Their meaning is identical to the standard notions of entailment and equivalence, respectively, if no such second-order formulas are involved: If G is a first-order formula or a second-order formula that has a resultant, then, also in the case where F is a second-order formula, it holds that $F \models_{\mathsf{FO}} G$ if and only if $F \models G$. If each of F and G is first-order or has a resultant, then $F \equiv_{\mathsf{FO}} G$ if and only if $F \equiv G$.

Theorem 11 (Approximating Resultants of $\exists\forall_{\mathrm{R}}$-formulas). *Let F be an $\exists\forall_{\mathrm{R}}$-formula and let $\mathcal{C} \supseteq \mathcal{V}_C(F)$ be a nonempty set of individual constants. Then there exist $\forall_{\mathrm{R}}$-formulas $G_0, G_1, G_2, \ldots$ such that*

(a) The set $\{G_0, G_1, G_2, \ldots\}$ is recursive.
(b) $G_0 \mathrel{=\!|} G_1 \mathrel{=\!|} G_2 \mathrel{=\!|} \ldots \mathrel{=\!|} F$.
(c) For all $i \in \mathbb{N}_0$ it holds that $\mathsf{DCA}^i_{\mathcal{C}} \wedge G_i \equiv \mathsf{DCA}^i_{\mathcal{C}} \wedge F$.
(d) For all $i, j \in \mathbb{N}_0$ such that $i \leq j$ it holds that $\mathsf{DCA}^i_{\mathcal{C}} \wedge G_i \models \mathsf{DCA}^j_{\mathcal{C}} \wedge G_j$.
(e) For all $i \in \mathbb{N}_0$ it holds that F has a resultant that is an $\forall_{\mathrm{R}}$-formula with quantifier prefix length i if and only if $G_i \equiv F$.
(f) F has a resultant if and only if there exists a $k \in \mathbb{N}_0$ such that $G_k \equiv F$.
(g) For all $i \in \mathbb{N}_0$ and $\forall_{\mathrm{R}}$-formulas H with quantifier prefix length i it holds that $F \models H$ if and only if $G_i \models H$.
(h) There is an effective method to compute from F and a given first-order formula H such that $F \models H$ a number $k \in \mathbb{N}_0$ such that $G_k \models H$.

(*i*) *The set* $\{i \mid i \in \mathbb{N}_0 \text{ and } G_i \equiv_{\mathsf{FO}} F\}$ *is co-recursively enumerable.*
(*j*) *The set* $\{i \mid i \in \mathbb{N} \text{ and } H_i \models_{\mathsf{FO}} F\}$*, where* $\{H_1, H_2, H_3, \ldots\}$ *is the set of all* $\exists\forall_{\mathrm{R}}$*-formulas, is co-recursively enumerable (under assumption of a countable vocabulary).*

Proof. Let

$$F \;=\; \exists p_1 \ldots \exists p_n \, \forall x_1 \ldots \forall x_m \, F',$$

where F' is quantifier-free. Let $F'[a_1, \ldots, a_m]$ denote F' with x_i replaced by some individual symbol a_i, for all $i \in \{1, \ldots, m\}$. Let $\mathcal{U}$ be a set $\{u_1, u_2, u_3, \ldots\}$ of individual symbols that are not in $\mathcal{C}$ and let $\mathcal{U}_i$ denote $\{u_1, \ldots, u_i\}$. For $i \in \mathbb{N}_0$ construct G_i as

$$G_i \;\stackrel{\text{def}}{=}\; \forall u_1 \ldots \forall u_i \, G_i',$$

where G_i' is the resultant, computed by Algorithm 3, of the following formula:

$$\exists p_1 \ldots \exists p_n \bigwedge_{a_1, \ldots, a_m \in \mathcal{C} \cup \mathcal{U}_i} F'[a_1, \ldots, a_m].$$

The following semantic property of G_i, for all $i \in \mathbb{N}_0$, then immediately follows from the construction of G_i:

$$(*) \qquad G_i \;\equiv\; \forall u_1 \ldots \forall u_i \, \exists p_1 \ldots \exists p_n \, \textstyle\bigwedge_{a_1, \ldots, a_m \in \mathcal{C} \cup \mathcal{U}_i} F'[a_1, \ldots, a_m].$$

(a) Follows from the construction of the formulas G_i: For any given first-order formula whose universal first-order quantifier prefix has length $i \geq 0$ it can be decided whether it is a member of $\{G_0, G_1, G_1, \ldots\}$ by comparing it syntactically with G_i.

(b) Let $i, j \in \mathbb{N}_0$ that $i \leq j$. Since then $\mathcal{U}_j \supseteq \mathcal{U}_i$ it follows that

$$\bigwedge_{a_1, \ldots, a_m \in \mathcal{C} \cup \mathcal{U}_j} F'[a_1, \ldots, a_m] \;\models\; \bigwedge_{a_1, \ldots, a_m \in \mathcal{C} \cup \mathcal{U}_i} F'[a_1, \ldots, a_m].$$

With $(*)$ this implies $G_j \models G_i$. To conclude the proof of (b) we show that for an arbitrary number $i \in \mathbb{N}_0$ it holds that $F \models G_i$. This can be proven in the following steps, where the last step, the contraction into G_i, is justified by $(*)$:

$$\begin{aligned}
& F \\
\equiv\; & \exists p_1 \ldots \exists p_n \, \forall x_1 \ldots \forall x_m \, F' \\
\models\; & \exists p_1 \ldots \exists p_n \, \forall u_1 \ldots \forall u_i \, \textstyle\bigwedge_{a_1, \ldots, a_m \in \mathcal{C} \cup \mathcal{U}_i} F'[a_1, \ldots, a_m] \\
\models\; & \forall u_1 \ldots \forall u_i \, \exists p_1 \ldots \exists p_n \, \textstyle\bigwedge_{a_1, \ldots, a_m \in \mathcal{C} \cup \mathcal{U}_i} F'[a_1, \ldots, a_m] \\
\equiv\; & G_i.
\end{aligned}$$

(c) The right-to-left direction follows immediately from (b). The left-to-right direction can be shown in the following steps, where the expansion of G_i at the first step is justified by $(*)$:

$$
\begin{aligned}
& \mathsf{DCA}^i_{\mathcal{C}} \wedge G_i \\
\equiv\ & \mathsf{DCA}^i_{\mathcal{C}} \wedge \forall u_1 \ldots \forall u_i\, \exists p_1 \ldots \exists p_n \bigwedge_{a_1,\ldots,a_m \in \mathcal{C} \cup \mathcal{U}_i} F'[a_1, \ldots, a_m] \\
\equiv\ & \exists u_1 \ldots \exists u_i\, \mathsf{DCA}^0_{\mathcal{C} \cup \mathcal{U}_i} \wedge \\
& \quad \forall u_1 \ldots \forall u_i\, \exists p_1 \ldots \exists p_n \bigwedge_{a_1,\ldots,a_m \in \mathcal{C} \cup \mathcal{U}_i} F'[a_1, \ldots, a_m] \\
\models\ & \exists u_1 \ldots \exists u_i\, (\mathsf{DCA}^0_{\mathcal{C} \cup \mathcal{U}_i} \wedge \exists p_1 \ldots \exists p_n \bigwedge_{a_1,\ldots,a_m \in \mathcal{C} \cup \mathcal{U}_i} F'[a_1, \ldots, a_m]) \\
\equiv\ & \exists u_1 \ldots \exists u_i\, \mathsf{DCA}^0_{\mathcal{C} \cup \mathcal{U}_i} \wedge \exists p_1 \ldots \exists p_n\, \forall x_1 \ldots \forall x_m\, F' \\
\equiv\ & \mathsf{DCA}^i_{\mathcal{C}} \wedge \exists p_1 \ldots \exists p_n\, \forall x_1 \ldots \forall x_m\, F' \\
\equiv\ & \mathsf{DCA}^i_{\mathcal{C}} \wedge F.
\end{aligned}
$$

(d) From $i \leq j$ it follows that $\mathsf{DCA}^i_{\mathcal{C}} \models \mathsf{DCA}^j_{\mathcal{C}}$. By ($c$) we can conclude $\mathsf{DCA}^i_{\mathcal{C}} \wedge G_i \equiv \mathsf{DCA}^i_{\mathcal{C}} \wedge F \models \mathsf{DCA}^j_{\mathcal{C}} \wedge F \equiv \mathsf{DCA}^j_{\mathcal{C}} \wedge G_j$.

(e) The right-to-left direction follows immediately from the construction of G_i: If $G_i \equiv F$, then G_i clearly is a resultant of F that is an $\forall$R-formula with quantifier prefix length i. The left-to-right direction can be show as follows: Assume that there exists an $\forall$R-formula H with quantifier prefix length i that is a resultant of F. Since $H \equiv F$ it follows from (c) that

$$\mathsf{DCA}^i_{\mathcal{C}} \wedge G_i \ \equiv\ \mathsf{DCA}^i_{\mathcal{C}} \wedge H.$$

This equivalence matches the precondition of Prop. 7 that lets us deduce $G_i \equiv H$, implying $G_i \equiv F$.

(f) Follows from (e) and Prop. 9.

(g) The right-to-left direction follows immediately from (b). The left-to-right direction can be shown in the following steps, where the last two equivalences follow from (c) and Prop. 6, respectively:

$$F \models H \ \text{ implies } \ \mathsf{DCA}^i_{\mathcal{C}} \wedge F \models H \ \text{ iff } \ \mathsf{DCA}^i_{\mathcal{C}} \wedge G_i \models H \ \text{ iff } \ G_i \models H.$$

(h) By Prop. 8, we can compute from F and H an $\forall$R-formula K such that $F \models K \models H$. Let k be the length of the quantifier prefix of K. From (g) it follows that $G_k \models K$, hence $G_k \models H$.

(i) The following algorithm halts for a given $i \in \mathbb{N}_0$ if and only if $G_i \not\equiv_{\mathsf{FO}} F$: Let j be an integer variable that satisfies the invariant $j > i$ and is initialized to $i + 1$. Proceed in a loop: Test whether $G_i \models G_j$, that is, whether $G_i \wedge \neg G_j$ is satisfiable. Since G_i and G_j are $\forall$R-formulas, $G_i \wedge \neg G_j$ is an $\exists\forall$R-formula, and thus decidable. If the test succeeds, then halt, else increment j by 1 and re-enter the loop.

That the algorithm indeed halts if and only if $G_i \not\equiv_{\mathsf{FO}} F$ can be shown as follows: By (b) it holds that $F \models G_i$. Hence $G_i \not\equiv_{\mathsf{FO}} F$ if and only $G_i \not\models_{\mathsf{FO}} F$. Consider the case $G_i \not\models_{\mathsf{FO}} F$ and first the subcase where there exists a natural number $j > i$ such that $G_i \not\models G_j$. Then the satisfiability test in the algorithm eventually succeeds and the algorithm halts. Now consider the alternate subcase where no such number j exists. From (b) it then follows that for all $l \in \mathbb{N}_0$ it holds that $G_i \models G_l$. With (h) we can conclude that it holds for all first-order formula H that if $F \models H$, then $G_i \models H$. Hence $G_i \models_{\mathsf{FO}} F$, contradicting the assumption $G_i \not\models_{\mathsf{FO}} F$ made for that case, and thus yielding the alternate subcase impossible.

Now consider the case $G_i \models_{\mathsf{FO}} F$. From the definition of $\models_{\mathsf{FO}}$ it follows that for all $j \geq 0$ it holds that if $F \models G_j$, then $G_i \models G_j$. With (b) it follows that for all $j \geq 0$ it holds that $G_i \models G_j$, which implies that the satisfiability test in the algorithm never succeeds and the algorithm thus loops forever.

(j) The following algorithm halts for a given $\exists\forall_{\mathrm{R}}$-formula H if and only if $H \not\models_{\mathsf{FO}} F$: Let j be an integer variable that is initialized to 0. Proceed in a loop: Test whether $H \not\models G_j$, that is, whether $H \wedge \neg G_j$ is satisfiable. Since H is an $\exists\forall_{\mathrm{R}}$-formula and G_i is an $\forall_{\mathrm{R}}$-formula, $H \wedge \neg G_j$ is an $\exists\forall_{\mathrm{R}}$-formula, and thus decidable. If the test succeeds, then halt, else increment j by 1 and re-enter the loop.

That the algorithm indeed halts if and only if $H \not\models_{\mathsf{FO}} F$ follows since $H \not\models_{\mathsf{FO}} F$ holds if and only if there exists a $j \in \mathbb{N}_0$ such that $H \not\models G_j$, or, equivalently, $H \models_{\mathsf{FO}} F$ if and only if for all $j \in \mathbb{N}_0$ it holds that $H \models G_j$. The left-to-right direction of this equivalence can be proven as follow: Assume the left side $H \models_{\mathsf{FO}} F$. By expanding $\models_{\mathsf{FO}}$ this can be expressed as: For all first-order formulas K it holds that if $F \models K$, then $H \models K$. Hence, for all $j \in \mathbb{N}_0$ it holds that if $F \models G_j$, then $H \models G_j$. With (b) it follows that for all $j \in \mathbb{N}_0$ it holds that $H \models G_j$, that is, the right side. The right-to-left direction of the equivalence to show can be proven as follows: From (h) it follows that for all first-order formulas K it holds that if $F \models K$, then there exists a $k \in \mathbb{N}_0$ such that $G_k \models K$. Hence, if for all $i \in \mathbb{N}_0$ it holds that $H \models G_j$, then for all first-order formulas K such that $F \models K$ it holds that $H \models K$. By contracting into $\models_{\mathsf{FO}}$, the latter statement can be expressed as: If for all $j \in \mathbb{N}_0$ it holds that $H \models G_j$, then for $H \models_{\mathsf{FO}} F$, that is, the right-to-left direction of the equivalence to show. □

The formulas G_i whose existence is claimed by Theorem 11 are constructed from F and i as follows: The first-order quantifiers in F, which are universal, are eliminated by expansion with respect to the members of $\mathcal{C}$ and i additional individual symbols $u_1, \ldots, u_i$. Then a resultant of the obtained formula, that is, of the second-order quantifier prefix of F applied to the quantifier-free expansion, is computed. The formula G_i is then obtained by prefixing that resultant, which is quantifier-free, with existential first-order quantifiers upon $u_1, \ldots, u_i$.

By property (b), with increasing i the formulas G_i get (not necessarily strictly) stronger, and all the formulas G_i are entailed by F.

Property (c) holds invariantly for all $i \in \mathbb{N}_0$: Formula G_i "under domain closure with i existential individuals" (that is, conjoined with $\mathsf{DCA}^i_{\mathcal{C}}$) is equivalent to F under domain closure with the same number of existential individuals. This property is used in the proofs of (d), (e), and (g).

Property (d), which follows from (c), shows that with increasing i the formulas G_i under domain closure with i existential objects get (not necessarily strictly) *weaker*, conversely to the formulas G_i themselves, as shown with (b).

Properties (e) and (f) show necessary and sufficient conditions for the existence of a resultant of F, and in case of existence give a resultant. The first of these, (e), states that F has a resultant that is a universal relational formula with quantifier prefix length i if and only if G_i is equivalent to F. This property follows from (c) and Prop. 7. With Prop. 9 it leads to (f), which states that F has a resultant if and only if it is equivalent to G_k for some $k \in \mathbb{N}_0$. The

right-to-left directions of the respective equivalences $G_i \equiv F$ and $G_k \equiv F$ are immediate from (b), such that the existence of a resultant can be also characterized with just the entailments $G_i \models F$ and $G_k \models F$, respectively, instead. These entailments have F on their right side, a second-order formula, such that, differently from first-order logic, there is in general no algorithm that halts if and only if such an entailment holds.

Property (g) shows that F and G_i have the same $\forall_R$-formulas with quantifier prefix length i as consequences. This follows from (c) and Prop. 6. It is used together with the strengthened Craig interpolation property Prop. 8 to prove (h), by which any first-order consequence of F is a consequence of some G_k, and, moreover, such an index k can be effectively computed from F and the given consequence. Property (h) is applied to prove (i) and (j).

Properties (i) and (j) show certain settings where co-recursive enumerability with respect to equivalence and entailment, respectively, of the second-order formula F can be established. Property (i) states that the set of (the index numbers i of) the formulas G_i that are different from F with respect to their first-order consequences is recursively enumerable. This means that there is an algorithm that halts for given i if and only if G_i is *not* a formula with the same first-order consequences as F. If F has a resultant, this is equivalent to the statement that G_i *is not a resultant of* F. Property (i) is proven by giving such an algorithm and showing its correctness with referring to (b) and (h). By (b) the negated equivalence $G_i \not\equiv_{\mathsf{FO}} F$ can also be expressed as the negated entailment $G_i \not\models_{\mathsf{FO}} F$, which in the case where F has a resultant is equivalent to $G_i \not\models F$. From the perspective of trying to find a resultant of F or, more generally, a formula with the same first-order consequences as F, the property (i) only justifies a method to exclude failing candidate formulas G_i.

Property (j) states that the set of (the code numbers in some arithmetization of syntax of) the first-order formulas that do *not* entail F with respect to its first-order consequences is recursively enumerable. This means that there is an algorithm that halts for a given first-order formula H and only if $H \not\models_{\mathsf{FO}} F$. Like (i), this property is proven by giving such an algorithm and showing its correctness with referring to (b) and (h). The property justifies a method that detects for a given first-order formula H in the case where F has a resultant that F is *not* a consequence of H, and in the case where F has no resultant that *not* all first-order consequences of F are included in the consequences of H.

5 Discussion and Open Issues

In this section, Craig's work [6] on recursive *bases* is briefly compared, attempts are made to place the results of Theorem 11 in the context of applications of second-order quantifier elimination, open issues are shown, and potential directions for further research are indicated.

Comparison to Craig's Construction of Recursive Bases. The setting in [6] is more general and comprehensive: First-order formulas under existential second-order quantification are considered without assuming syntactic restrictions and also the case without equality is considered. Our syntactic restriction allows an apparently simpler construction of the approximation formulas G_i for

which monotonicity (i.e., $G_0 \models\!\!\models G_1 \models\!\!\models G_2 \models\!\!\models \ldots$) directly follows. The restriction also allows to derive further properties of the approximation formulas: They are universal and the length of their quantifier prefix is related to that of entailed universal first-order consequences of the second-order formula. Decidability of specific entailment problems, which is implied by the syntactic restrictions, leads to co-recursive enumerability of certain sets.

Our construction of an approximation formula G_i for a second-order formula $F = \exists \boldsymbol{p}\, F'$ where F' is first-order involves the construction of a first-order prefix $\boldsymbol{Q}_i$ and of an intermediate quantifier-free formula F'_i such that $F = \exists \boldsymbol{p}\, F' \models \exists \boldsymbol{p}\boldsymbol{Q} F'_i \models \boldsymbol{Q}\exists \boldsymbol{p} F'_i \equiv G_i$. The setting in [6] is the same, except that the first entailment is replaced by an equivalence, that is, it holds that $\exists \boldsymbol{p}\, F' \equiv \exists \boldsymbol{p}\boldsymbol{Q} F'_i$. Actually, the techniques in [6] seem even to preserve $F' \equiv \boldsymbol{Q} F'_i$. Of course, more weakly constrained transformations, in our case mainly justified through the use of domain closure axioms, are favorable. However, it remains to be investigated in how far the weaker constraints are made possible through the restricted formula class and whether there are associated complexity properties.

Entailments Involving Existential Second-Order Formulas. Property (j) of Theorem 11 can be considered in the context of mechanical verification and falsification (that is, existence of an algorithm that terminates in case a statement does hold or does not hold, respectively) of entailments of the forms $F \models H$ and $H \models F$, where F is a second-order formula with an existential second-order quantifier prefix applied to a first-order formula and H is a first-order formula.

An application of the first form $F \models H$ is to verify that a first-order formula A is semantically independent from predicates $p_1 \ldots, p_n$, which can be expressed as $\exists p_1 \ldots \exists p_n\, A \models A$. The first form $F \models H$ is straightforwardly accessible to *verification*: The set $\{i \mid i \in \mathbb{N} \text{ and } F \models H_i\}$, where $\{H_1, H_2, H_3, \ldots\}$ is the set of all first-order formulas, is recursively enumerable, which follows from recursive enumerability of the set of (the code numbers of) the valid first-order formulas. The entailment $F \models H_i$ is equivalent to the first-order entailment $F' \models H_i$, where F' is obtained from F by dropping the second-order prefix and renaming the quantified predicates with fresh symbols. The entailment $F' \models H_i$ can then be expressed as validity of $F' \rightarrow H_i$. Moreover, if F and H_i are restricted such that $F' \wedge \neg H_i$ belong to a decidable formula class, then $\{i \mid i \in \mathbb{N} \text{ and } F \models H_i\}$ is recursive, allowing then also to *falsify* entailments of the form $F \models H$.

An application of the second form $H \models F$ is expressing for first-order formulas A and B that $A \wedge B$ is a conservative extension of A as the entailment $A \models \exists p_1 \ldots \exists p_n (A \wedge B)$, where $p_1, \ldots, p_n$ are the predicates that occur in B but not in A. Justified by property (j) of Theorem 11, the second form $H \models F$ (if restricted to $\exists\forall_{\mathrm{R}}$-formulas H and $\exists\forall$-formulas F) can be mechanically *falsified*.

To sum up, entailments of the form $F \models H$ can always be verified and for formula classes that lead to decidable formulas $F \wedge \neg H$ also be falsified. Property (j) of Theorem 11 extends this, by establishing that also entailments of the form $H \models F$ can be falsified, for certain formula classes.

Approximate Resultants with Respect to Quantifier Prefix Length. By property (g) of Theorem 11, the $\exists\forall_{\mathrm{R}}$-formula F and the formulas G_i constructed from it have the same $\forall_{\mathrm{R}}$-formulas *with quantifier prefix length* i as

consequences. In a sense, the formulas G_i can be considered as capturing the first-order semantics of F "up to quantifier prefix length i". This suggests to consider G_i as resultant of a generalized form of elimination where not just the existentially quantified predicates are "forgotten", but also the part of the formula's meaning that would be only expressible with quantifier prefix length $> i$. Exploring this idea is an open issue.

Showing Non-Recursiveness. Properties (i) and (j) of Theorem 11 show co-recursive enumerability of certain sets related to second-order quantifier elimination. It remains to consider the question whether this is the strongest recursiveness property that can be asserted about these sets, that is, to show whether they are actually *not recursive*. It is expected that this holds because the formula $\exists f\,(x \wedge \neg fy \wedge \forall u \forall v\,(\neg fu \vee fv \vee \neg nuv)$ used in [1] to show non-existence of a resultant is actually an $\exists\forall_{\mathrm{R}}$-formula.

Potential Approaches for Strengthening the Co-Recursive Enumerability. Of course, it would be of interest, not just for practical application, to strengthen the co-recursive enumerability of finding resultants and verifying entailments shown with properties (i) and (j) of Theorem 11 to recursiveness, at least for special cases. So far, this is an open issue. A direction for further investigation might be trying to determine for certain $\exists\forall_{\mathrm{R}}$-formulas the maximally required quantifier prefix length of the resultant. A further direction could be ensuring that a semantic fixed point G_k of the sequence $G_0, G_1, G_2, \ldots$ subsumes all G_i with $i \geq k$, that is, for all $i \geq k$ it holds that $G_i \equiv G_k$. This would follow, for example, if for all $i, j \in \mathbb{N}_0$ it holds that if $G_i \equiv G_j$, then $G_{i+1} \equiv G_{j+1}$. A third direction would be trying to express *for all $i \geq k$ it holds that $G_i \equiv G_k$* in some algorithmically verifiable way.

Possibly Generalization to Further Formula Classes. Theorem 11 takes decidability of the Bernays-Schönfinkel-Ramsey class ($\exists\forall_{\mathrm{R}}$-formulas) as basis to derive co-recursive enumerability of problems related to computing elimination resultants of existential second-order quantifiers upon universal relational formulas ($\exists\forall$-formulas). This raises the question, whether the techniques applied there can be generalized to further formula classes.

A straightforward transfer appears to be the computation of resultants of formulas of the Bernays-Schönfinkel-Ramsey class under existential second-order quantification, by switching the existential second- and first-order quantifier prefixes and then considering elimination of the inner $\exists\forall_{\mathrm{R}}$-formula. However, with this approach a further source of failure to find resultants sneaks in: There are $\exists\exists\forall_{\mathrm{R}}$-formulas that have a resultant, but where after the suggested quantifier switching the obtained inner $\exists\forall_{\mathrm{R}}$-formula does not have a resultant. As an example, consider a formula $\exists p\,(x = a \vee F)$ that does not have a resultant. However, $\exists p\,\exists x\,(x = a \vee F)$ clearly has for arbitrary formulas F the resultant $\top$.

Avoiding Explicit Ground Expansion. The construction of the approximation formulas G_i described in the proof of Theorem 11 involves ground expansion of the input formula F. As in instance-based theorem proving, such expansions are useful as a conceptual construct, but their construction should in practice usually be avoided as much as possible. With respect to elimination, a poten-

tial approach might be the use of quantifier relativizations to compactly express formulas equivalent to expansions. Possibly the resultant required in the construction of G_i can then be computed by applying the elimination method for single ground atoms described in [13] to the finite number of atoms upon which the quantifiers are relativized. Also techniques of [6], where quantified formulas are duplicated by conjoining copies of them, but without performing instantiation, might be relevant here.

6 Conclusion

We have considered second-order quantifier elimination for a class of relational formulas characterized by a restriction of the quantifier prefix: existential predicate quantifiers followed by universal individual quantifiers. The main original motivation was to transfer instance-based techniques from automated theorem proving to second-order quantifier elimination. The technical result, however, does not indicate an immediate possibility for such a transfer, but gives some insight into the elimination problem for this class: The set of elimination resultants of a given formula and the set of formulas entailing the given second-order formula of that class is co-recursively enumerable. Candidate resultants can be generated, and there is an algorithm that halts on exactly those candidates that are *not* a resultant. Similarly, there is a method to detect that a given first-order formula does *not* entail the given second-order formula. By comparing formulas with respect to their first-order consequences, it is possible to express the respective theorem statements in a generalized way that applies to given second-order formulas independently of whether they have a resultant. These results were proven on the basis of small number of formula-based tools used in automated deduction. Actually, the results and involved constructions might be seen as a specialization to a formula class of Craig's setting of determining recursive *bases* for subtheories of first-order formulas. The hope is that some inspiration and material for further investigation of "eliminability", that is, existence of a resultant, or, more generally, of a formula that is equivalent with respect to first-order consequences, is provided.

Acknowledgments. This work was supported by DFG grant WE 5641/1-1.

References

1. Ackermann, W.: Untersuchungen über das Eliminationsproblem der mathematischen Logik. Math. Ann. 110, 390–413 (1935)
2. Baumgartner, P., Thorstensen, E.: Instance based methods – A brief overview. KI 24(1), 35–42 (Apr 2010)
3. Behmann, H.: Beiträge zur Algebra der Logik, insbesondere zum Entscheidungsproblem. Math. Ann. 86(3–4), 163–229 (1922)
4. Börger, E., E., Grädel, Gurevich, Y.: The Classical Decision Problem. Springer (1997)
5. Conradie, W.: On the strength and scope of DLS. J. Applied Non-Classical Logic 16(3–4), 279–296 (2006)
6. Craig, W.: Bases for first-order theories and subtheories. J. Symb. Log. 25(2), 97–142 (1960)

7. Doherty, P., Łukaszewicz, W., Szałas, A.: Computing circumscription revisited: A reduction algorithm. J. Autom. Reasoning 18(3), 297–338 (1997)
8. Doherty, P., Łukaszewicz, W., Szałas, A.: General domain circumscription and its effective reductions. Fundamenta Informaticae 36, 23–55 (1998)
9. Eberhard, S., Hetzl, S., Weller, D.: Boolean unification with predicates. J. Logic and Computation 27(1), 109–128 (2017)
10. Fermüller, C., Leitsch, A., Hustadt, U., Tammet, T.: Resolution decision procedures. In: Robinson, A., Voronkov, A. (eds.) Handb. of Autom. Reasoning, vol. 2, pp. 1793–1849. Elsevier (2001)
11. Fitting, M.: First-Order Logic and Automated Theorem Proving. Springer, 2nd edn. (1995)
12. Gabbay, D.M., Schmidt, R.A., Szałas, A.: Second-Order Quantifier Elimination: Foundations, Computational Aspects and Applications. College Publications (2008)
13. Lin, F., Reiter, R.: Forget It! In: Working Notes, AAAI Fall Symposium on Relevance. pp. 154–159 (1994)
14. Löwenheim, L.: Über Möglichkeiten im Relativkalkül. Math. Ann. 76, 447–470 (1915)
15. Löwenheim, L.: Funktionalgleichungen im Gebietekalkül und Umformungsmöglichkeiten im Relativkalkül. History of Philosophy and Logic 28, 305–336 (2007), assumed to be written in 1935 [20]
16. Reiter, R.: Deductive quastion-answering on relational databases. In: Gallaire, H., Minker, J. (eds.) Logic and Databases, pp. 149–178. Plenum Press, New York (1978)
17. Schröder, E.: Vorlesungen über die Algebra der Logik, vol. 1. Teubner (1890)
18. Skolem, T.: Untersuchungen über die Axiome des Klassenkalküls und über Produktations- und Summationsprobleme welche gewisse Klassen von Aussagen betreffen. Videnskapsselskapets Skrifter I. Mat.-Nat. Klasse(3) (1919)
19. Smullyan, R.M.: First-Order Logic. Springer, New York (1968), also republished with corrections by Dover publications, New York, 1995
20. Thiel, C.: A short introduction to Löwenheim's life and work and to a hitherto unknown paper. History of Philosophy and Logic 28, 289–302 (2007)
21. Van Benthem, J., Doets, K.: Higher-order logic. In: Handbook of Philosophical Logic, vol. 1, pp. 189–243. Springer, second edn. (2001)
22. Wernhard, C.: Heinrich Behmann's contributions to second-order quantifier elimination. Tech. Rep. KRR 15–05, TU Dresden (2015)
23. Wernhard, C.: Second-order quantifier elimination on relational monadic formulas – A basic method and some less expected applications. In: TABLEAUX 2015. LNCS (LNAI), vol. 9323, pp. 249–265. Springer (2015)
24. Wernhard, C.: The Boolean solution problem from the perspective of predicate logic. In: FroCoS 2017. LNCS (LNAI), vol. 10483, pp. 333–350 (2017)