



Reintroducing the Second Player in EPR

Leroy Chew¹, Mikoláš Janota¹, Miroslav Olšák², and Martin Suda¹

¹ Czech Technical University in Prague, Prague, Czech Republic.
{leroyche,mikolas.janota,martin.suda}@cvut.cz

² Charles University, Prague, Czech Republic. mirek@olsak.net

Abstract

We present a PSPACE-complete fragment of first order logic we call QEALM (QBF-like EPR by Alternating Level-ordered Miniscoping). Unlike other well-known fragments it is closer to QBF (Quantified Boolean Formulas) in nature because it has game-like alternations and can be restricted to complete problems for the polynomial hierarchy. QEALM is a natural first-order analogue to QBF just as EPR (Effectively Propositional logic) is a first-order analogue to Dependency QBF.

We prove the PSPACE and Polynomial Hierarchy completeness theorems of QEALM along with several other nice properties such as closure under Robinson’s resolution and retained hardness after intersection with Schaefer fragments. We identify problems in the TPTP library that fall into this fragment and their level in the polynomial hierarchy.

1 Introduction

In automated reasoning, translations to formal logic are an effective tool for efficient solving, trusted proving and a solid theoretical backbone. Whether to choose a propositional or first-order logic encoding can depend on the problem. In propositional logic all objects are binary and is often sufficient for computational tasks. But what is possible may not always be practical, and First-Order Logic (FOL) offers quantifiers, variables and predicates. Satisfiability for propositional logic is NP-complete [7], whereas unsatisfiability for FOL is only semi-decidable. And the computational complexity of the problem can often dictate the best choice for encoding.

If we fix our focus to NEXPTIME, we can find complete problems using either propositional or first-order ideas. The Bernays-Schönfinkel fragment is nearly a century old [20] and defines the FOL fragment that is free of function symbols and existential quantification. The Bernays-Schönfinkel fragment is also known as Effectively Propositional (EPR), because of an exponential reduction to propositional logic, hence its satisfiability is NEXPTIME for which it is also complete. For propositional logic, we obtain NEXPTIME-completeness by first extending it with Boolean quantifiers. This gives the PSPACE-complete Quantified Boolean Formulas (QBF). To get NEXPTIME-completeness, one allows Henkin quantification [8], which relaxes the quantifier ordering to non-transitivity which defines the Dependency QBF (DQBF) problem. This means, for NEXPTIME problems, we have a choice of a Boolean or a first-order encoding.

In this paper we aim to have the same dichotomy for PSPACE. We can already extend propositional logic to the PSPACE-complete QBF, so our aim is to define an analogous PSPACE-complete fragment of FOL. Our proposed fragment (eventually called QEALM) will share some

similarities to QBF, but has the advantage of being able to handle non-Boolean elements. Because QEALM is a sub-fragment of Bernays-Schönfinkel, (paralleling how QBF extends to DQBF) it means that the game evaluation is not done in the quantifier prefix, as all variables are universal. Instead, the ordered list of arguments for each predicate symbol, takes the place of a prefix. To achieve this in PSPACE we have restricted QEALM to CNF problems, where each clause must have the QEALM condition (Definition 1). Therefore the union and intersection of QEALM problems is also a QEALM problem. This is much like how the Schaefer fragments Horn, Krom, Affine [23] operate. Unlike Schaefer classes but similar to QBF, the fragment can be easily restricted to fixed alternations to get complete problems for the polynomial hierarchy.

1.1 Related Work

We note several connections between our work and other work in first-order logic. Our idea is similar to the *fluted fragment* of first-order logic [19], which gains decidability by requiring the order of quantification to match the order of variables as arguments in predicates. As we stay in Bernays-Schönfinkel, our work removes the quantifier order entirely from the classification.

Our work is directly inspired by translations from QBF to FOL, in particular translations to Bernays-Schönfinkel directly ([25]). We describe this translation in Section 2.2.1. In addition, our detection tool uncovered that our new fragment is not only an extension of a FOL translation of QBF, but can be understood as extensions of FOL translations from other PSPACE logics, in particular *basic path logic* [24], a fragment of FOL introduced as a target of the optimised functional translation [16] of certain modal logics. While the motivations behind that work are different, this is an interesting connection, that we intend to explore in future work.

It should be noted that PSPACE has been studied previously in a FOL setting. There are a number of known PSPACE-complete Schaefer fragments of EPR as detailed in Table 1. There are two common kinds of PSPACE-complete problems: problems from unbounded alternation and problems from reachability. QBF is naturally of the former kind, whereas some well-known PSPACE problems such as planning are of the latter. Krom and Det-Horn more naturally capture reachability and planning as they allow us to encode transition rules and start and target states as clauses, where unsatisfiability equals reachability. Morally, we can informally think of EPR-Krom and EPR-Det-Horn in CoNPSpace more so than PSPACE, as these were CoNL-complete propositional problems (NL=CoNL) before we took them to first order and used Savitch's Theorem [22] for the quadratic transformation to QBF. Thus we can justify the advantages of our fragment over Krom and Det-Horn. A more concrete example is that it is not clear how to restrict EPR-Krom and EPR-Det-Horn to complete classes of the polynomial hierarchy PH [27], as QBF naturally does.

Fragment	Prop. complexity	EPR complexity
3CNF	NP-complete [7]	NEXPTIME-complete [15]
Horn	P-complete [13]	EXPTIME-complete [18]
Det-Horn	NL-complete [14]	PSPACE-complete [18]
Krom (2-CNF)	NL-complete [14]	PSPACE-complete [18]
Affine	Mod ₂ -L-complete [6]	Mod ₂ -PSPACE

Table 1: Complexity of EPR fragments.

Much like Horn can be generalised into HORN-renamable, based on the symmetry of true and false. Problems that are not QEALM can become so under the application of a symmetry.

1.2 Organisation

This paper first presents the QEALM fragment (Definition 1) and proves its PSPACE-completeness (Theorem 1). Naturally, we show both inclusion (Lemma 4) and hardness in PSPACE (Lemma 6). We show that the intersection of QEALM with Schaefer fragments retains PSPACE-hardness (Theorem 3). In contrast to these Schaefer fragments, our fragment QEALM will have a game-like alternation, where we can restrict the number of alternations to specific levels of the polynomial hierarchy. Thus we show completeness for PH as well (Theorem 2). We additionally show its closure under Robinson’s resolution (Theorem 4), a property that separates it from QBF and previous translations from QBF into FOL.

2 Preliminaries

We assume familiarity with classical first-order logic [26]. In particular, terms are built from variables and function symbols; each function and predicate symbol f has a fixed arity $\mathbf{a}(f)$. Function symbols with arity 0 are called constants. A *literal* l is an *atom*, e.g., $l = r(t_1 \dots t_{\mathbf{a}(r)})$, or a negation of an atom $l = \neg r(t_1 \dots t_{\mathbf{a}(r)})$. A *clause* is a disjunction of literals, e.g., $C = l_1 \vee l_2 \vee l_3$, and a *CNF* (conjunctive normal form) is a conjunction of clauses, e.g., $C_1 \wedge C_2 \wedge C_3$, where variables occurring in the clauses are implicitly assumed to be universally quantified. The quantifiers $\forall$ and $\exists$ bind a variable x and act on a formula ϕ , i.e., $\forall x\phi$. A sentence is a formula where all variables are bound by some quantifier. A sentence is in *prenex* form if no subformula is defined by applying a connective ($\neg$, $\wedge$, $\vee$) to a quantified subformula.

The Bernays-Schoenfinkel fragment comprises sentences of the form $\exists^*\forall^*\phi$, where ϕ is quantifier-free and terms are only variables. Equivalently, via Skolemization, it is of the form $\forall^*\phi$, where terms are only constants and variables, i.e., there are no function applications but predicates are allowed, which is the view we adopt in this paper. Moreover, we can also assume ϕ is in CNF, as the standard clausification procedures preserve the fragment.

An interpretation is a tuple (D, I) where D is a non-empty set, called the domain, and I prescribes values for function and predicate symbols. Whenever convenient, we view an interpretation for a predicate as a function to $\{0, 1\}$. In Bernays-Schoenfinkel, there are no function symbols of arity greater than zero, and therefore the Herbrand universe is finite—it is the set of constant symbols that appear in the formula; this shows the decidability of Bernays-Schoenfinkel via the Herbrand theorem. Throughout the paper, we therefore assume that D is in fact the set of constants of the formula in question.

As usual, a *substitution* σ is a finite mapping from variables to terms which, when applied to a formula ϕ , gives rise to the formula $\phi\sigma$ in which each free occurrence of a variable x from the domain of σ gets (simultaneously) replaced by the corresponding term $t = \sigma(x)$ from σ ’s range. We refer to the resulting formula $\phi\sigma$ as an *instance* of ϕ , and to the operation (and where confusion cannot arise, also to the substitution itself) as *instantiation*.

Robinson’s Resolution Given a CNF formula ϕ , Robinson’s Resolution construed as a proof system consist of the Axiom rule

$$\frac{}{C} \text{ (Axiom)}$$

where C is a clause from ϕ , and two derivation rules, Resolution and Factoring, defined as follows:

$$\frac{C_1 \vee \neg p(t_1 \dots t_n) \quad C_2 \vee p(s_1 \dots s_n)}{C_1\sigma \vee C_2\sigma} \text{ (Resolution)}$$

where the atoms $p(t_1 \dots t_n)$ and $p(s_1 \dots s_n)$ are unifiable and $\sigma = \text{mgu}(p(t_1 \dots t_n), p(s_1 \dots s_n))$ is their most general unifier. (The premises C_1 and C_2 are assumed to not share variables. This can be achieved by first renaming the premises apart, if necessary.)

$$\frac{C \vee p(t_1 \dots t_n) \vee p(s_1 \dots s_n)}{C\sigma \vee p(t_1 \dots t_n)\sigma} \text{ (Factoring)}$$

where $\sigma = \text{mgu}(p(t_1 \dots t_n), p(s_1 \dots s_n))$.

Propositional resolution is a simplification of Robinson's Resolution. Assuming clauses are sets of literals, we do not need a factoring rule, and the resolution rule does not need to consider unification, the pivot atoms must match exactly. We can use propositional resolution on FOL problems that are fully ground as unification is no longer possible.

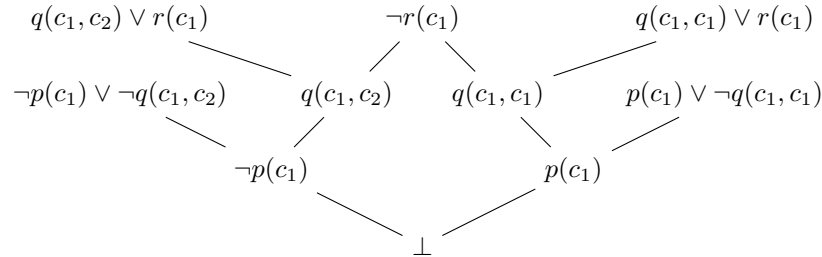
2.1 Observations on First-Order Logic

The satisfiability problem for EPR requires us to find at least one suitable interpretation of the predicate symbols. Therefore, unsatisfiability requires that there are no suitable interpretations, in other words, for every possible interpretation there is some assignment to the universal variables that makes the formula false. In the general case, the falsifying universal assignment depends on and *after* the chosen interpretation of the predicate symbols and—when considering uninterpreted predicate symbols—several different assignments might be needed to demonstrate unsatisfiability. However, there are certain scenarios where a common universal assignment falsifies all interpretations of the predicate symbols. Consider the following:

Example 1. Consider a Bernays-Schönfinkel formula, with constants c_1 and c_2 :

$$\begin{array}{l} \forall x, y \quad (p(x) \vee \neg q(x, c_1)) \quad \wedge \quad (\neg p(x) \vee \neg q(x, c_2)) \\ (\neg r(x)) \quad \wedge \quad (q(c_1, y) \vee r(c_1)) \quad \wedge \quad (q(c_2, c_1)) \end{array}$$

and a propositional resolution refutation over ground instances:



Our main observation is that the first argument in every literal is always the same ground term, in this case c_1 . This is true, due in part to resolution steps requiring matching arguments on the pivots. Generally the pivot arguments do not match with the other literals in the clause, but in this formula we have insisted on it in our axioms and it persists throughout the proof. Despite $q(c_2, c_1)$ being an axiom, it ends up not being used in the proof because it is impossible to resolve it with any clause with a mismatch in the first argument.

In contrast, the second argument of q not only can contain c_1 and c_2 in the proof, but is actually required to use both for the refutation to go through. This is because it can be joined by $p(x)$ which does not necessarily share the same argument with the second argument of q in $p(x) \vee \neg q(x, c_1)$ and $\neg p(x) \vee \neg q(x, c_2)$. In other words, because there is some connecting clause that mixes elements, both can be connected in the proof.

Observation 1. *Suppose we have a first-order problem (without equality) written as a CNF where we have the following restrictions:*

- *only allow predicate symbols which have an arity of at least 1,*
- *within each clause C among all literals throughout the clause, the first argument must be the same term t_C (whether it be a variable or other term).*

If the above conditions are met and the formula is unsatisfiable then there is at least one global ground term t , such that the following transformation produces another unsatisfiable problem

- *If possible, $C \in \phi$ is instantiated by the minimal substitution such that the first argument t_C in every literal becomes t .*
- *If not possible (t_C cannot unify with t), we remove C entirely from the CNF.*

In other words, having an argument that matches everywhere within clauses, the CNF can be simplified (with the choice of the right ground term) to an equisatisfiable case in which the argument matches everywhere within *and between* clauses via instantiation and pruning. Here we mention the *first* argument of each literal, but this can be generalised symmetrically under a rearrangement of the arguments of the predicate symbols.

While this observation is semantic in nature, our best understanding of its correctness comes from using the completeness and soundness of Resolution, as every unsatisfiable CNF has a ground refutation, and we can use that to prove the existence of t . This observation does not require us to be in Bernays-Schönfinkel, but we must forbid equality, because the conditions will automatically fail for symmetry and transitivity.

Proof. We can use the fact that for unsatisfiable first-order clausal problem, grounding a formula and using propositional resolution on the ground clauses is a sound and complete proof process. We further use the fact about trimming Resolution proofs, that proofs follow a DAG-like structure and any clauses not connected to a path to the empty clause can be trimmed.

We will prove by induction on the number of lines in a ground resolution derivation that the first argument of every literal is the ground term t .

Base Case: If we have a derivation of length 1, it just contains a single grounded axiom clause. Because of our condition, the first argument for every literal is the same ground term.

Inductive Step: If we have a derivation of clause C after n lines, we have two subderivations of length $< n$ that derive the two clauses that we resolve. So we can assume by induction hypothesis each of the subderivations have a common first argument ground term t within each subderivation. But this must also be common between the two subderivations otherwise the two pivot literals (that contain at least one argument) cannot resolve.

Therefore, by induction, all axiom ancestors of the empty clause must have the same term in the first argument for every clause. By the soundness of propositional Resolution this means there is some ground term that can replace the variable in first argument everywhere via instantiation and for us to still have a contradiction. We can then remove all clauses that are not required in the proof and it will still be a contradiction. $\square$

2.2 Quantified Boolean Formulas (QBF) and Dependency Quantified Boolean Formulas (DQBF)

The Boolean quantifiers $\forall_B$ and $\exists_B$ can be used to extend propositional logic. They are not to be used in the same way as quantifiers in first-order logic. Instead we quantify a propositional

atom symbol that appears without Boolean quantification in a formula (free) over the domain $\{\top, \perp\}$, where $\top, \perp$ are true and false in this domain (we use the subscript to distinguish it from the symbols in FOL). These formulas are called QBFs (quantified Boolean formulas) and have the following meaning:

$$\forall_B x \phi := \phi[\perp/x] \wedge \phi[\top/x] \quad \exists_B x \phi := \phi[\perp/x] \vee \phi[\top/x].$$

Like in other languages, algorithms exist to prenex a QBF. Additionally QBF satisfiability can be understood as the truth of the QBF sentence where all free propositional atoms are placed into an outer existential quantifier. This problem is PSPACE-complete. In fact, it is understood as canonically PSPACE-complete. PSPACE is a subset of NEXPTIME, therefore more solving techniques and reductions are available for QBF satisfiability than with Bernays-Schönfinkel.

One observation about QBFs is that they can be thought of as semantic games between a universal and an existential player. In such a game, the turns are given by the prefix from left to right. Whoever's quantifier appears next gets to set a Boolean value ($\{\perp, \top\}$) to replace the quantified atom. At the end of the game, the universal player wins if the resulting expression evaluates to $\perp$ and the existential player wins if the resulting expression evaluates to $\top$. A QBF is true if and only if the existential player has a winning strategy and false if and only if the universal player has a winning strategy. These strategies can be thought of as the Skolem and Herbrand functions, respectively, which are now Boolean functions. A QBF is true (false) if and only if there are Skolem (Herbrand) functions for the existentially (universally) quantified atoms and the formula becomes a tautology (contradiction) after substituting in the Skolem (Herbrand) functions. The quantifier order matters as it affects the size of the domain of the Skolem (Herbrand) functions.

2.2.1 DQBF and the connection to EPR

One extension of prenex QBF that is relevant to this paper is (S-form) DQBFs (Dependency QBFs). In DQBF we once again have a propositional problem with a quantifier prefix, however there is no intrinsic order of the prefix. Instead, S-form DQBFs use Henkin quantifiers for $\exists_B$ that explicitly specify the universal atom symbols that the quantified existential atom depends on. The *dependency set* D_e denotes the set of universal variables that existential variable e depends on. A DQBF is still true if and only if it is a tautology after Skolemisation of the existential variables and every QBF can be written as a DQBF. But the linear two player game can no longer apply in general, so it is difficult to make a simple branching procedure. DQBF turns out to be NEXPTIME-complete and equivalent to the Bernays-Schönfinkel fragment.

Let us now describe a transformation [25] from (D)QBF to EPR that uses Skolemisation. Translating DQBF truth ($\top$) and falsity ($\perp$) as FOL constants **true** and **false**, we can express the Skolemisation using a first-order predicate symbol p and function symbols for Skolem functions:

$$\begin{aligned} [\top]_p &= p(\mathbf{true}), & [e]_p &= p(f_e(D_e)), & [u]_p &= p(u), & [\phi_1 \vee \phi_2]_p &= [\phi_1]_p \vee [\phi_2]_p, \\ [\perp]_p &= p(\mathbf{false}), & [\neg e]_p &= \neg p(f_e(D_e)), & [\neg u]_p &= \neg p(u), & [\phi_1 \wedge \phi_2]_p &= [\phi_1]_p \wedge [\phi_2]_p, \end{aligned}$$

where u is a universal variable and e an existential variable with its dependency set D_e . For each existential e there is a newly introduced symbol f_e of arity $|D_e|$ and its arguments in $f_e(D_e)$ are the corresponding universal variables from D_e in some fixed order.

Suppose we have a DQBF with universal variables U , existential variables E and propositional matrix ϕ , then we can express the problem of whether the Skolemisation results in a tautology in the following way in FOL: $\forall U[\phi]_p \wedge \neg p(\mathbf{false})$.

This is not yet in the Bernays-Schönfinkel fragment as we have function symbols for the Skolem functions. These can easily be transformed into predicate symbols as they always occur immediately inside predicate p . Thus $p \circ f_x$ can be replaced by its own predicate symbol $x(\dots)$ of arity $|D_x|$. Once we complete this for all Skolem functions, the problem lands in EPR.¹

Finally, we can refine the formulas to get rid of the predicate symbol p by resolving away all literals involving p against the units $p(\text{true})$, $\neg p(\text{false})$.² This leaves all predicate symbols to be the Skolem functions and the arguments among literals will belong to a certain structure. We generalise this structure for our PSPACE fragment QEALM in the next section.

3 A QBF-like fragment of Bernays-Schönfinkel

If the Bernays-Schönfinkel fragment corresponds to the DQBF problem, what fragment of first-order logic corresponds to QBF? Some obvious conditions would be that it would itself be a fragment of Bernays-Schönfinkel and then since QBF is PSPACE-complete this fragment we would want this also to be PSPACE-complete.

We define our fragment utilising the condition of Observation 1, which allows us to have a QBF-like semantic game. For this, we need to fix an ordering of each predicate's arguments. We will see later that there are some practical opportunities to relax this ordering (Definition 7).

Definition 1 (QBF-like EPR via Alternating Level-ordered Miniscoping (QEALM-fragment)). *A Bernays-Schönfinkel instance is in the QEALM-fragment if the following holds:*

In each clause if a variable symbol u appears in multiple literals $l_1 \dots l_k$ (this is not necessarily all literals in the clause),

1. *the first position it occurs (i.e. the lowest index i) is the same in all $l_1 \dots l_k$,*
2. *for all $0 \leq j \leq i$ for any $1 \leq p \leq q \leq k$ the j -th argument of l_p must be equal to the j -th argument of l_q , i.e., there is a shared initial segment.*

Example 2. *We take the Bernays-Schönfinkel problem with constants c_1, c_2 :*

$$\begin{aligned} & \forall u_1, u_2, u_3, u'_3, u_4 \quad (y(u_1, u_2, u_3, u_4)) \wedge \\ & (\neg x(u_1, c_1, u_3, c_1) \vee \neg y(u_1, c_1, u_3, u_4) \vee \neg x(u_1, c_1, u'_3, u_1)) \wedge \\ & (x(u_1, u_1, u_3, c_1) \vee \neg y(u_1, u_1, u_3, u_4) \vee x(u_1, u_1, c_2, u_1)) \end{aligned}$$

Each clause has a prefix tree. For example in the second and third clauses are described by trees in Figure 1.

The arguments of the literals appear as nodes on tree. Here all literal arguments appear on the leaves. Variables that are first introduced on one branch such as u_4 cannot be introduced in another branch as the same variable, however duplicate variables from earlier in the tree can be reused, and constant symbols can appear on any branch. The leaves are the literals, we adopt the convention of a distinction between different unique literals, so when the arguments are the same but we use different predicate symbols we make a final branching.

The idea is this tree is traversed over the course of a two player game similar to the game found in QBF. In QBF the game follows the linearly ordered quantifiers (for prenex formulas).

¹To see this transformation preserves satisfiability, we interpret x via $p \circ f_x$ and, in the less obvious direction, use the fact that only a two-valued domain can be used (cf. Lemma 3 in [25]).

²Again it is easy to see this is a sound step (via soundness of resolution), but the fact it does not introduce spurious counterexamples is less obvious. An indirect argument goes via completeness of the QBF calculus $\forall\text{Exp}+\text{Res}$ [12, 4].

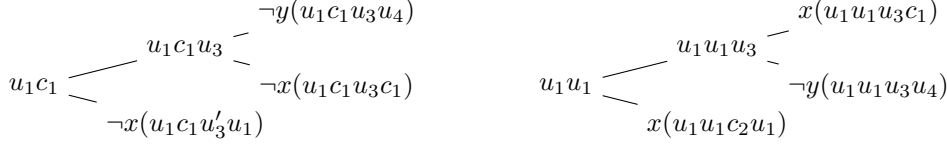


Figure 1: The prefix trees governing the second (left) and third (right) clauses in Example 2

The tree branching dictates the existential plays choices, while the value of the universal variables along those branches is decided by the universal player. Because we intend to solve such formulas by instantiating the variables in order, after the first occurrence repeated variables act more like constants and this is reflected in the definition of membership in the QEALM-fragment.

The fragment is a generalisation of how QBFs (written as DQBFs) translate under the transformation in Section 2.2.1. However one key difference, is that the literals can exhibit variable disjointness within a clause. This is not really necessary for PSPACE-hardness, as translated QBFs are already PSPACE hard, but it will be necessary later in Theorem 4 for closure under Robinson’s Resolution.

Let us now show that the QEALM-fragment has an algorithm that decides its satisfiability in PSPACE. The idea of the algorithm is to recursively alternate between a universal loop that checks instantiations for falsity, and an existential loop that queries branches on disjunctions while checking for satisfiability. The branching for variable disjointness seen in Example 2 and Figure 1 is an important feature, because miniscoping can split the quantifiers up onto separate subclauses and then an existential loop can check whether any miniscoped subclauses are satisfiable. Likewise, a non-empty set of arguments shared among the literals of a clause will allow the universal loop to query a joint instantiation to the shared variables. Before we describe how this works, let us first define the variables that the universal loop of the algorithm will attempt to assign.

Definition 2 (Outer Variables). *Outer variables of a clause are those that appear in every literal of the clause.*

These are named outer variables, because—as we will see—while all other variables can be handled by nested miniscoping, the outer variables necessarily scope the entire clause. We eventually want variables that scope the entire CNF, but because conjunction commutes with the universal quantifier, variables in different clauses may as well be completely different. Instead we look for positions of arguments that are amenable to Observation 1.

Definition 3 (Outer Position).

- For a clause C , i is an outer position ($i \in \text{OutPos}(C)$) if and only if there is an outer variable u such that i is the least position of an occurrence of u in any literal in C .
- For a clause C , i is a prefix position ($i \in \text{PrePos}(C)$) if and only if C is a singleton, or for each $j \leq i$ there is a term t_j that is either a constant or an outer variable, such that the j -th position of every literal in C contains t_j .
- We overload the notation $\text{OutPos}(\phi)$ for a CNF ϕ . Position i is an outer position for CNF ϕ ($i \in \text{OutPos}(\phi)$) if there is some clause C where i is an outer position, and in all other clauses D , position i is a prefix position ($\exists C \in \phi$ s.t. $i \in \text{OutPos}(C)$ and $i \in \bigcap_{D \in \phi} \text{PrePos}(D)$).

Global outer positions are those that appear outer in at least one clause, but they must be in the joint argument prefix of every other clause otherwise they might be assigned prematurely.

Example 3. Let $u_1 \dots$ denote universal variables, $c_1 \dots$ denote constant symbols and $p_1 \dots$ denote predicate symbols.

$$\begin{aligned} \forall u_1, u_2, u_3, u_4 \quad & (p_1(u_1, c_1, u_1, u_2) \vee p_2(u_1, c_1, u_1, u_2, c_2) \vee p_3(u_1, c_1, u_1, u_2, u_3, u_4)) \wedge \\ & \forall u_1 \quad (\neg p_1(c_1, u_1, c_3, u_4) \vee \neg p_2(c_1, u_1, c_4, c_3, c_1)) \end{aligned}$$

In the first clause, positions 1, 2, 3 and 4 are prefix and positions 1 and 4 are outer. In the second clause position 1 is prefix and position 2 is outer. Therefore positions 1 and 2 are outer globally.

Lemma 1. Suppose we have a CNF ϕ in the QEALM-fragment and for every $i \in \text{OutPos}(\phi)$, we have a constant symbol c_i . Let us build a substitution σ such that for every variable u that appears in some argument $i \in \text{OutPos}(\phi)$ for some literal occurring in ϕ , we replace u with c_i . When we apply that substitution to ϕ , the resulting $\bigwedge_{C \in \phi} C\sigma$ remains in QEALM-fragment.

Proof. When we instantiate a variable with a constant, we replace it everywhere. Therefore any initial argument that agreed across literals before the instantiation remains in agreement. Since we replace variables, we do not create any more cases that need to be checked. Furthermore, when we replace a variable, we replace all the variable's repetitions in a literal. $\square$

These definitions are needed for the universal loop. The following are for the existential.

Definition 4 (Inseparable Clause). An inseparable clause K is either a singleton literal or requires that there is some variable that is present in all literals of K . In the QEALM-fragment, this is the same as every pair of literals sharing a variable.

Definition 5 (Component). A component³ K of C is a non-empty inseparable subclause, where if literal a occurs in K then if there is any literal b that occurs in C that shares a variable with a then $b \in K$.

Example 4. In clause $a(u, v, w) \vee b(u, v) \vee p(x) \vee q()$, the components are $a(u, v, w) \vee b(u, v)$, $p(x)$ and $q()$. For example, subclause $a(u, v, w) \vee b(u, v)$ is an inseparable clause that has outer variables u and v .

$\forall u, v, w, x \quad a(u, v, w) \vee b(u, v) \vee p(x) \vee q()$ carries the same meaning as the miniscoped formula $(\forall u, v, w \quad a(u, v, w) \vee b(u, v)) \vee (\forall x \quad p(x)) \vee q()$.

Lemma 2. Given a CNF $\phi \wedge C$ in QEALM-fragment with a clause C . If $\phi \wedge C$ is satisfiable, then there is some component K of C such that $\phi \wedge K$ is satisfiable.

Proof. We rely on the following two propositions:

1. Two distinct components of a clause do not share variables.
2. A clause in QEALM-fragment is the disjunction of its components.

Proposition 1 is true because if we have two inseparable subclauses K_1 and K_2 , internally all literals share a variable, and then when $l_1 \in K_1$ shares a variable with $l_2 \in K_2$ it means $l_1 \in K_2$ and $l_2 \in K_1$, but then by the QEALM conditions, there will be enough shared prefixes that force that all literals of K_1 must be in K_2 and vice versa. Proposition 2 is true because singletons are

³We borrow this notion from AVATAR [29], where components are introduced as the maximal non-splittable subclauses.

inseparable subclauses and if we take the closure of literal sets under the operation of including a literal that shares a variable, the first variable argument to appear is always the same, hence the maximal closure of each literal is an inseparable subclause.

Now suppose $\phi \wedge C$ is satisfiable, then we take the interpretation of the predicate symbols that satisfies it. Now suppose, for contradiction, that for each component K of C , there is some universal assignment α_K that falsifies it under our interpretation. Then we can restrict α_K to just the variables of K to get β_K . The union of all β_K over every component K will be a disjoint union and must falsify every K , and hence falsifies the clause C . By contradiction there must be at least one component K that is always satisfied. Therefore $\phi \wedge K$ is satisfiable. $\square$

We present Algorithm 1, which decides the satisfiability of the QEALM-fragment.

Algorithm 1: Recursive algorithm for satisfiability of the QEALM-fragment

```

Function SolveByAlternation( $\phi$ ):
  if  $\phi$  is quantifier-free then
    return Sat( $\phi$ )
  if all clauses are inseparable then
    // Universal search
    for  $\sigma \leftarrow$  total assignments at OutPos( $\phi$ ) do
       $\phi|_\sigma \leftarrow \bigwedge_{C \in \phi} C\sigma$  // instantiate  $\phi$ 
      if SolveByAlternation( $\phi|_\sigma$ ) = UNSAT then
        return UNSAT
    return SAT
  else
    // Existential search
    for  $\tau \leftarrow$  one component per clause do
       $\phi|_\tau \leftarrow \bigwedge_{C \in \tau} C$  // select one component per clause
      if SolveByAlternation( $\phi|_\tau$ ) = SAT then
        return SAT
    return UNSAT

```

Example 5. We can take the Bernays-Schönfinkel problem from Example 2 with constants c_1, c_2 . It is safe to use the universal quantification on all clauses individually. We can imagine Algorithm 1 running like a game between a universal and existential player. Whose turn it is, depends whether all clauses are inseparable or not.

$$\begin{aligned}
& \forall u_1, u_2, u_3, u_4 \quad (y(u_1, u_2, u_3, u_4)) \wedge \\
& \forall u_1, u_3, u'_3, u_4 \quad (\neg x(u_1, c_1, u_3, c_1) \vee \neg y(u_1, c_1, u_3, u_4) \vee \neg x(u_1, c_1, u'_3, u_1)) \wedge \\
& \forall u_1, u_3, u_4 \quad (x(u_1, u_1, u_3, c_1) \vee \neg y(u_1, u_1, u_3, u_4) \vee x(u_1, u_1, c_2, u_1))
\end{aligned}$$

If we follow Algorithm 1, we first observe that all clauses are inseparable and that the first and second position always contain the same argument within clauses. Therefore the universal player chooses the value of the variables appearing on the first two arguments. Technically, the various copies of u_1 across clauses could be different variables since each clause may as well have its own quantifiers, but it never helps the universal player to assign them differently when

they are outer variables (see Observation 1). The meaning of the outer variables is that we can take them outside of the formula:

$$\begin{aligned} & \forall u_1, u_2 (\\ & \quad \forall u_3, u_4 (y(u_1, u_2, u_3, u_4)) \wedge \\ & \quad \forall u_3, u'_3, u_4 (\neg x(u_1, c_1, u_3, c_1) \vee \neg y(u_1, c_1, u_3, u_4) \vee \neg x(u_1, c_1, u'_3, u_1)) \wedge \\ & \quad \forall u_3, u_4 (x(u_1, u_1, u_3, c_1) \vee \neg y(u_1, u_1, u_3, u_4) \vee x(u_1, u_1, c_2, u_1))) \end{aligned}$$

For a universal assignment we assign both the first and second argument (u_1 and u_2) to c_1 and the second argument. Each application of a universal assignment yields another problem still in the fragment.

$$\begin{aligned} & \forall u_3, u_4 (y(c_1, c_1, u_3, u_4)) \wedge \\ & \forall u_3, u'_3, u_4 (\neg x(c_1, c_1, u_3, c_1) \vee \neg y(c_1, c_1, u_3, u_4) \vee \neg x(c_1, c_1, u'_3, c_1)) \wedge \\ & \forall u_3, u_4 (x(c_1, c_1, u_3, c_1) \vee \neg y(c_1, c_1, u_3, u_4) \vee x(c_1, c_1, c_2, c_1)) \end{aligned}$$

At this point some clauses are separable into components, this means we can rewrite the problem using miniscoping.

$$\begin{aligned} & (\forall u_3, u_4 y(c_1, c_1, u_3, u_4)) \wedge \\ & (\forall u_3, u_4 \neg x(c_1, c_1, u_3, c_1) \vee \neg y(c_1, c_1, u_3, u_4)) \vee (\forall u'_3 \neg x(c_1, c_1, u'_3, c_1)) \wedge \\ & (\forall u_3, u_4 x(c_1, c_1, u_3, c_1) \vee \neg y(c_1, c_1, u_3, u_4)) \vee (x(c_1, c_1, c_2, c_1)) \end{aligned}$$

Now it is fine for the existential player to choose the disjuncts he will use to gain a satisfiable ground problem. There are $1 \times 2 \times 2$ options this time. After choosing the disjunct we remain in the fragment, and now return to every clause being inseparable. However unfortunately for the existential player no matter the choice here, if we continue to play this game the universal player can force a contradiction.

One alternative formulation is to understand components as abstracted to propositional literals and we loop through assignments to those variables during the existential search. This formulation may lead to more interesting solving approaches (i.e. clause learning or CEGAR: counter-example guided abstraction and refinement), but we do not need it here to show correctness and constraint learning complicates our incoming argument about space complexity.

Lemma 3 (Correctness). *If ϕ is a QEALM-fragment problem that does not contain the empty clause then $\text{SolveByAlternation}(\phi)$ (Algorithm 1) returns SAT if and only if there is a satisfying interpretation for ϕ .*

Proof. We can perform induction on the number of quantified variables + predicate symbols appearing in the formula

Base Case: Fully Ground. If ϕ is ground and $\text{SolveByAlternation}(\phi)$ returns SAT then there is a satisfying Boolean assignment to the occurring ground literals. This can be used to construct the truth table for the predicates, and these predicates will satisfy ϕ . If ϕ is ground and $\text{SolveByAlternation}(\phi)$ returns UNSAT then there is no satisfying Boolean assignment to the ground values, hence there is no truth table interpretation of the predicates which does not falsify ϕ , and a predicate interpretation needs a truth table to exist, so there is no satisfying interpretation.

Inductive Step: Universal Search. If ϕ has a satisfying interpretation then the same satisfying interpretation satisfies all clauses under instantiation, therefore $\text{SolveByAlternation}(\phi)$ returns SAT after it exhausts all assignments to the variables at $\text{OutPos}(\phi)$. If ϕ has no satisfying assignment then it has a Ground Resolution refutation, however said refutation cannot mix variables from different instantiations of the fully unified argument prefix (there are no resolution paths see Obs 1). Therefore, there is some instantiation of the S variables where ϕ has no satisfying interpretation and hence $\text{SolveByAlternation}(\phi)$ returns UNSAT.

Inductive Step: Existential Search. If ϕ is unsatisfiable then any problem composed of subclauses of ϕ is also unsatisfiable, therefore $\text{SolveByAlternation}(\phi)$ returns UNSAT. Now suppose ϕ has a satisfying interpretation. We can take ϕ out of its prefix form. Since $\forall$ and $\wedge$ commute, each clause could have its own quantification, then we push all universal quantifiers to the level of the components. Lemma 2 is important because components are disjoint and quantified variables are not shared between them. So we have a disjunction where each disjunct is universally quantified. The satisfying interpretation must satisfy at least one disjunct in each conjunct, and every assignment of disjuncts gets selected in the loop. $\square$

Lemma 4 (PSPACE inclusion). *Given a suitable SAT solver black box, the amount of memory of Algorithm 1 is bounded from above by a polynomial in the size of the input CNF ϕ , if ϕ is a QEALM-fragment problem.*

Proof. The algorithm requires a stack of recursive calls. The stack size of recursive calls from a universal search is bounded by the largest arity of a predicate or the number of universal quantifiers, since each instantiation sets a constant value to at least one variable that is shared in all literals. The stack of the recursive calls from an existential search is bounded by the number of times we can split at least one clause, so an upper bound is the total number of literals. Therefore there are at most a linear number of recursive calls at any one time. Moreover, each recursive call only uses a polynomial amount of memory for the loops, propositional logic and the SAT call. Hence the algorithm uses PSPACE. $\square$

This algorithm essentially saves memory because Observation 1 allows us to query the outer variables of different clauses jointly. The QEALM condition allows us to recursively remain in the position where we can either apply miniscoping or Observation 1 and afterwards remain in QEALM. Without the QEALM conditions, we may eventually arrive at a position where we can apply neither. In fact, with a bit more detail we can do better than just PSPACE and identify the fragments that belong in classes of the polynomial hierarchy. The idea is to identify which argument positions may force an existential search in Algorithm 1. If we bound the number of times an existential search happens between universal queries we bound the number of alternations by roughly twice of that.

Definition 6. *Let Φ be a problem in the QEALM-fragment. Position $i > 0$ is a fork index if there is a clause $C \in \Phi$ and a pair of literals l_1 and l_2 in C (not necessarily distinct) such that i is the minimal value such that all variables shared between l_1 and l_2 occur as arguments in positions $\leq i$.*

Example 6. *Let $u_1 \dots$ denote universal variables, $c_1 \dots$ denote constant symbols and $p_1 \dots$ denote predicate symbols. For $\forall u_1 u_2 u_3 (p_1(u_1, u_2) \vee p_2(u_1, u_2, c_1, u_3)) \wedge (\neg p_2(u_1, u_2, u_1, u_3))$. The fork indices are 2 and 4.*

After a universal search the least current fork index is removed because joint variables are instantiated, whereas an existential search will result in no new fork index.

Lemma 5 ($\Sigma_{2k-1}^P, \Pi_{2k}^P$ inclusion). *Given a set $\{\Phi_j\}$ of problems in the QEALM-fragment, let the number of fork indices be bound by k . Then $\{\Phi_j\}$ is a family in the complexity class:*

- Π_n^P if each formula of $\{\Phi_j\}$ has only inseparable clauses, where $n = 2k$.
- Σ_n^P in all $\{\Phi_j\}$, where $n = 2k + 1$, otherwise.

Proof. By induction on the maximum number of recursive calls of Algorithm 1 would need to solve it, which is a finite value by Lemma 4.

Base Case: When there is no more recursive calls, there are no quantified variables and no fork indices. Thus we can solve the problem as a Σ_1^P problem with SAT solving. If all clauses are inseparable then there are no disjunctions and we fall into $\text{PTIME} = \Pi_0^P$.

Inductive Step (Π_{2k}^P case): Suppose all clauses, of the matrix ϕ are inseparable. Assume we are not in the base case, and there is at least one variable, so there is some minimum fork index $m > 0$. For every clause $C \in \phi$ and every $0 < j \leq m$ must be a prefix position, otherwise due to the definition of QEALM there is a fork index strictly less than j . Suppose we have some literal $l \in C$ that contains a variable v at argument position $0 < j \leq m$, since this is a prefix position it must either be an outer position for C or be some repetition of a v that occurred first at argument $i \leq j$. Whatever the case, during the universal search in Algorithm 1, σ will assign this variable to a constant. Therefore all arguments up to and including m for all literals in all clauses in $\bigwedge_{C \in \phi} C\sigma$ will not be variables. Therefore m is no longer a fork index in $\bigwedge_{C \in \phi} C\sigma$.

Suppose $p > 0$ is a fork index in some clause $C\sigma$ then we have a pair of literals $l_1\sigma$ and $l_2\sigma$ that share all common variables minimally after the p th argument. But in that case there is some variable u occurring at p that is not assigned by σ , u must be present in l_1 and l_2 . The only way p cannot be a fork index in C is if there is some variable v shared between l_1 and l_2 that first occurs after p at argument position q that is assigned by σ . That means q is a prefix position in all clauses, so p must be prefix in all clause. q must be outer in C . So p must also be outer in C , so p would not be a fork index after instantiation as presumed.

Thus we decrease the number of forks indices by 1, but now have clauses that are not inseparable therefore $\bigwedge_{C \in \phi} C\sigma$ is always in Σ_{2k+1-2}^P via the induction hypothesis. Since, we query over a polynomial size σ to find a Σ_{k+1}^P to falsify, deciding ϕ is in Π_{2k}^P .

Induction Hypothesis (Σ_{2k+1}^P case): Suppose we have a number of separable clause. In the existential search of Algorithm 1, we filter this to selecting individual components, all clauses in $\bigwedge_{C \in \tau} C$ are inseparable, but we create no new fork indices as each clause is a subset of version in ψ , and so we consider a subset of the set of pairs of literals to find the fork indices. Therefore $\bigwedge_{C \in \tau} C$ is a Π_{2k}^P problem. Since we loop over the choice of (polynomial size) τ and accept when one selection works we are in Σ_{2k+1}^P . $\square$

We can take inclusion further and show completeness; first in the PSPACE case and then for the classes of the polynomial hierarchy.

Lemma 6 (PSPACE-hardness). *Any closed (totally quantified) QBF problem can be transformed into a QEALM-fragment problem with polynomial time computable function f , where $f(\Psi)$ is satisfiable if and only if the QBF Ψ is true.*

Sketch Proof. For f , we use the transformation detailed in Section 2.2.1 up to the end including the removal of the first order interpretation of universal literals. We only consider prenex QBFs in CNF (without tautological clauses) but this will be sufficient for PSPACE-hardness. Note here that our domain is QBFs not DQBF, so the dependency set of existential variables are those universal variables left of it. A prenex QBF is true if and only if it has a set of satisfying Skolem functions, which is precisely what is encoded in the transformation. $\square$

Theorem 1. *The QEALM-fragment is PSPACE-complete.*

Proof. For hardness we have the transformation in Lemma 6. For inclusion in PSPACE we use Algorithm 1, except where there is an empty clause then the formula is false. The properties of Lemma 3 and 4 complete the argument. $\square$

Theorem 2 ($\Sigma_{2k-1}^P, \Pi_{2k}^P$ -completeness). *The following is true:*

- *The set of all QEALM-fragment problems which have 0 as a fork index and k or fewer fork indices in total is Σ_{2k-1}^P -complete.*
- *The set of all QEALM-fragment problems which have k or fewer fork indices in total is Π_{2k}^P -complete.*

Proof. Using Lemma 5 we get inclusion. Hardness can be shown by transforming a Prenex QBF with a bounded number of alternations to QEALM-fragment and while staying within the boundaries of side conditions in the Theorem statement.

Suppose we have a Prenex QBF which starts with an existential quantifier and then has $2k - 1$ quantifier blocks $\exists_B X_1 \forall_B U_2 \exists_B X_2 \dots \exists_B X_k$ (this is Σ_{2k-1}^P -complete) or we start with a universal quantifier and have $2k$ quantifier blocks $\forall_B U_1 \exists_B X_1 \forall_B U_2 \exists_B X_2 \dots \exists_B X_k$ (this is Π_{2k}^P -complete). In either case let it end with an existential block. After the prefix, there is the propositional part known as the matrix. We can use a Tseitin transformation to change it into conjunctive normal form. The transformation introduces new existentially quantified variables that are to be placed in the final existential block after all other variables. This preserves satisfiability, as any winning strategies can be extended to play the new variables according to their definitions. Therefore, after the Tseitin transformation, we retain the same number of quantifier blocks.

We can then use the conversion to the QEALM-fragment as given in Section 2.2.1, after which we observe the following: 0 is a fork index if and only if the QBF begins with an existential quantifier and $i > 0$ is a fork index if and only if there is some $m > 0$ such that the sum $\sum_{j=1}^m |U_j| = i$. Hence there are going to be k fork indices in total and the condition on the 0-th index is also respected. $\square$

The above hardness results, use the accessible transformation using `qbf2epr`, other translations can be used to prove stronger results. Namely, that the intersection of Krom, Horn, Affine and the QEALM-fragment remains PSPACE-complete.

Theorem 3. *The QEALM-fragment is still PSPACE-hard with the following restrictions.*

1. *Every clause has size at most 2 (is Krom).*
2. *Every clause has at most one positive literal (is Horn).*
3. *For every 2-element clause $a \rightarrow b$, also the opposite implication $b \rightarrow a$ is present among the clauses (is Affine).*

On the other hand we can relax the ordering found in the QEALM-fragment, to categorise a wider range of PSPACE problems that may be of practical interest.

Definition 7 (QEALM re-arrangeable). *A Bernays-Schönfinkel problem ϕ is QEALM re-arrangeable if for every predicate symbol p there is a permutation $s_p : [a(p)] \rightarrow [a(p)]$, such that if we rearrange the order of arguments of every occurrence of every p and do so for every predicate, then we have a QEALM problem.*

QEALM re-arrangeability is a similar notion to HORN-renameability. The satisfiability of QEALM re-arrangeability is also PSPACE-complete, because we can enumerate all possible symmetries with one additional alternation. While the detection of whether an instance is in QEALM is in polynomial time in the worst case, the detection of QEALM re-arrangeable problems may not be. But we would hope in practice there will be many easily detectable cases.

Finally, on the theoretical side, we present an important first-order closure property.

Theorem 4. *The QEALM-fragment is closed under the rules of Robinson's resolution.*

Proof. We can argue that Robinson's rules Resolution and Factoring preserve this class.

Resolution. Let us use uppercase C, D, E for clauses, p, q, r for predicate symbols and lower-case greek letters for argument sets. Suppose $C \vee p(\alpha)$ resolves with $D \vee \neg p(\beta)$ to get E , let us take two literals $q(\gamma)$ and $r(\delta)$ in E and suppose they have some shared variable u , the first occurrence of u in q is argument number i , and in r is j . This can only occur if these first u occurrences 1) were equal before the resolution, or 2) unified as a result of the resolution step.

Before instantiation $q(\gamma)$ and $r(\delta)$ exist as literals $q(\epsilon)$ and $r(\zeta)$. We generally assume that two clauses share no variables between them until after unification. So case 1 only occurs when the $q(\epsilon)$ and $r(\zeta)$ already appear in the same clause and so $i = j$ and all variables $< i$ are already the same by QEALM rules. The further instantiation that happens inside the resolution step cannot make these variables different. In the second case the unification occurs because of the instantiation of the two clauses, the clause $C \vee p(\alpha)$ is instantiated to effect the variables of α and all other variables in C that are equal to those in α , likewise with D and β .

We then argue in case 2 that $q(\epsilon)$ and $r(\zeta)$ are not from the same clause. WLOG we assume that they both are in C and $i \leq j$. Let v be the i -th argument of ϵ and w be the j -th argument of ζ , at least one of these has to change as a result of resolution to match the other, but this only occurs if both v and w occur in α otherwise the unifier is not the most general. Then by the definition of QEALM-fragment and our assumption that $i \leq j$, v must occur in ζ and thus we are in Case 1. So without loss of generality $q(\epsilon)$ is in C and $r(\zeta)$ is in D and ϵ has a variable v which also occurs in α (occurring first in the i -th argument) and ζ has a variable w also occurring in β (first in the j -th argument) and v and w will become the same variable u .

In order for v and w to both become the same variable they need to be part of some branch of variables that share a common argument index across α and β . Such a path will always unify to the same term, and if a constant term appears in the path it cannot be a variable. If we suppose $i < j$ there is some term t in the i -th argument of ζ (and thus in the i -th argument of β) that does not become u . If t is constant then v cannot become u under the unification, so t must be a variable, and it must become u , like v under instantiation and therefore $j \leq i$. This works symmetrically and so $j = i$ is the only possibility. Therefore for $k < i$ all the arguments of $q(\epsilon)$ in C are the same as the arguments of $p(\alpha)$ and likewise all the $k < i$ arguments of $r(\zeta)$ in D are the same as $\neg p(\beta)$. Therefore after instantiation these also become equal.

Factoring. In factoring, the clause is most generally instantiated so that two literals $p(\alpha)$ and $p(\beta)$ under the same predicate p can become identically unified to $p(\gamma)$. Because of the condition of the fragment, there is some maximal i (possibly zero) such that for $j \leq i$ all variables already are identical and for $j > i$ all variables are different. Therefore the instantiation will only occur on the arguments $j > i$ that correspond to these different variables. Thus for any other literal $q(\delta)$ in the clause they become affected if they share a variable with $p(\alpha)$ (or $p(\beta)$ but WLOG let us assume $p(\alpha)$), in which case the first argument $j > i$ where the variable appears is the same. All previous variables to j in δ are already the same as in α . Furthermore those δ variables before argument i already match with β . Any of the δ variables before j get the identical treatment to α and agree with γ . This means if any pair of literals that shares a

variable u is created from this instantiation they will have already agreed on the variables $k \leq i$ and will be instantiated to equal on the variables after i up to the argument containing u . $\square$

Krom and Horn fragments of Bernays-Schönfinkel are also closed under Robinson’s resolution. QEALM re-arrangeability is also closed under Robinson’s resolution as the rules commute with the application of global symmetries on the arguments. Note that the translation of QBF in Lemma 6 can not be closed under Robinson’s resolution, because resolution steps produce formulas in QEALM that have no preimage in the QBF translation. The original translation `qbf2epr` is also not closed under Robinson’s resolution.

4 Software and Experimentation

4.1 Implementing a QEALM Detector

In the TPTP library [28] version 9.1.0, in its CNF subset, there are 936 Bernays-Schönfinkel problems. We wrote a small extension of the Vampire theorem prover [1] (mainly, to reuse its efficient parser) to test for membership in QEALM and establish the number of fork indices.⁴

We discovered 308 QEALM-fragment problems of which 50 are purely propositional ($\#$ fork indices ≤ 1), 109 monadic ($\#$ fork indices $= 2$), and 49 contain predicates with up to two “variable-active” argument positions ($\#$ fork indices $= 3$).⁵ Many of these last named 49 problems are ALC problems translated from propositional multi-modal K logic formulae generated according to the scheme described by Hustadt and Schmidt [10] (e.g., the problem SYN521-1). The problems with higher (in principle unbounded) number of fork indices are from the following two families:

- binary counter formulas by Pérez and Voronkov [17] (which are also Krom), e.g., the problem MSC015-1.005, and
- optimised functional translations of modal QBF formulae by Hustadt and Schmidt [11], e.g., the problem SYN870-1.

It is possible that more thorough preprocessing of the input would detect more examples. We only used the default argument ordering of the predicates. We also remark that our procedure does not detect 81 Bernays-Schönfinkel problems translated to TPTP by `qbf2epr` [25]. Strictly speaking, they do not belong to the QEALM-fragment, although by eliminating the two—always present—clauses $p(\text{true})$, $\neg p(\text{false})$, by exhaustively resolving against all other p -literals as described in Sect. 2.2.1 (and simply dropping the superfluous clause $\text{true} \neq \text{false}$), the membership could be recovered.

4.2 Solving QEALM with Vampire

To probe the landscape of the discovered QEALM problems in TPTP (cf. Sect. 4.1), we tried to solve each of them using Vampire. The full results are explained in an extended version of this paper⁶. Here are our two main findings:

⁴Available at <https://github.com/vprover/vampire/tree/martin-epr-fragment>.

⁵For example, the problem MGT041-2 from this group contains a predicate of arity 3, but there a variable never occurs at its third argument position.

⁶Available at <https://arxiv.org/abs/2602.16410>

- While Vampire’s default strategy exhibits exponential behaviour on the binary counter formulas and produces the obvious ground proof which “actually does the counting” (by deriving exponentially many unit clauses), by changing the ordering and literal selection function⁷ the prover can be nudged to run in what appears to be polynomial time (roughly $\Theta(k^{3.5})$, where k is the number of the counter’s bits) and to produce non-ground proofs which avoid the exponential worst-case.
- On the translations of modal QBF formulae, the number of fork indices is a good predictor of the effort needed to solve the formula by saturation (unlike C , the number of clauses in the source QBF in the TPTP header): Over the 90 of 92 such problems that Vampire solved under a fixed budget of $1.28 \cdot 10^{11}$ instructions, the Spearman rank correlation between the number of fork indices and the logarithm of the instructions burned is $\rho = +0.76$ ($p < 10^{-17}$), the median cost rising from $548 \cdot 10^6$ instructions at 21 fork indices to $25\,577 \cdot 10^6$ at 57, whereas for C it is $\rho = -0.05$ ($p = 0.64$).

Beyond these two findings, the space behaviour of the prover deserves a remark. Across the whole collection, Vampire’s peak memory stayed in the tens of megabytes on the large majority of the problems, and it grew markedly more slowly than the running time: over the range of fork indices spanned by the QBF translations, the instructions burned grow by an order of magnitude more than the memory consumed. We do not offer this as empirical evidence of the PSPACE bound—saturation carries no space guarantee, and the runs that exhausted the instruction budget did so while occupying hundreds of megabytes— but it does suggest that on QEALM-fragment problems the clause sets that a saturation-based prover must keep in memory remain modest, and that time rather than space is the binding resource in practice.

Currently, there are no dedicated practical QEALM solvers. Algorithm 1 presents a PSPACE approach, but lacks any heuristics and optimisations. A positive indication that we can solve QEALM problems more practically, is the closeness to QBF. There are of course a number of directions that can be taken in future steps, all of which can be inspired by QBF solving.

5 Conclusion

This work widens our understanding of first-order logic by finding a new PSPACE-complete fragment, that shares many similarities to QBF. Remarkably, despite not being defined on propositional connectives, the QEALM-fragment has multiple properties shared by Krom and Horn. QEALM instances can be detected per-clause, QEALM is closed under Robinson’s resolution, and retains hardness when intersected with Krom, Horn and Affine (unlike QBF [5, 21]).

Part of our motivation is to utilise the PSPACE-completeness and connection to QBF for benefits in solving, certification and theory. At present our recursive solving algorithms show PSPACE inclusion, but lacks optimisations we would need to use it practically.

One area of future interest that can also take advantage of is QBF proof complexity, as our understanding of it has grown substantially over the last decade. For example, some QBF proof systems are known to have super-polynomial lower bounds only if either proof complexity or circuit complexity open problems are solved [3]. In theory, a proof system with this property exists in every PSPACE language through a QBF translation, but finding one that is natural and suitable for applications is more difficult. Unfortunately, Robinson’s resolution is known to have exponential lower bounds even for the QEALM-fragment. In fact, it runs foul of something similar to a well-known QBF proof complexity pitfall known as “semantic hardness” [2], which we aim to demonstrate in future work.

⁷Choosing `--kbo_weight_scheme precedence` and a size-preferring `--selection 2` (see [9]).

Acknowledgements

This work was supported by the European Union under the project ROBOPROX (reg. no. CZ.02.01.01/00/22_008/0004590) and by the Czech Science Foundation project 24-12759S.

References

- [1] Filip Bártek, Ahmed Bhayat, Robin Coutelier, Márton Hajdú, Matthias Hetzenberger, Petra Hozová, Laura Kovács, Jakob Rath, Michael Rawson, Giles Reger, Martin Suda, Johannes Schoisswohl, and Andrei Voronkov. The vampire diary. In Ruzica Piskac and Zvonimir Rakamaric, editors, *Computer Aided Verification - 37th International Conference, CAV 2025, Zagreb, Croatia, July 23-25, 2025, Proceedings, Part III*, volume 15933 of *Lecture Notes in Computer Science*, pages 57–71. Springer, 2025.
- [2] Olaf Beyersdorff, Joshua Blinkhorn, and Luke Hinde. Size, cost, and capacity: A semantic technique for hard random QBFs. *Logical Methods in Computer Science*, Volume 15, Issue 1, 02 2019.
- [3] Olaf Beyersdorff, Ilario Bonacina, Leroy Chew, and Jan Pich. Frege systems for quantified Boolean logic. *J. ACM*, 67(2), apr 2020.
- [4] Olaf Beyersdorff, Leroy Chew, and Mikoláš Janota. New resolution-based QBF calculi and their proof complexity. *ACM Trans. Comput. Theory*, 11(4):26:1–26:42, 2019.
- [5] Hans Kleine Büning, Marek Karpinski, and Andreas Flögel. Resolution for quantified Boolean formulas. *Inf. Comput.*, 117(1):12–18, feb 1995.
- [6] Gerhard Buntrock, Carsten Damm, Ulrich Hertrampf, and Christoph Meinel. Structure and importance of logspace-mod class. *Math. Systems Theory*, 25:223–237, 1992.
- [7] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing, STOC '71*, page 151–158, New York, NY, USA, 1971. Association for Computing Machinery.
- [8] L. Henkin. Some remarks on infinitely long formulas. In *Infinitistic Methods, Proceedings of Symposium on Foundations of Mathematics*, pages 167–183, 1961.
- [9] Krystof Hoder, Giles Reger, Martin Suda, and Andrei Voronkov. Selecting the selection. In Nicola Olivetti and Ashish Tiwari, editors, *Automated Reasoning - 8th International Joint Conference, IJCAR 2016, Coimbra, Portugal, June 27 - July 2, 2016, Proceedings*, volume 9706 of *Lecture Notes in Computer Science*, pages 313–329. Springer, 2016.
- [10] Ullrich Hustadt and Renate A. Schmidt. On evaluating decision procedures for modal logic. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI 97, Nagoya, Japan, August 23-29, 1997, 2 Volumes*, pages 202–209. Morgan Kaufmann, 1997.
- [11] Ullrich Hustadt and Renate A. Schmidt. MSPASS: modal reasoning by translation and first-order resolution. In Roy Dyckhoff, editor, *Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEUX 2000, St Andrews, Scotland, UK, July 3-7, 2000, Proceedings*, volume 1847 of *Lecture Notes in Computer Science*, pages 67–71. Springer, 2000.
- [12] Mikoláš Janota and Joao Marques-Silva. Expansion-based QBF solving versus Q-resolution. *Theor. Comput. Sci.*, 577:25–42, 2015.
- [13] Neil D. Jones and William T. Laaser. Complete problems for deterministic polynomial time. *Theoretical Computer Science*, 3(1):105–117, 1976.
- [14] Neil D. Jones, Y.E. Lien, and William T. Laaser. New problems complete for nondeterministic log space. *Mathematical Systems Theory*, 10:1–17, 1976.
- [15] Harry R. Lewis. Complexity results for classes of quantificational formulas. *Journal of Computer and System Sciences*, 21(3):317–353, 1980.
- [16] Hans Jürgen Ohlbach and Renate A. Schmidt. Functional translation and second-order frame properties of modal logics. *J. Log. Comput.*, 7(5):581–603, 1997.

- [17] Juan Antonio Navarro Pérez and Andrei Voronkov. Planning with effectively propositional logic. In Andrei Voronkov and Christoph Weidenbach, editors, *Programming Logics - Essays in Memory of Harald Ganzinger*, volume 7797 of *Lecture Notes in Computer Science*, pages 302–316. Springer, 2013.
- [18] David A. Plaisted. Complete problems in the first-order predicate calculus. *J. Comput. Syst. Sci.*, 29(1):8–35, aug 1984.
- [19] Willard V Quine. On the limits of decision. *Akten des XIV. Internationalen Kongresses für Philosophie*, 3:57–62, 1969.
- [20] F. P. Ramsey. On a problem of formal logic. *Proceedings of the London Mathematical Society*, s2-30(1):264–286, 1930.
- [21] Anja Remshagen. The complexity of constrained quantified formulas. In *Proceedings of the IADIS international conference intelligent systems and agents 2010: Part of the IADIS multi conference on computer science and information systems*, 2010.
- [22] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970.
- [23] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC ’78, page 216–226, New York, NY, USA, 1978. Association for Computing Machinery.
- [24] Renate A. Schmidt. Decidability by resolution for propositional modal logics. *J. Autom. Reason.*, 22(4):379–396, 1999.
- [25] Martina Seidl, Florian Lonsing, and Armin Biere. qbf2epr: A tool for generating EPR formulas from QBF. In Pascal Fontaine, Renate A. Schmidt, and Stephan Schulz, editors, *Third Workshop on Practical Aspects of Automated Reasoning, PAAR-2012, Manchester, UK, June 30 - July 1, 2012*, volume 21 of *EPiC Series in Computing*, pages 139–148. EasyChair, 2012.
- [26] Raymond M. Smullyan. *First-Order Logic*. Springer Berlin Heidelberg, 1968.
- [27] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.
- [28] G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0. *Journal of Automated Reasoning*, 59(4):483–502, 2017.
- [29] Andrei Voronkov. AVATAR: the architecture for first-order theorem provers. In Armin Biere and Roderick Bloem, editors, *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, volume 8559 of *Lecture Notes in Computer Science*, pages 696–710. Springer, 2014.