Reformulating Queries: Theory and Practice (Full version)

Anonymous

Abstract

We consider a setting where a user wants to pose a query against a dataset where some background knowledge, expressed as logical sentences, is available, but only a subset of the information can be used to answer the query. We thus want to *reformulate* the user query against the subvocabulary, arriving at a query that is equivalent to the user's query assuming the background theory, but using only the restricted vocabulary. We consider two variations of the problem, one where we want any such reformulation and another where we restrict the size. We first present a classification of the complexity of the problem, showing that distinct techniques are needed for the two variations above. We then present algorithms for solving the problems in practice and evaluate their performance.

1 Introduction

A basic problem dealt with in databases and knowledge representation is *query reformulation with background knowledge*. The input to the problem includes a logical formula Q, the *input query*, and a set of sentences of first-order logic Σ, the *integrity constraints* or *background theory*. We also have a specification of an "interface restriction": what data can we actually access? For example, we can specify the allowed data as a sub-vocabulary $\mathcal{V}$ of the vocabulary of Σ and Q. We want to know if the answer to Q can be obtained using the restricted interface. It is well-known that this is the case if and only if there is another formula Q' that is logical equivalent Q over all instances satisfying Σ (for short, Σ-equivalent to Q), and which uses only relations from $\mathcal{V}$. If Q' is in first-order logic, we call this an $\mathsf{FO}, \mathcal{V}, \Sigma$-*reformulation of* Q. We can likewise restrict Q' to be in a subset of first-order logic, e.g., by restricting the connectives or quantifiers.

Variants of the reformulation problem have been considered in several communities:

• Within databases the emphasis has been on theories given as "dependencies" — Horn sentences in predicate (first-order) logic. The bulk of the work has been on "view-based reformulation": the vocabulary is divided up into the "base relations", and the "view relations" which are derived from the base ones. Each view relation $V_i \in \mathcal{V}$ is associated with a defining formula φ_i, and the sentences Σ state that V_i corresponds exactly to its definition: $V_i(\vec{x}) \leftrightarrow \varphi_i(\vec{x})$. The interface relations are then the $V_i \in \mathcal{V}$. The problem has been extensively studied in the case where the "view definitions" φ_i are built up from $\wedge, \exists$ (that is, they are *conjunctive queries*, or *CQs*). One desires a target Q' as another CQ, perhaps adding a requirement that Q' contains no non-redundant atoms. The view-based reformulation problem has been extended to reformulation with dependencies, using the *chase and backchase (C&B) method* [Deutsch *et al.*, 2006; Popa, 2000]. The method proceeds by first generating consequences under the dependences (the *chasing phase*) and then searching for a minimal subset of the consequences that are equivalent to the original formula Q (the *backchasing phase*). The C&B exploits properties specific to dependencies: the full set of consequences can often be generated compactly; further the consequences can be instrumented in such a way as to make the backchasing phase efficient [Meier, 2014; Ileana *et al.*, 2014]. Reformulation over both subsignatures and functional interfaces is considered in [Toman and Weddell, 2011; Benedikt *et al.*, 2016]. [Toman and Weddell, 2011] describes implementations of reformulation via interpolation and via the C&B method, but the implementation discussion focuses (as with [Meier, 2014; Ileana *et al.*, 2014]) on the case without disjunction. In contrast, [Benedikt *et al.*, 2016] deals strictly with theoretical issues, providing complexity bounds on the existence of a reformulation for richer languages such as the guarded fragment, but not for intermediate languages, and not for reformulation with a size bound.

• In the knowledge representation community both the background knowledge and the problem statement have been different. For background theories, the focus has been on *description logics*. The signature is usually restricted to have only unary and binary predicates; but the sentences may allow more flexibility than the Horn fragments considered in the DB community, including both disjunction and negation. The emphasis has not been on reformulation as defined above, where we desire a target formula that gives exactly the same results as the input on an instance satisfying the background theory. Instead the emphasis is on a more general problem, getting the best upper approximation of a formula that uses a given vocabulary. Variants of this problem have been studied as a *forgetting* or *uniform interpolation* problem [Lang *et al.*,

2003; Lutz and Wolter, 2011; Koopmann and Nikitina, 2017; Konev *et al.*, 2009; Lutz *et al.*, 2012; Nikitina and Rudolph, 2014]. These works are interested in approximating theories by other theories (e.g., in a restricted signature), while staying within a particular decidable fragment. This is motivated by issues of modularity in ontologies [Kontchakov *et al.*, 2009; Jiménez-Ruiz *et al.*, 2008].

In this paper our input will be a theory representing background knowledge and a simple formula (e.g., CQ or Boolean combination of CQs). We look for a formula that is *exactly* equivalent relative to the background theory, as in the DB work above. Thus the tools will be quite different from works considering the "best approximation" of a theory as in [Lutz and Wolter, 2011; Koopmann and Nikitina, 2017; Lutz and Wolter, 2011; Konev *et al.*, 2009; Lutz *et al.*, 2012; Nikitina and Rudolph, 2014]. However, unlike work in the DB community, we consider theories that allow disjunction and negation. We distinguish the problem of determining if *some* reformulation exists from the problem of finding one that is within a certain *size bound*; the latter problem has not been studied even in the case of dependencies. Both problems can be attacked by a reduction to entailment. For finding some reformulation there is a reduction using interpolation in theorem proving, dating back to the birth of interpolation [Craig, 1957b]. For size-bounded reformulation one can use a "guess-and-check" method that is analogous to the C&B, but accommodating disjunction and negation in the theories. Surprisingly, we give lower bounds showing that both these approaches provide optimal complexity for their respective problems, showing this over a range of logics and targets for reformulation. After completing an examination of the complexity of reformulation in Section 3, we turn to practice in Section 4. We discuss several ways of implementing the approaches of Section 3, focusing on an approach using a new variant of Huang's interpolation algorithm for resolution [Huang, 1995]. We perform an extensive experimental comparison of the "Optimized Huang" approach on top of an existing theorem prover with several alternatives, including both an extension of C&B to disjunction (based on [Deutsch *et al.*, 2008]) and an approach using an interpolation algorithm that is integrated with a theorem-prover. *To the best of our knowledge our work is the first look at query reformulation over a subsignature for logics that include disjunction in arbitrary arity, either from the point of view of complexity or in practice.*

2 Preliminaries

The most basic problem we consider is the *reformulation existence problem*, denoted $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathcal{L})$, where:

- Q is a first-order sentence (the input query),
- Σ a finite set of first-order sentences (the background theory),
- $\mathcal{V}$ a subset of the relations mentioned in Σ,
- $\mathcal{L}$ is a fragment of first-order logic (e.g., conjunctive queries).

The problem holds if and only if there is a query Q' in $\mathcal{L}$ that is a Σ-reformulation of Q over $\mathcal{V}$. That is, we wish to determine if any reformulation in $\mathcal{L}$ using vocabulary $\mathcal{V}$ exists.

The *size-restricted reformulation problem* is denoted $\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathcal{L}, k)$. It has an additional integer argument k, written in unary. It returns true whenever there is a Σ-reformulation of Q over $\mathcal{V}$ of size at most k.

We first consider these problems when the vocabulary is in the setting of propositional logic. There we consider input queries Q consisting of a single proposition, Σ consisting of propositional sentences, and $\mathcal{V}$ being a subset of the propositions mentioned in Σ. We can consider getting reformulations Q' that are propositional sentences (PROP, $\mathcal{V}$, Σ-reformulations), ones that do not use negation (PROP^{+}, $\mathcal{V}$, Σ-reformulations), sentences in disjunctive normal form (DNF, $\mathcal{V}$, Σ-reformulations), DNFs that are also monotone (that is, disjunctions of conjunctions of positive literals, $\mathsf{PROP}^{\mathsf{UCQ}}$, $\mathcal{V}$, Σ-reformulations), and sentences that use only conjunction ($\mathsf{PROP}^{\mathsf{CQ}}$, $\mathcal{V}$, Σ-reformulations). For our theories Σ we will consider general propositional formulas, Horn formulas (clauses in which there is at most one positive literal), as well as "disjunctive Horn" formulas, that is, either implications of the form $\bigwedge_i A_i \rightarrow \bigvee_j B_j$ with A_i, B_j propositions or disjunctions $\bigvee_j B_j$ where B_j propositions.

We also consider the first order case, where we will always consider input queries that are CQs. Our target reformulation Q' can be either a CQ, a positive existential FO (FO^{+}) formula, a union of CQs (UCQ), or a general first-order sentence: thus we talk about $\mathsf{CQ}, \mathcal{V}, \Sigma$-reformulations, $\mathsf{UCQ}, \mathcal{V}, \Sigma$-reformulations, etc. Note that for $\exists\mathsf{REF}$, there is no difference between PROP^{+} and $\mathsf{PROP}^{\mathsf{UCQ}}$ targets, since these have the same expressiveness. We thus use "monotone reformulation" for both. Similarly, in the first-order case, "monotone reformulation" refers to either FO^{+} or UCQ.

For the first-order case, we begin with *existential rules* or *Tuple-Generating Dependencies* (TGDs), universal quantifications of formulas of the form $\bigwedge_i R_i(\vec{x}) \rightarrow \exists \vec{y}\ S(\vec{x}, \vec{y})$. Most reasoning problems concerning general TGDs are undecidable; but they become decidable when the TGDs are *full*: that is, there are no existentials on the right side. They are also decidable when rules with existentials in the head can not fire unboundedly often; this holds for the weakly-acyclic TGDs [Fagin *et al.*, 2005].

In this paper we also investigate the reformulation problem for theories consisting of *disjunctive TGDs*, sentences of the form: $\bigwedge_i R_i(\vec{x}) \rightarrow \exists \vec{y}\ \bigvee_i S_i(\vec{x}, \vec{y})$. These sentences arise naturally in data integration, both to model bi-directional relationships between a local and global schema and to capture data type restrictions. For brevity we focus on the case of *full disjunctive TGDs* ($\mathsf{FullDisTGDs}$), where there are no existentially quantified variables in the head.

When φ_1 and φ_2 are logical sentences we write $\varphi_1 \models \varphi_2$ (φ_1 *entails* φ_2) if any model of φ_1 is a model of φ_2.

3 The complexity of reformulation

We now investigate our reformulation problems in theory. All of our upper bounds follow from reduction to entailment of CQs for the logical fragments in question. The lower bounds will require more effort.

Complexity of $\exists\mathsf{REF}$. The fact that the $\exists\mathsf{REF}$ problem for both general reformulation and monotone reformulation

Table 1: Summary of results for $\exists$REF

Constraints	mon. or gen.	CQ
PROP	CoNP	CoNP
$\vee$Horn	CoNP	CoNP
PROP$^{\text{HORN}}$	P	P
FullTGD	ExpTime, NP	ExpTime, NP
WATGD	2ExpTime	2ExpTime
FullDisTGD	CoNExpTime, Π_2^p	Undecidable

Table 2: Summary of results for REF$^{\leq}$

Constraints	monotone	CQ
PROP	Σ_2^p	Σ_2^p
$\vee$Horn	Σ_2^p	Σ_2^p
PROP$^{\text{HORN}}$	NP	NP
FullTGD	ExpTime, NP	ExpTime, NP
WATGD	2ExpTime	2ExpTime
FullDisTGD	CoNExpTime, Σ_3^p	CoNExpTime, Σ_3^p

can be reduced to entailments from interpolation, dates back to work of Craig [Craig, 1957b]. Craig's reduction takes as input a set of relations Sch, creating a copy $\mathsf{Sch}' = \{R' \mid R \in \mathsf{Sch}\}$. For any formula φ over Sch, let φ' be formed by replacing any relation R by R'. Given a set of relations $\mathcal{V}$ let $\mathsf{ForwAx} = \bigwedge_{R\in\mathcal{V}} \forall \vec{x}\ R(\vec{x}) \rightarrow R'(\vec{x})$ and $\mathsf{BackAx} = \bigwedge_{R\in\mathcal{V}} \forall \vec{x}\ R'(\vec{x}) \rightarrow R(\vec{x})$.

From the work of Craig [Craig, 1957b] and Lyndon [Lyndon, 1959] we obtain:

Proposition 1. $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{FO})$ *holds iff* $Q \wedge \Sigma \wedge \mathsf{ForwAx} \wedge \mathsf{BackAx} \wedge \Sigma' \wedge \neg Q'$ *is not satisfiable.* $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{UCQ})$ *holds iff* $Q \wedge \Sigma \wedge \mathsf{ForwAx} \wedge \Sigma' \wedge \neg Q'$ *is not satisfiable.*

They also show that reformulation can be read off from a proof of unsatisfiability in a suitable proof system, using an interpolation algorithm. We discuss this further in Section 4. For now we note that this result immediately provides upper bounds when Σ is restricted to range over fragments with decidable entailment. For example, applying this to propositional logic we get upper bounds on determining if there is a propositional reformulation and monotone reformulation:

Corollary 1. $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{PROP})$ *and* $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{PROP}^+)$ *are in* CoNP *as* Q *ranges over conjunctions of propositions,* Σ *over propositional sentences.*

The CoNP upper bound is inherited if Σ consists of "disjunctive Horn clauses" $\vee$Horn, that is, formulas of the form $\bigwedge A_i \rightarrow \bigvee_j B_j$. Alternatively, these are clauses with at least one positive literal. In the case where the background theory Σ is in propositional Horn logic, entailment is in P and thus $\exists$REF for PROP and PROP$^+$ is in P as well.

The same approach can be applied to predicate (first-order) logic. For full TGDs, entailment is in ExpTime, and becomes NP if we fix the arity of the schema. Thus applying the reduction above, $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{FO})$ and $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{UCQ})$ are both in ExpTime and in NP for fixed arity.

If we consider weakly acyclic TGDs, entailment is 2ExpTime-complete [Calì *et al.*, 2010]. Using this we see that $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{FO})$ and $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{UCQ})$ are in 2ExpTime. For full disjunctive TGDs, we can easily show that entailment is CoNExpTime-complete, and becomes Π_2^p-complete when the arity of relations is fixed; note that although the syntax of disjunctive TGDs matches Disjunctive Datalog, the semantics is classical entailment, different from [Eiter *et al.*, 1997]. Thus the "Craig reduction" above shows $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{FO})$ and $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{UCQ})$ are in CoNExpTime and in Π_2^p for fixed arity.

The Craig reduction does not provide any information about reformulating when the theories are propositional but the target does not allow disjunction. But there is a simpler reduction to entailment for $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{PROP}^{\mathsf{CQ}})$: letting S be the propositions in $\mathcal{V}$ that are entailed by $Q \wedge \Sigma$, this holds exactly when $S \wedge \Sigma$ entails Q. We can guess counterexample models for implication of each proposition not in S and also a counterexample to $S \wedge \Sigma$ entailing Q to get a CoNP bound.

In the case of theories Σ that are Horn (in particular, those given by TGDs) it is known that if a CQ Q has a UCQ reformulation over $\mathcal{V}$ with respect to Σ then it has a CQ reformulation as well. Indeed, one of the disjuncts in the UCQ reformulation must already give a CQ reformulation. Thus we get upper bounds for CQ-reformulation for classes of TGDs. For disjunctive TGDs, disjunction may be essential in the reformulation; thus for CQ-reformulation of disjunctive TGDs, we can not make use of either the Craig reduction or a reduction that considers each atom at a time (as for PROP$^{\mathsf{CQ}}$). We can show that $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{CQ})$ is undecidable when Σ varies over full disjunctive TGDs this problem is undecidable. The proof is by reduction from Datalog containment. Intuitively, solving $\exists$REF requires us to be able to decide whether a UCQ Q is equivalent to a CQ Q' with respect to a set of full TGDs Σ, and this is the same as checking that the Datalog programs $\Sigma \cup \{Q \rightarrow \mathsf{Goal}\}$ and $\Sigma \cup \{Q' \rightarrow \mathsf{Goal}\}$ are equivalent.

Table 1 summarizes our bounds for $\exists$REF. The second column shows the bounds one gets for reformulating as a general formula (propositional or first-order, depending on the theories Σ), as a monotone formula, or as a union of conjunctions: in each case the bounds are the same. The last column shows the complexity of reformulating as conjunction of atoms. In the case of FullTGD and FullDisTGD, we give the complexity both for the general case and when the arity of the schema is fixed. One can easily show that all of the complexity bounds are tight, by a reduction from entailment to reformulation. Full proofs are in the appendix.

Complexity of REF$^{\leq}$**.** The problem REF$^{\leq}$ does not have a similar direct reduction to entailment. But one can reduce to "guessing plus entailment". The machine guesses Q' and then checks entailment using the oracle. For example, from this "guessing and verifying" technique we get an upper

bound on $\mathsf{REF}^{\leq}$ for propositional logic.

Proposition 2. *The problems* $\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathsf{PROP}, k)$, $\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathsf{PROP}^{+}, k)$, *and* $\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathsf{PROP}^{\mathsf{CQ}}, k)$ *are in* Σ_2^p, *as* Σ *varies over formulas in propositional logic. For* Σ *varying over* $\mathsf{PROP}^{\mathsf{HORN}}$, *these are all in* NP.

For full TGDs, the approach above gives a bound of ExpTime for all of the variants of $\mathsf{REF}^{\leq}$, since the verification stage is Datalog evaluation, done in ExpTime, and this subsumes the time taken for the guessing phase. The problem is in NP when the arity is fixed. For weakly-acyclic TGDs the approach gives 2ExpTime even for fixed arity. For FullDisTGD it gives coNExpTime (since the guessing phase is again neglible on top of entailment) and Σ_3^p for fixed arity, since the guessing stage adds an alternation on top of the Π_2^p entailment problem.

The complexity bounds for $\mathsf{REF}^{\leq}$ are shown in Table 2. In the second column we list the complexity for finding a small reformulation as a monotone formula or disjunction of conjunctions; the complexity is the same in all cases. The last column shows that finding a small reformulation as a conjunction of atoms always has the same complexity. As with $\exists$REF, for FullTGD and FullDisTGD we consider both the general case and fixed arity.

All of the lower bounds provided by the guess-and-check method turn out to be tight, but unlike for $\exists$REF, this is not obvious. For $\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathsf{PROP}, k)$ where Σ ranges over arbitrary propositional formulas, the Σ_2^p lower bound follows from results on minimization of Boolean formulas in [Umans, 2001]. Indeed, using related results [Buchfuhrer and Umans, 2011] one can show hardness for the variant of the problem where we fix the number of alternations of $\vee$ and $\wedge$. The technique of [Buchfuhrer and Umans, 2011] requires formulas with hard satisfiability problems, while $\vee$Horn formulas are always satisfiable. Thus for $\vee$Horn we need a more-involved "native" reduction for Σ_2^p-hardness:

Theorem 1. $\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathcal{L}, k)$ *for* $\mathcal{L} \in \{\mathsf{PROP}^{+}, \mathsf{PROP}^{\mathsf{UCQ}}, \mathsf{PROP}^{\mathsf{CQ}}\}$ *is* Σ_2^p*-hard when* Σ *ranges over* $\vee$Horn *formulas.*

Sketch. The proof is by reduction of $\exists\forall$3SAT problem. Suppose $\varphi = \exists\bar{x}\forall\bar{y}\neg\psi$ is formula with $\bar{x}$ and $\bar{y}$ tuples of propositional variables, and ψ a conjunction of clauses $\ell^1 \vee \ell^2 \vee \ell^3$ with each ℓ^j a variable from $\bar{x} \cup \bar{y}$ or its negation.

We construct a set of propositional disjunctive Horn formulas Σ and a set of propositional variables $\mathcal{V}$ such that φ is valid if and only if there exists a positive formula Q' over $\mathcal{V}$ of size k such that Q' is equivalent to a propositional variable Q over Σ. The vocabulary $\mathcal{V}$ consists of variables TrueX_i and FalseX_i for each x_i in $\bar{x}$, while Σ guarantees the following:

1. Q implies all the variables in $\mathcal{V}$, so the equivalence of Q and Q' over Σ boils down to the containment of Q' in Q;
2. containment of Q' in Q over Σ is only possible if Q' implies either TrueX_i or FalseX_i for each x_i; since the size of Q' is bounded by k, it means that Q' can only be of the form $\mathsf{AsX}_1 \wedge \cdots \wedge \mathsf{AsX}_k$, for $\mathsf{AsX}_i \in \{\mathsf{TrueX}_i, \mathsf{FalseX}_i\}$—that is, Q' corresponds to an assignment of variables x_i;
3. if a dataset and Σ imply TrueX_i or FalseX_i for each x_i, then they imply either TrueY_i or FalseY_i for each universally quantified y_i in $\bar{y}$, which corresponds to an assignment of the universally quantified variables in φ;
4. if one of the minimal extensions of a dataset satisfying the rules ("disjunctive chase models") *falsifies* a clause in ψ under this encoding, which corresponds to falsifying the whole ψ, then this extension implies Q.

These guarantees imply the statement of the theorem. □

For arbitrary FullTGD, WATGD, and FullDisTGD sentences, our upper bounds match those for entailment, and there is an easy reduction from entailment that gives a matching lower bound. But when the arity is fixed, entailment for FullDisTGD is Π_2^p, while our reduction to entailment gives only a Σ_3^p upper bound for $\mathsf{REF}^{\leq}$. We show Σ_3^p-hardness of $\mathsf{REF}^{\leq}$ for bounded arity FullDisTGD via a direct reduction from the $\exists\forall\exists$3SAT problem. The reduction uses a similar idea to the Σ_2^p-reduction in the propositional case, but it is more involved: inner existential quantification is encoded by a homomorphism check from a candidate reformulation Q' to Q. Full details are in the appendix.

Theorem 2. $\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathcal{L}, k)$ *for any* $\mathcal{L} \in \{\mathsf{FO}^{+}, \mathsf{UCQ}, \mathsf{CQ}\}$ *is* Σ_3^p*-hard when* Σ *ranges over sets of* FullDisTGD *formulas and the arity of relations is bounded by* 2.

4 Implementing reformulation

We consider how to implement methods for finding a reformulation. The results of Section 3 show that for $\exists$REF there is a polynomial reduction to entailment, giving the possibility of performing reformulation on top of a theorem-prover. In contrast, our lower bounds for $\mathsf{REF}^{\leq}$ imply that in the presence of disjunction such a reduction is unlikely. We thus return to the method of Craig outlined in Section 3. A *Craig interpolant* for an entailment $\varphi_1 \models \varphi_2$ is a formula ψ such that $\varphi_1 \models \psi \models \varphi_2$ and ψ contains only relations common to φ_1 and φ_2. A *Lyndon interpolant* further has the property that a relation appears positively in ψ only if it appears positively in both φ_1 and φ_2, and similarly for appearing negatively. [Craig, 1957a] showed that first-order Craig interpolants exist for every first-order entailment, while [Lyndon, 1959] showed the same for Lyndon interpolants. The argument of [Craig, 1957b] shows that any Craig interpolant for the entailment

$$Q \wedge \Sigma \wedge \mathsf{BackAx} \models (\mathsf{ForwAx} \wedge \Sigma' \rightarrow Q')$$

where ForwAx, Σ', Q' are as in Section 3, can be converted to a $\Sigma, \mathcal{V}$-reformulation of Q; conversely the entailment is also a necessary condition for such a reformulation to exist. A variation of the argument shows that any Lyndon interpolant for

$$Q \wedge \Sigma \models (\mathsf{ForwAx} \wedge \Sigma') \rightarrow Q'$$

is itself a monotone $\Sigma, \mathcal{V}$-reformulation of Q, and the entailment is likewise necessary. Note that for reformulating a conjunction of literals over propositional theory, these two notions co-incide. Thus implementing $\exists$REF using Craig's

Figure 1: Evaluation of reformulation approaches.

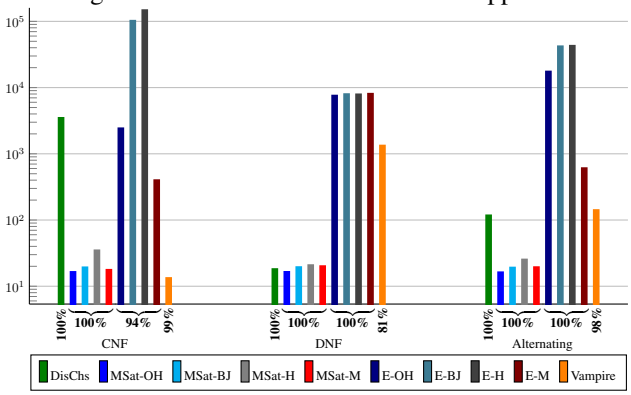
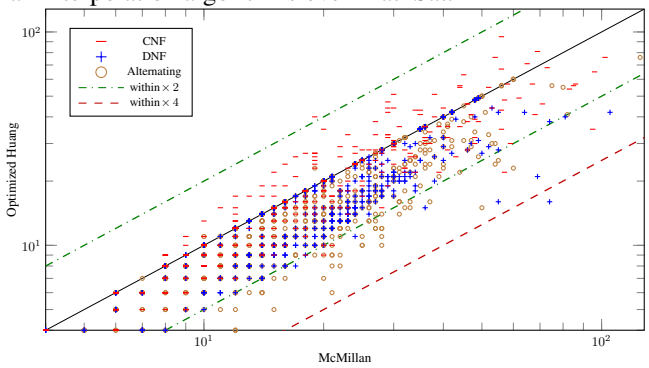


Figure 2: Comparison between Optimized Huang and McMillan interpolation algorithms over MathSat.



technique requires finding proofs witnessing the appropriate entailment, and then applying an interpolation procedure. We implemented monotone $\Sigma, \mathcal{V}$-reformulation for target languages with disjunction following this high-level approach, proving an entailment and applying an interpolation procedure to it. Since the goal of this work is not to build a theorem-prover from scratch, we considered both using a theorem-prover that already supports interpolation, and building our own interpolation procedure on top of existing theorem-provers. The first-order theorem prover which has the most robust native support for interpolation is Vampire [Hoder *et al.*, 2010; 2012]. Thus we test Vampire's interpolation as a "black-box" to get reformulations, using the reductions to interpolation used for all of our ∃REF problem. We complemented this with our own implementation of interpolation for resolution theorem-proving, discussed next.

A number of interpolation procedures exist for resolution, each of them creating a "partial interpolant" proceeding top-down on a resolution proof. The final interpolant is the formula associated with a leaf node representing the final contradiction in the resolution proof. Huang [Huang, 1995] and Bonacina and Johannson [Bonacina and Johansson, 2011] provide similar algorithms; the base cases assign either True or False to input clauses, while the inductive step for resolution may introduce atoms. McMillan's algorithm [McMillan, 2003], in contrast, has a more complex base case, while the propagation through resolution steps is purely compositional. For problems where we want a monotone reformulation, we need an implementation of "Lyndon interpolation" – one that does not introduce relations negated unless they appear negatively on both sides of the entailment. Huang's algorithm does not have this property, but we developed its variant that achieves this while also performing certain optimizations that we found crucial in producing good quality reformulations.

We describe the algorithm now for universal theories (the extension to handle general theories is via skolemization, as described in [Huang, 1995]). Consider an entailment $\Sigma_1 \models \Sigma_2$, where Σ_1 and Σ_2 are clauses. In a resolution proof, we search for a contradiction from $\Sigma_1 \cup \neg\Sigma_2$ where we must convert $\neg\Sigma_2$ to a conjunction of clauses. A resolution proof is a DAG, with each node representing a derived formula within the proof. Root nodes are either elements of Σ_1 or $\neg\Sigma_2$, while non-root node with formula φ has two parents representing formulas that resolve to φ. In a complete proof there is at least one leaf node which is associated with a contradiction.

For purposes of interpolation, clauses in each node are first annotated with a *provenance*. Root nodes are annotated LEFT if they are in Σ_1 and RIGHT if they are in $\neg\Sigma_2$. The result of a resolution step is a clause in which each literal results applying a unifier to a literal in one of the parents, and inherits the provenance of that literal.

Following [Huang, 1995], we assign partial interpolants for root nodes as: False (resp. True), if the root corresponds to a clause in Σ_1 (resp. Σ_2). For a non-root node c with parents p_1 and p_2, we form the interpolant by induction. Let φ_1 (resp. φ_2) be the interpolant for p_1 (resp. p_2). Let the atoms involved in resolution be $\alpha(\rho_1 \dots \rho_n)$ in p_1 an $\neg\alpha(\tau_1 \dots \tau_n)$ in p_2, and let σ be the unifier. We do case analysis on the provenance of $\alpha(\rho_1 \dots \rho_n)$ in p_1 and $\neg\alpha(\tau_1 \dots \tau_n)$ in p_2.

- If the provenance is LEFT (resp. RIGHT) in both, we produce $\sigma(\varphi_1) \vee \sigma(\varphi_1)$ (resp. $\sigma(\varphi_1) \wedge \sigma(\varphi_2)$).
- If the provenance for $\alpha(\rho_1 \dots \rho_n)$ in p_1 is LEFT and the provenance for $\neg\alpha(\tau_1 \dots \tau_n)$ in p_2 is RIGHT, we produce $\sigma(\varphi_1 \vee (\alpha(\rho_1, \dots \rho_n) \wedge \varphi_2))$.
- If the provenance for $\alpha(\rho_1 \dots \rho_n)$ in p_1 is RIGHT and the provenance for $\neg\alpha(\tau_1 \dots \tau_n)$ in p_2 is LEFT, we produce $\sigma(\varphi_2 \vee (\neg A \wedge \varphi_1))$.

The distinction from Huang's original algorithm is that [Huang, 1995] collapses the last two cases into one, producing a unifier of $(\neg\alpha \wedge \varphi_1) \vee (\alpha \wedge \varphi_2)$ in both cases. Indeed, it is for this reason that Huang lacks the Lyndon property. Our modification is inspired by a similar case distinction in [Bonacina and Johansson, 2011]. In the above algorithm, we assume that at each inductive step the propositional factoring and absorption rules are applied, so that False $\vee \varphi$ is simplified to φ, False $\wedge \varphi$ to False, and so forth. Further, in the case where α is propositional, we simplify the second-to-last case by producing $\sigma(\varphi_1 \vee (\alpha \wedge \varphi_2'))$, where φ_2' substitutes True for occurrences of α and then simplies as above; similarly for the last case, substituting for False in φ_1.

In the above algorithm, we assume that at each inductive step the propositional factoring and absorption rules are applied, so that False $\vee \varphi$ is simplified to φ, False $\wedge \varphi$ to False, and so forth. Further, in the case where α is propositional, we simplify the second-to-last case by producing $\sigma(\varphi_1 \vee (\alpha \wedge \varphi_2'))$, where φ_2' substitutes True for occurrences of α and then simplies as above; similarly for the last case, substituting for False in φ_1. We implement a number of other simplifications, described in [Anonymous, 2017]. We refer to this as the *Optimized Huang Algorithm* below.

Proposition 3. *The "Optimized Huang Algorithm" produces an interpolant φ for $\Sigma_1 \models \Sigma_2$, and any negated literal in φ occurs negated in Σ_1 and Σ_2.*

All of these interpolation procedures require resolution proofs as inputs. For propositional logic there are several options, but we focused on MathSat [Cimatti *et al.*, 2013] to generate proofs; for first-order logic the only theorem-prover that both produces completely detailed resolution proofs and supports quantification was E [Schulz, 2013]. We also look at a variant of the chase-based techniques used for TGDs (e.g., [Deutsch *et al.*, 2006]): we take a "disjunctive chase" of the input query Q under the background theory – firing any deterministic rules and then forking into two models on each disjunction. This produces a set S of models such that another query Q' is a consequence of Q if it holds in each member of S. We can thus search for a reformulation by looking for CQs of $\mathcal{V}$ formulas and testing whether they hold in each S. We implemented this technique using the disjunctive logic programming system DLV [Leone *et al.*, 2006] to create the set S, focusing on the propositional case.

Testbed. We built a generator for reformulation problems, based on two components. The first is a generator of random *target reformulations*: formulas in the desired class (e.g., $\mathsf{PROP}, \mathsf{PROP}^+$). The second is a generator that takes as input a target reformulation φ and outputs sentences Σ, input query Q, and signature V such that φ is a Σ-reformulation of Q over $\mathcal{V}$. The tool generates sentences that follow the structure of φ to witness that Q implies φ; e.g., if φ is $B \wedge (A_1 \vee A_2)$, Σ will contain clauses equivalent to $Q \rightarrow C$, $C \rightarrow B$, $C \rightarrow D$, $D \rightarrow A_1 \vee A_2$, and additionally some "noise clauses" (that can be controlled by an input parameter). Σ will likewise contain sentences that witness φ implies Q, again with a tunable noise parameter.

We tested the following algorithms. 1. Native end-to-end interpolation using Vampire 4.1 for CASC J8. 2. Our implementations of the interpolation algorithms of [Huang, 1995], [Bonacina and Johansson, 2011], and [McMillan, 2003], and our implementation of Optimized Huang, on top of both the MathSat and E theorem provers. 3. As an alternative to interpolation for propositional theories, we implemented a variant of the C&B for logics with disjunction, based on the "disjunctive chase" of [Deutsch *et al.*, 2008], making use of the Disjunctive Datalog engine DLV. Specifically, we generate all minimal models of $Q \wedge \Sigma$ using DLV. We then let φ be the disjunction over all models M of the conjunction of the literals in $\mathcal{V}$ of M, and check if φ implies Q under Σ. It is straightforward to see that for propositional logic, if there is any reformulation over $\mathcal{V}$, this approach will find one. We report experiments only for the case of propositional formulas. Since the interpolation algorithms themselves are linear, the runtime performance of interpolation-based approaches is dominated by theorem-proving time. MathSat was the fastest, requiring only a few milliseconds. The disjunctive chase approach was next, always terminating within 86secs. We timed out any other computation within $5\times$ the time of the disjunctive chase, since the latter gives an idea of the complexity of brute-force search.

We summarize results on the size of reformulations in Figure 1, with further details on the experiments available in the appendix. On the left we see the average output size of all tools when the "ideal target reformulation" is a CNF, a DNF, or an alternation of disjunction and conjunction, with the number of alternations varying from 2 to 5. For each category the results are averages over more than 600 reformulation problems randomly generated. The bars show the averages only over problems where the tools could prove the existence of a reformulation at all, with the percentages where the theorem prover could find any reformulation prior to timeout shown below each bar. The disjunctive chase approach always generates reformulations as DNF, so it is not surprising that it performs badly in cases where the ideal reformulation is a CNF or alternates connectives. Even in the case of DNF, it is inferior to that of interpolation-based procedures; intuitively the disjunctive chase fails to identify succinct representations achieved by leveraging repetition of atoms across different minimal models. Reformulation using Vampire end-to-end failed to find any reformulation within the timeout in a significant percentage of the cases, and further the size of reformulation was, in average, quite poor. Due to the integrated nature of this approach we can only speculate on the reason; but our conjecture is that the interpolation procedures of Vampire are very geared towards examples coming from verification (and in particular the use of interpreted theories, as in [Hoder *et al.*, 2010; 2012]). Our experiments can be seen as indicating a strong difference between interpolation-for-reformulation and interpolation-for-verification.

The remaining entries in the left figure use our own interpolation implementation over distinct theorem provers. They show the superiority of MathSat over E for all interpolation algorithms, and the superiority of our optimized Huang algorithm and McMillan's algorithm over the other implementations. Thus the right of Figure 1 contains a finer comparison of these two algorithms on top of MathSat. We see that Optimized Huang is superior on most examples, although not all. We also note that our experiments show McMillan's algorithm to be much more robust to inefficiencies in the underlying theorem prover – that is, even with long proofs it can produce relatively smaller reformulations. This may be explained by the compositionality of McMillan's algorithm, with the cases for resolution being simple combination of components. On the contrary, optimized Huang introduces a literal in the interpolant at each LEFT-RIGHT and RIGHT-LEFT resolution step. This may consequently induce some redundancy in the obtained formula that is not solvable by our current simplification techniques.

5 Conclusion

Our theoretical results shows that the most straightforward algorithms for $\exists\mathsf{REF}$ and $\mathsf{REF}^{\leq}$ provide the optimal complexity. While $\exists\mathsf{REF}$ can be polynomially reduced to entailment over the corresponding logic, our lower bounds show that $\mathsf{REF}^{\leq}$ requires an additional layer of non-determinism. There is no obvious implementation other than brute-force for this. In our experiments we focused on the interpolation-based approach, showing that for propositional theories good performance can be achieved with an optimized resolution-based interpolation algorithm on top of a high-performance

theorem-prover. Limitations in the proofs exposed by first-order theorem-provers currently represent a bottleneck to application of this approach to richer logics. An interesting direction is to allow the search heuristics of a theorem-prover to be guided by the size of partial-interpolants, thus giving "interpolant-driven theorem-proving". We are examining the implementation of this over open-source provers such as E.

References

[Anonymous, 2017] Anonymous. Query reformulation: Theory and practice, 2017. Available at `www.github.com/qreform/ijcai17qreform`.

[Benedikt *et al.*, 2016] Michael Benedikt, Balden Ten Cate, Julien Leblay, and Efthymia Tsamoura. *Generating plans from proofs: the interpolation-based approach to query reformulation*. Morgan Claypool, 2016.

[Bonacina and Johansson, 2011] Maria Paola Bonacina and Moa Johansson. On interpolation in decision procedures. In *Automated Reasoning with Analytic Tableaux and Related Methods*. 2011.

[Buchfuhrer and Umans, 2011] David Buchfuhrer and Christopher Umans. The complexity of boolean formula minimization. *J. Comput. Syst. Sci.*, 77(1):142–153, 2011.

[Calì *et al.*, 2010] Andrea Calì, Georg Gottlob, and Andreas Pieris. Query Answering under Non-guarded Rules in Datalog+/-. In *Web Reasoning and Rule Systems*, 2010.

[Cimatti *et al.*, 2013] Alessandro Cimatti, Alberto Griggio, Bastiaan Joost Schaafsma, and Roberto Sebastiani. The MathSAT5 SMT solver. In *TACAS*, 2013.

[Craig, 1957a] William Craig. Linear reasoning. a new form of the Herbrand-Gentzen theorem. *The Journal of Symbolic Logic*, 22(03):250–268, 1957.

[Craig, 1957b] William Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *Journal of Symbolic Logic*, 22(3):269–285, 1957.

[Deutsch *et al.*, 2006] Alin Deutsch, Lucian Popa, and Val Tannen. Query reformulation with constraints. *SIGMOD Record*, 35(1):65–73, 2006.

[Deutsch *et al.*, 2008] Alin Deutsch, Alan Nash, and Jeff Remmel. The chase revisited. In *PODS*, 2008.

[Eiter *et al.*, 1997] Thomas Eiter, Georg Gottlob, and Heikki Mannila. Disjunctive Datalog. *ACM Trans. Database Syst.*, 22(3):364–418, 1997.

[Fagin *et al.*, 2005] Ronald Fagin, Phokion G. Kolaitis, Renee J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005.

[Hoder *et al.*, 2010] Kryštof Hoder, Laura Kovács, and Andrei Voronkov. *Interpolation and Symbol Elimination in Vampire*. 2010.

[Hoder *et al.*, 2012] Krystof Hoder, Andreas Holzer, Laura Kovács, and Andrei Voronkov. Vinter: A Vampire-Based tool for interpolation. In *APLAS*, 2012.

[Huang, 1995] Guoxiang Huang. Constructing Craig interpolation formulas. In *Computing and Combinatorics*. 1995.

[Ileana *et al.*, 2014] Ioana Ileana, Bogdan Cautis, Alin Deutsch, and Yannis Katsis. Complete yet practical search for minimal query reformulations under constraints. In *SIGMOD*, 2014.

[Jiménez-Ruiz *et al.*, 2008] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Ulrike Sattler, Thomas Schneider, and Rafael Berlanga. Safe and economic re-use of ontologies: A logic-based methodology and tool support. In *ESWC*, 2008.

[Konev *et al.*, 2009] Boris Konev, Dirk Walther, and Frank Wolter. Forgetting and uniform interpolation in large-scale description logic terminologies. In *IJCAI*, 2009.

[Kontchakov *et al.*, 2009] Roman Kontchakov, Luca Pulina, Ulrike Sattler, Thomas Schneider, Petra Selmer, Frank Wolter, and Michael Zakharyaschev. Minimal module extraction from DL-Lite ontologies using QBF solvers. In *IJCAI*, 2009.

[Koopmann and Nikitina, 2017] Patrick Koopmann and Nadeschda Nikitina. Small is beautiful: Computing minimal equivalent EL concepts. In *AAAI*, 2017.

[Lang *et al.*, 2003] Jérôme Lang, Paolo Liberatore, and Pierre Marquis. Propositional independence: Formula-variable independence and forgetting. *J. Artif. Intell. Res. (JAIR)*, 18:391–443, 2003.

[Leone *et al.*, 2006] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The DLV system for knowledge representation and reasoning. *TOCL*, 7(3):499–562, 2006.

[Lutz and Wolter, 2011] Carsten Lutz and Frank Wolter. Foundations for uniform interpolation and forgetting in expressive description logics. In *IJCAI*, 2011.

[Lutz *et al.*, 2012] Carsten Lutz, Inanç Seylan, and Frank Wolter. An automata-theoretic approach to uniform interpolation and approximation in the description logic EL. In *KR*, 2012.

[Lyndon, 1959] Roger C. Lyndon. An interpolation theorem in the predicate calculus. *Pacific Journal of Mathemantics*, 9:129–142, 1959.

[McMillan, 2003] K.L. McMillan. Interpolation and sat-based model checking. In *CAV*. 2003.

[Meier, 2014] Michael Meier. The backchase revisited. *VLDB Journal*, 23(3):495–516, 2014.

[Nikitina and Rudolph, 2014] Nadeschda Nikitina and Sebastian Rudolph. (non-)succinctness of uniform interpolants of general terminologies in the description logic EL. *Artif. Intell.*, 215:120–140, 2014.

[Popa, 2000] Lucian Popa. *Object/Relational Query Optimization with Chase and Backchase*. PhD thesis, U. Penn., 2000.

[Sagiv and Yannakakis, 1980] Yehoshua Sagiv and Mihalis Yannakakis. "Equivalences Among Relational Expressions with the Union and Difference Operators". *Journal of the ACM*, 27(4):633–655, 1980.

[Schulz, 2013] Stephan Schulz. System Description: E 1.8. In *LPAR*, 2013.

[Shmueli, 1993] Oded Shmueli. Equivalence of datalog queries is undecidable. *The Journal of Logic Programming*, 15(3):231 – 241, 1993.

[Toman and Weddell, 2011] D. Toman and G. Weddell. *Fundamentals of Physical Design and Query Compilation*. Morgan Claypool, 2011.

[Umans, 2001] Christopher Umans. The minimum equivalent DNF problem and shortest implicants. *J. Comput. Syst. Sci.*, 63(4):597–611, 2001.

Proofs for reformulation results for $\exists$REF

We first show both upper and lower bounds for $\exists$REF for general propositional logic formulas:

Proposition 4. *The problems* $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{PROP})$*,* $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{PROP}^+)$*, and* $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{PROP}^{\mathsf{CQ}})$ *as* Q *ranges over conjunctions of propositions and* Σ *over propositional sentences, are all* CoNP*-complete.*

Proof. The upper bound for the first two comes from Corollary 1. For the third, we can compute in CoNP the list of propositions in $\mathcal{V}$ that are implied by $Q \wedge \Sigma$. It suffices to check whether the conjunction of these propositions and Σ implies Q, which can also be done in CoNP.

The lower bound for each is a reduction from propositional validity. Given φ for which we want to determine validity, we let Σ be $Q \leftrightarrow (\varphi \vee A)$ where A is a proposition not in φ, and we let $\mathcal{V} =$ If φ is valid then Q is equivalent to True for models satisfying Σ, and thus we can reformulate Q as True. Conversely if φ can be reformulated over $\mathcal{V}$, the reformulation must be True or False, and Σ ensures that it cannot be False. Thus φ must be valid according to Σ and this easily implies that φ is valid unconditionally. □

We get the same bounds for $\vee\mathsf{Horn}$ formulas.

Proposition 5. *The problems* $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{PROP})$*,* $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{PROP}^+)$ *and* $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{PROP}^{\mathsf{CQ}})$ *as* Q *ranges over conjunctions of propositions and* Σ *over* $\vee\mathsf{Horn}$*, are all* CoNP*-complete.*

Proof. First, we show that entailment of propositions via $\vee\mathsf{Horn}$ theories is CoNP hard. Given an arbitrary CNF formula φ we can convert each clause that contains a positive literal into a $\vee\mathsf{Horn}$ formula. Each "all-negative" clause of the form $\bigvee \neg P_i$ we convert into a formula $\bigwedge_i P_i \rightarrow Q'$. We claim that φ is unsatisfiable if and only if the formulas above imply Q'. In one direction if, there is a model of the formulas above and $\neg Q'$, then clearly the all-negative clauses must hold, and thus φ is satisfiable. Conversely, if there is a model of φ, we extend it to make Q' false to get a counterexample to entailment.

Next we note that entailment of an atomic proposition Q from $\vee\mathsf{Horn}$ formulas can be reduced to reformulation with $\vee\mathsf{Horn}$ formulas. Given Σ and Q we simply add $Q \rightarrow Q'$ for Q' a fresh symbol, and ask whether Q' can be reformulated over the vocabulary $\{Q\}$. If Q' can be reformulated, then clearly the reformulation must be Q, and thus Σ entails Q. Conversely, if Σ entails Q then clearly Q' is equivalent to Q modulo Σ. □

When we turn to propositional horn formulas, entailment becomes P. Using Corollary 1 this gives us P bounds for reformulation over PROP and PROP^+. Looking at one proposition in $\mathcal{V}$ at a time, we get a P bound for reformulating over $\mathsf{PROP}^{\mathsf{CQ}}$. In fact, it is easily seen (a variation of [Sagiv and Yannakakis, 1980]) that if a conjunction Q can be reformulated in PROP^+ over a theory given by a set of $\mathsf{PROP}^{\mathsf{HORN}}$ formulas, then it can be reformulated as a conjunction. In summary, we see:

Proposition 6. *: The problems* $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{PROP})$*,* $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{PROP}^+)$ *and* $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{PROP}^{\mathsf{CQ}})$ *as* Q *ranges over conjunctions of propositions and* Σ *over* $\mathsf{PROP}^{\mathsf{HORN}}$*, are all in* P*.*

.1 $\exists$REF for predicate logic theories

We begin with the case of TGDs.

We first note that for any class of TGDs $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{UCQ})$ and $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{CQ})$ are the same. If a UCQ Q' is equivalent to Q modulo TGDs Σ, Q' must hold in the case of Σ, and thus one of its disjuncts must hold in the chse; therefore one of the disjuncts must be equivalent modulo Σ.

Proposition 7. *The problems* $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{FO})$*,* $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{UCQ})$ *and* $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{CQ})$ *as* Q *ranges over CQs and* Σ *over* $\mathsf{FullTGD}$ *are all* ExpTime*-complete. When the arity is fixed, the problems become* NP *complete.*

Proof. The reduction to entailment gives the ExpTime and NP upper bounds, since chasing with full TGDs is simply a fixpoint computation as in Datalog evaluation. Conversely, we can reduce from Datalog evaluation to get ExpTime-completeness and NP-completeness. Consider a Datalog program P with goal predicate G which we want to evaluate on an input instance D consisting of elements $\{d_1 \dots d_k\}$. Let Q be $\exists v_1 \dots v_k \; DOM(v_1 \dots v_k)$, where DOM is a another new relation of arity k. We let Σ be formed by:

- turning each rule of P into a full TGD,
- adding rules $DOM(x_1 \dots x_d) \rightarrow A(x_{j_1} \dots x_{j_k})$ for each fact $A_i(d_{j_1} \dots d_{j_k}) \in D$.
- adding a rule $DOM(x_1 \dots x_k) \wedge G() \rightarrow G'(x_1 \dots x_k)$
- adding rule $G'(x_1 \dots x_k) \rightarrow DOM(x_1 \dots x_k)$

Let $\mathcal{V} = \{G'\}$ and consider the query

$$Q = \exists v_1 \dots v_k \; DOM(v_1 \dots v_k)$$

We claim that the goal predicate of P is true on D if and only if Q has a Σ-reformulation over $\mathcal{V}$.

In one direction, suppose the goal predicate of P evaluates to true on D. We claim that $Q' = \exists v_1 \dots v_k \; G'(v_1 \dots v_k)$ is a Σ-reformulation of Q over $\mathcal{V}$. If Q' holds on an instance I satisfying Σ then the last rule implies that Q holds. Conversely if Q holds then the facts corresponding to D are populated in I by the first set of rules; and thus $G()$ holds using the second set of rules. Finally Q' holds using the last rule.

Conversely suppose Q has a monotone reformulation Q' over $\mathcal{V}$. Let I_0 be the canonical database of Q, consisting of a single fact $DOM(c_1 \dots c_k)$. It is clear that Q' must have some atom containing G', since Q is neither a tautology or a contradiction relative to Σ. Thus in particular Q' must be derived from I_0 using the rules of Σ. Clearly a derivation must use the third rule to generate the G' atom. But this can only happen if $G()$ is derived. From a derivation of $G()$ from the fact $DOM(c_1 \dots c_k)$ we can read off a derivation of the goal predicate of P on D as required. □

Weakly-acyclic TGDs. We now consider weakly acyclic TGDs. Entailment of CQs under these sentences is 2EXPTIME-complete [Calì *et al.*, 2010], even when the arity of the schema is fixed. Thus using the Craig and Lyndon reductions, we get that $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{FO}$ and $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{UCQ})$ are in 2EXPTIME. A matching lower bound follows by a reduction from entailment, similar to those above.

Full disjunctive TGDs. For arbitrary FullDisTGD entailment is CONEXPTIME-complete. Although we imagine this has been known before, we provide an argument here. Note that the combined complexity of the "possibility problem" for Disjunctive Datalog has been examined in detail in [Eiter *et al.*, 1997]; but this does not co-incide either with entailment or its negation.

The CONEXPTIME upper bound is simply by universally choosing disjuncts in the head to fire and then proceeding until a fixpoint is reached. The lower bound is by direct reduction from the acceptance problem for a NEXPTIME machine. We give a sketch of the argument here. Consider a non-deterministic machine M running in time $2^{p(m)}$ for polynomial p and an input w of length m to the complement. Letting $n = p(m)$, we code the value on a tape of such a machine with relations:

- $\mathsf{IsOne}(t_1 \dots t_n, s_1 \dots s_n)$ representing that at time $\vec{t}$, the cell at index $\vec{s}$ has value 1, and similarly $\mathsf{IsZero}(\vec{t}, \vec{s})$ saying that the cell has value 0
- $\mathsf{StateIs}_q(t_1 \dots t_n)$ asserting that the state at time $t_1 \dots t_n$ is q
- $\mathsf{Head}(t_1 \dots t_n, s_1 \dots s_n)$ stating that at time $t_1 \dots t_n$ the head is on s.
- $\mathsf{IsOne}(x)$ and $\mathsf{IsZero}(x)$ stating that bit x represents 1 (resp. 0).

We will have relations $\mathsf{AllEq}_i(x_1 \dots x_i, y_1 \dots y_i)$ representing that $\vec{x}$ and $\vec{y}$ agree on all digits. The following rules will generate the needed equalities, assuming that we begin with two distinct elements inhabiting IsOne and IsZero:

$$\mathsf{IsOne}(x) \wedge \mathsf{IsOne}(y) \vee \mathsf{IsZero}(x) \vee \mathsf{IsZero}(y) \rightarrow \mathsf{AllEq}_1(x, y)$$
$$\mathsf{AllEq}_i(x_1 \dots x_i, y_1 \dots y_i) \wedge [\mathsf{IsOne}(x) \wedge \mathsf{IsOne}(y) \vee \mathsf{IsZero}(x) \wedge \mathsf{IsZero}(y)] \rightarrow \mathsf{AllEq}_{i+1}(x_1 \dots x_{i+1}, y_1 \dots y_{i+1})$$

Using this we have successor relation $\succ (x_1 \dots x_n, y_1 \dots y_n)$ and a set of rules that code its semantics;

$$\bigwedge_{i \leq j} \mathsf{IsOne}(x_i) \wedge \mathsf{IsZero}(x_{j+1}) \wedge \bigwedge_{i \leq j} \mathsf{IsZero}(y_1 \dots y_i)$$
$$\wedge \mathsf{IsOne}(y_{j+1}) \wedge \mathsf{AllEq}_{n-(j+2)}(x_{j+2} \dots x_n, y_{j+2} \dots y_n)$$
$$\rightarrow \succ (x_1 \dots x_n, y_1 \dots y_n)$$

Now given an exponentially long successor relation we can easily write disjunctive rules stating that the configurations satisfy the transition relations.

When the arity is fixed entailment becomes CONP-complete by an analogous argument.

Thus the Craig reduction gives CONEXPTIME and Π_2^p upper bounds for $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{FO}$ and $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{UCQ})$, with the lower bounds coming via reduction from entailment.

We now consider $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{CQ})$, which we claim is undecidable. We show undecidability of a simpler problem: determining if there exists a CQ Q' that is a Σ-reformulation over $\mathcal{V}$ of Q, where Σ consists of full TGDs and Q is a UCQ. This problem can easily be reduced to $\exists\mathsf{REF}(Q, \Sigma, \mathcal{V}, \mathsf{CQ})$, by creating a single disjunctive rule relating Q to its disjuncts. We show that the problem above is undecidable via a reduction from Datalog containment, shown undecidable in [Shmueli, 1993].

Given Q_1 and Q_2 in Datalog with goal predicates G_1 and G_2, respectively, we consider $Q = G \wedge G_1 \vee G_2$ $Q = (G \wedge G_1) \vee G_2$ where G is a new relation and our full TGDs Σ correspond to the two Datalog programs. We claim that Q can be reformulated modulo Σ to a CQ (over any vocabulary) iff Q_1 is contained in Q_2. Clearly if Q_1 is contained in Q_2, then Q is equivalent to the CQ G_2 modulo Σ. In the other direction, if Q is equivalent to a CQ Q' then Q' must be equivalent to one of the disjuncts of Q modulo Σ: as mentioned earlier, this is because Q must hold in the chase of the canonical database of Q', and hence one of the disjuncts of Q must hold in the chase. If Q' is equivalent to $G \wedge G_1$, we know that Q is equivalent to $G \wedge G_1$ modulo Σ. But this would mean that G_2 implies $G \wedge G_1$ modulo Σ, which is impossible since the relation G is never mentioned in a rule

of Σ. Hence we must have Q' is equivalent to G_2 modulo Σ. This means that $G \wedge G_1$ must imply G_2 modulo Σ. Since G is fresh this means that G_1 must imply G_2 modulo Σ. From this we see that the Datalog program given by Q_1, G_1 is contained in that of Q_2, G_2.

Proofs for $\mathsf{REF}^{\leq}$

Proposition 8. *The problems* $\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathsf{PROP})$ *and* $\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathsf{PROP}^{+})$ $\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathsf{PROP}^{\mathsf{UCQ}})$ *as Q ranges over conjunctions of propositions and Σ over propositional sentences, are Σ_2^p-complete.*

Proof. We first consider $\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathsf{PROP})$. [Buchfuhrer and Umans, 2011] shows that minimization of a propositional formula in DNF within the set of all formulas is Σ_2^p-hard. We can reduce this problem to $\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathsf{PROP})$ by taking a DNF $\varphi = \bigvee_{i \leq n} \varphi_i$ and producing:

- query Q, where Q is a new symbol
- Σ equivalent to $Q \leftrightarrow R_i$ and $R_i \leftrightarrow \varphi_i$
- target alphabet the literals of φ

It is clear that reformulations of Q correspond to formulas equivalent to φ.

For the second, we perform the same reduction, but we add also copies P_i' for each literal in φ, adding to Σ additional formulas $\neg P_i \leftrightarrow P_i'$, and take our target vocabulary to use both copies of the literals. Thus formulas equivalent to φ correspond to monotone reformulations of Q of the same size.

For the third we make use of the fact that finding a minimal equivalent DNF is Σ_2^p-hard [Umans, 2001], and the "monotonization" reduction above. □

Proposition 9. *The problems* $\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathsf{PROP}, k)$, $\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathsf{PROP}^{+}, k)$, $\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathsf{PROP}^{\mathsf{UCQ}}, k)$, $\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathsf{PROP}^{\mathsf{CQ}}, k)$ *as Q ranges over conjunctions of propositions and Σ over propositional Horn sentences, are* NP*-complete.*

Proof. We reduce from the set cover problem, which has input a family $S_1 \ldots S_n$ of finite sets and a number k, and asked whether there are at most k S_i's whose union covers each element in $\bigcup_{i \leq n} S_i$.

We start by giving a reduction that works for the first 3 problems above. Given an instance of set cover as above, let $\{x_1 \ldots x_d\}$ be the union of the S_i. Our reduction produces a formulas over propositions $x_i : i \leq d$, $S_i : i \leq n$ and a symbol Q. The theory Σ consists of:

$$Q \leftrightarrow \bigvee_{i \leq d} x_i$$

$$S_i \leftrightarrow \bigvee_{k \in S_i} x_j$$

The signature $\mathcal{V}$ consists of $\{S_1 \ldots S_n\}$. From a set cover with $S_{j_1} \ldots S_{j_k}$ we can use the reformulation $\bigvee_i S_{j_i}$, and from a reformulation with at most k S_i's, we can read off a set cover with the same S_i's.

In the above reformulation problem, we needed a reformulation that used disjunction. For the fourth problem, we need to consider reformulations that use only conjunction. However we can "dualize" the problem above. We have extra propositions R and $S_i' i \leq n$, and add to Σ formulas:

$$R \leftrightarrow \neg Q$$

$$S_i' \leftrightarrow \neg S_i$$

A conjunctive reformulation of R with the S_i''s will correspond to a disjunctive reformulation of Q with the S_i's. □

We now give the proof of hardness with $\vee$Horn formulas (Theorem 1):

$\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathsf{PROP}^{+}, k)$, $\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathsf{PROP}^{\mathsf{UCQ}}, k)$ and $\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathsf{PROP}^{\mathsf{CQ}}, k)$ are all Σ_2^p-hard when Σ ranges over $\vee$Horn formulas.

Proof. For simplicity, we assume that the size of a positive propositional formula is the number of occurrences of propositional variables in this formula.

The proof is by reduction of $\exists\forall$3SAT problem. Suppose $\varphi = \exists\bar{x}\forall\bar{y}\neg\psi$ is formula where $\bar{x} = x_1, \ldots, x_k$ and $\bar{y} = y_1, \ldots, y_m$ are propositional variables, and ψ is a formula in 3CNF—that is, a conjunction of clauses $\ell^1 \vee \ell^2 \vee \ell^3$ with each ℓ^j a variable from $\bar{x} \cup \bar{y}$ or the negation of such a variable.

We construct a set of propositional disjunctive Horn clauses Σ and a set of propositional variables $\mathcal{V}$ such that φ is valid if and only if there exists a positive formula Q' over $\mathcal{V}$ of size k such that Q' is equivalent to a propositional variable Q over Σ.

In particular, the vocabulary $\mathcal{V}$ consists of variables TrueX_i and FalseX_i for each x_i.

On the high level, the clauses in Σ guarantee that

1. Q implies all the variables in $\mathcal{V}$, so the equivalence of Q and Q' over Σ boils down to the containment of Q' in Q over Σ;
2. containment of Q' in Q over Σ is only possible if Q' implies either TrueX_i or FalseX_i for each x_i; since the size of Q' is bounded by k, it means that Q' can only be of the form $\mathsf{AsX}_1 \wedge \cdots \wedge \mathsf{AsX}_k$, for $\mathsf{AsX}_i \in \{\mathsf{TrueX}_i, \mathsf{FalseX}_i\}$—that is, Q' corresponds to an assignment of variables x_i;
3. if a dataset and Σ imply TrueX_i or FalseX_i for each x_i, then they imply TrueY_i or FalseY_i for each y_i in $\bar{y}$, which corresponds to an assignment of the universally quantified variables in φ;
4. if a chase of a dataset implies a *falsification* of a clause in ψ under this encoding, which corresponds to falsification of the whole ψ, then it implies Q as well.

In the following formal definition of Σ we use disjunction on the left of implication and conjunction on the right for brevity. The first group of Σ, corresponding to the guarantee 1 above, consists of the rules

$$\mathsf{Q} \;\rightarrow\; \mathsf{TrueX}_i \wedge \mathsf{FalseX}_i, \qquad \text{for each } i = 1, \ldots, k. \tag{1}$$

The second group, that corresponds to the guarantee 2 together with the next groups, consists of the rules

$$\begin{aligned} \mathsf{TrueX}_1 \vee \mathsf{FalseX}_1 &\;\rightarrow\; \mathsf{XAssignedUntil}_1, \\ \mathsf{XAssignedUntil}_{i-1} \wedge (\mathsf{TrueX}_i \vee \mathsf{FalseX}_i) &\;\rightarrow\; \mathsf{XAssignedUntil}_i, \\ &\qquad \text{for each } i = 2, \ldots, k, \\ \mathsf{XAssignedUntil}_k &\;\rightarrow\; \mathsf{XAssignedAll}. \end{aligned} \tag{2}$$

The third group, corresponding to guarantee 3, consists of the rules

$$\mathsf{XAssignedAll} \;\rightarrow\; \mathsf{TrueY}_j \vee \mathsf{FalseY}_j, \qquad \text{for each } j = 1, \ldots, m. \tag{3}$$

Finally, the fourth group, corresponding to guarantee 4 (and a part of the guarantee 2), consists of the rules

$$\mathsf{XAssignedAll} \wedge \mathsf{AsZ}^1 \wedge \mathsf{AsZ}^2 \wedge \mathsf{AsZ}^3 \;\rightarrow\; \mathsf{Q}, \tag{4}$$

for each clause $\ell^1 \vee \ell^2 \vee \ell^3$ in ψ, where, for each $j = 1, 2, 3$, AsZ^j is

- TrueX_i if ℓ^j is a negation of an existential variable $x_i \in \bar{x}$,
- FalseX_i if ℓ^j is an existential variable $x_i \in \bar{x}$,
- TrueY_i if ℓ^j is a negation of a universal variable $y_i \in \bar{y}$,
- FalseY_i if ℓ^j is a universal variable $y_i \in \bar{y}$.

Next we give a formal proof of the correctness of the construction.

Suppose first that φ is valid. We need to prove that there exists Q' equivalent to Q. To this end, consider an assignment of $\bar{x}$ witnessing the validity of φ and let $Q' = \mathsf{AsX}_1 \wedge \cdots \wedge \mathsf{AsX}_k$, where, for each $i = 1, \ldots, k$, $\mathsf{AsX}_i = \mathsf{TrueX}_i$ if x_i evaluates to $true$ by the assignment and $\mathsf{AsX}_i = \mathsf{FalseX}_i$ otherwise. Rules (1) guarantee that any dataset that contains Q and satisfies Σ contains also all variables in Q'. Hence we only need to prove that any dataset that implies Q' and satisfies Σ contains also Q. To this end, consider a dataset D that contains all variables in Q' and satisfies Σ. Rules (2) guarantee that D contains $\mathsf{XAssignedAll}$. Hence, rules (3) guarantee that D contains also TrueY_i or FalseY_i for each y_i in $\bar{y}$. Hence, overall, D encodes one or more extensions of the assignment of $\bar{x}$ to $\bar{y}$. Fix one of these extensions. Since φ is valid, there exists a clause in ψ that evaluates to $false$ under the overall assignment of $\bar{x} \cup \bar{y}$. Therefore, rule (4), corresponding to this clause, guarantees that D contains Q, as required.

Suppose now that there exists a positive query Q' of size at most k equivalent to Q. We need to prove that φ is valid.

First, we prove the following claim.

Claim 1. *Query Q' implies $\mathsf{TrueX}_i \vee \mathsf{FalseX}_i$ for all $i \in \{1, \ldots, k\}$ (the implication here does not take Σ into account— that is, we claim that $Q' \rightarrow \mathsf{TrueX}_i \vee \mathsf{FalseX}_i$, not $Q', \Sigma \rightarrow \mathsf{TrueX}_i \vee \mathsf{FalseX}_i$).*

Proof. Assume, for the sake of contradiction, that this is not the case. Consider the smallest i such that Q' does not imply $\mathsf{TrueX}_i \vee \mathsf{FalseX}_i$, and form the dataset

$$D = \{\mathsf{TrueX}_j, \mathsf{FalseX}_j \mid j \neq i\} \cup \{\mathsf{XAssignedUntil}_j \mid j < i\}.$$

On the one hand, one can see by inspection that D satisfies Σ. We claim that D satisfies Q' as well. Indeed, Q' can be decomposed as a disjunction where each disjunct represents which of TrueX_i, FalseX_i is true:

$$\begin{aligned}[(\mathsf{TrueX}_i \wedge \mathsf{FalseX}_i) \wedge \gamma_1] \\ \vee [\mathsf{TrueX}_i \wedge \gamma_2] \\ \vee [\mathsf{FalseX}_i \wedge \gamma_3] \\ \vee \gamma_4,\end{aligned}$$

where the $\gamma_1, \ldots, \gamma_4$ are formulas that are either $false$ or monotone formulas in TrueX_j, FalseX_j for $j \neq i$, such that at least one of $\gamma_1, \ldots, \gamma_4$ is not $false$. If γ_4 is $false$ then Q' implies $\mathsf{TrueX}_i \vee \mathsf{FalseX}_i$, contrary to assumption. But if γ_4 is a monotone formula in $\mathsf{TrueX}_j, \mathsf{FalseX}_j$ for $j \neq i$ then D must satisfy γ_4 and hence must satisfy overall Q'.

On the other hand, D does not contain the variable Q—that is, does not imply Q as a query. This contradicts the fact that Q' and Q are equivalent for instances satisfying the formulas, because D is a witness for non-equivalence. □

Since by Claim 1 query Q' implies $\mathsf{TrueX}_i \vee \mathsf{FalseX}_i$ for all i Q' is either True or must mention, for each i, one of $\mathsf{TrueX}_i \vee \mathsf{FalseX}_i$. In the case Q' is True, the conclusion is obvious. In the other case, since the size of Q' is bounded by k, we can conclude that $Q' = \mathsf{AsX}_1 \wedge \cdots \wedge \mathsf{AsX}_k$, for $\mathsf{AsX}_i \in \{\mathsf{TrueX}_i, \mathsf{FalseX}_i\}$.

Consider the assignment of $\bar{x}$ corresponding to Q'. Next we prove that ψ evaluates to $false$ under any extension of this assignment to $\bar{y}$. Consider such an extension and a dataset D consisting of

- all the variables in Q';
- $\mathsf{XAssignedUntil}_i$ for all $i = 1, \ldots, k$;
- $\mathsf{XAssignedAll}$;
- for all $i = 1, \ldots, m$, TrueY_i if the extension assigns y_i to $true$ and FalseY_i otherwise.

Let D' be the result of chasing D under all the non-disjunctive rules. Such D' satisfies all the rules of Σ, since the disjunctive rules was already satisfied by construction. Also, D' implies Q', so the chase contains Q as well, because Q' and Q are equivalent over Σ. By rules (4), which are the only rules implying Q, this is possible only if there is a clause in ψ that evaluates to $false$ under the overall assignment encoded by D. So ψ evaluates to $false$ as well, as required. □

Full TGDs. When background knowledge is captured using sentences that are Full TGDs, entailment is in EXPTIME and in NP if the arity is bounded. Thus $\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathsf{UCQ}, k)$ can be solved in EXPTIME and in NP using the guess-and-check method explained earlier. The matching hardness follows immediately from the reduction from entailment.

Weakly-acyclic TGDs. For weakly-acyclic TGDs, entailment is in 2EXPTIME even for fixed arity. Since guessing a reformulation of size k can also be simulated in a 2EXPTIME machine, $\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathsf{UCQ}, k)$ and $\mathsf{REF}^{\leq}(Q, \Sigma, \mathcal{V}, \mathsf{CQ}, k)$. are both in 2EXPTIME just by guessing the reformulation and checking it again. Since we always have a lower bound from entailment, the bounds are tight.

Full disjunctive TGDs.

Theorem 3. *Problem* $\mathsf{SMINREF}(Q, \Sigma, \mathcal{V}, \mathsf{FO}_{pos}, k)$ *is* Σ_3^p*-hard, even if the arity of relations in the schema is bounded by 2.*

Proof. For simplicity, we assume that the size of a positive first-order formula is the number of atoms in this formula (the predicates that appear in the reduction are all at most binary).

The proof is by reduction of $\exists\forall\exists$3SAT problem. Suppose $\varphi = \exists\bar{x}\forall\bar{y}\exists\bar{z}\psi$ is a formula where $\bar{x} = x_1, \ldots, x_n$, $\bar{y}$ and $\bar{z}$ are propositional variables, and ψ is a formula in 3CNF—that is, a conjunction of clauses $\ell^1 \vee \ell^2 \vee \ell^3$ with each ℓ^j a variable from $\bar{x} \cup \bar{y} \cup \bar{z}$ or the negation of such a variable.

For each clause $\gamma = \ell^1 \vee \ell^2 \vee \ell^3$ in ψ let α_γ^j, $j = 1, \ldots, 7$, be an enumeration of the assignments to variables in ℓ^1, ℓ^2 and ℓ^3 that satisfy the clause.

Let σ be the conjunction of

$$\mathsf{VTrue}_s(t_s) \wedge \mathsf{VFalse}_s(f_s)$$

for all $s \in \bar{x} \cup \bar{y} \cup \bar{z}$, and

$$\mathsf{ClAssignment}_\gamma^j(v_\gamma^j) \wedge \mathsf{Clause}_\gamma(v_\gamma^j) \wedge \mathsf{Arg}^1(v_\gamma^j, u_\gamma^{j,1}) \wedge \mathsf{Arg}^2(v_\gamma^j, u_\gamma^{j,2}) \wedge \mathsf{Arg}^3(v_\gamma^j, u_\gamma^{j,3}),$$

where $u_\gamma^{j,i}$, $i = 1, 2, 3$, is t_s if the propositional variable s in ℓ^i is assigned to $true$ by α_γ^j and f_s if is assigned to $false$. So, overall, σ contains 7 variables v_γ^j for each clause γ and 2 variables t_s and f_s for each propositional variable s. We also denote $\bar{v}_\sigma$ the tuple of all variables of σ, that is, $\{v_\gamma^1, \ldots, v_\gamma^7, \mid \gamma \text{ in } \psi\} \cup \{t_s, f_s \mid s \in \bar{x} \cup \bar{y} \cup \bar{z}\}$.

So above v_γ^i is a distinct variable corresponding to each assignment to clause γ. The rules of σ intuitively encode the semantics of the clause.

Let ξ be the conjunction of

$$\mathsf{XAssigned}(t_x) \wedge \mathsf{XAssigned}(f_x)$$

for all $x \in \bar{x}$.

Let τ be the conjunction of

$$\mathsf{Clause}_\gamma(w_\gamma) \wedge \mathsf{Arg}^1(w_\gamma, w_{s^1}) \wedge \mathsf{Arg}^2(w_\gamma, w_{s^2}) \wedge \mathsf{Arg}^3(w_\gamma, w_{s^3}),$$

for each clause $\gamma = \ell^1 \vee \ell^2 \vee \ell^3$ in ψ, where $s^i \in \bar{x} \cup \bar{y} \cup \bar{z}$, $i = 1, 2, 3$, is the variable in ℓ^i.

Finally, let π be the conjunction of

$$\begin{array}{ll} \mathsf{XAssigned}(w_x), & \text{for all } x \in \bar{x}, \\ \mathsf{YAssigned}(w_y), & \text{for all } y \in \bar{y}, \text{ and} \\ \mathsf{ZAssigned}(w_z), & \text{for all } z \in \bar{z}. \end{array}$$

Observe that the variables of τ and π are disjoint from those of σ.

Then, let Σ consist of the implication:

$$\sigma \wedge \tau \wedge \pi \rightarrow \xi, \tag{5}$$

the implications

$$\begin{array}{rcl} \sigma \wedge (\mathsf{XAssigned}(t_{x_1}) \vee \mathsf{XAssigned}(f_{x_1})) & \rightarrow & \mathsf{XAssignedUntil}_1(), \\ \sigma \wedge \mathsf{XAssignedUntil}_{i-1}() \wedge & & \\ (\mathsf{XAssigned}(t_{x_i}) \vee \mathsf{XAssigned}(f_{x_i})) & \rightarrow & \mathsf{XAssignedUntil}_i(), \\ & & \text{for each } i = 2, \ldots, n, \\ \mathsf{XAssignedUntil}_n() & \rightarrow & \mathsf{XAssignedAll}(), \end{array} \tag{6}$$

and the implications

$$\sigma \wedge \mathsf{XAssignedAll}() \rightarrow \mathsf{YAssigned}(t_y) \vee \mathsf{YAssigned}(f_y), \qquad \text{for all } y \in \bar{y}, \tag{7}$$

$$\sigma \wedge \mathsf{XAssignedAll}() \rightarrow \mathsf{ZAssigned}(t_z) \wedge \mathsf{ZAssigned}(f_z), \qquad \text{for all } z \in \bar{z}. \tag{8}$$

Finally, let Q be the Boolean CQ that is the existential closure of $\sigma \wedge \tau \wedge \pi$. Let

$$\mathcal{V} = \{\mathsf{ClAssignment}^1_\gamma, \ldots, \mathsf{ClAssignment}^7_\gamma, \mathsf{Clause}_\gamma \mid \gamma \text{ in } \psi\} \cup \{\mathsf{VTrue}_s, \mathsf{VFalse}_s \mid s \in \bar{x} \cup \bar{y} \cup \bar{z}\} \cup \{\mathsf{Arg}^1, \mathsf{Arg}^2, \mathsf{Arg}^3, \mathsf{XAssigned}\},$$

Let $k = |\sigma| + n$, where $|\sigma|$ is the number of atoms in σ; without loss of generality, we assume that k is the restriction on the number of atoms in Q'.

Having completed the construction, we now prove that φ is valid if and only if there exists a positive existential first order Boolean query Q' of size at most k that is equivalent to Q under Σ.

For the forward direction, suppose that φ is valid. That is, there exists an assignment of variables $\bar{x}$ such that for all assignments of $\bar{y}$ there is an assignment of $\bar{z}$ with ψ evaluating to $true$ by the overall assignment. Consider the query Q' defined as follows:

$$\exists \bar{v}_\sigma \; (\sigma \wedge \mathsf{XAssigned}(u_{x_1}) \wedge \cdots \wedge \mathsf{XAssigned}(u_{x_n})),$$

where each u_{x_i}, $i = 1, \ldots, n$, is t_{x_i}, if x_i is assigned to $true$ by the assignment, and f_{x_i}, otherwise. First, note that Q' has precisely k atoms. Therefore, it is enough to prove that Q' is equivalent to Q under Σ. Suppose for the sake of contradiction that this is not the case. We argue that Q and Σ implies Q'. Observe that the body of Q, that is $\sigma \wedge \tau \wedge \pi$, is precisely the body of rule (5), so this rule fires. The head of the rule (5), ξ, is the "assignment" of all variables to both $true$ and $false$ is σ, while Q' conjoins σ with atoms in which only one of $true$ and $false$ assigned for each x. Thus ξ implies Q'.

Therefore, there exists a database D that satisfies both Q' and Σ, but does not satisfy Q. Since D satisfies Q', there exists a homomorphism h from Q' to D. Since D satisfies Σ, by rules (6) it contains $\mathsf{XAssignedUntil}_i()$ for all i and $\mathsf{XAssignedAll}()$. Therefore, by rules (8), D contains either $\mathsf{YAssigned}(h(t_y))$ or $\mathsf{YAssigned}(h(f_y))$ for all $y \in \bar{y}$ and both $\mathsf{ZAssigned}(h(t_z))$ and $\mathsf{ZAssigned}(h(f_z))$ for all $z \in \bar{z}$. Consider any extension of the assignment of $\bar{x}$ to $\bar{y}$ that agrees with D, in the sense that $\mathsf{YAssigned}(h(t_y))$ holds in D if y is assigned to $true$ and $\mathsf{YAssigned}(h(f_y))$ holds otherwise. Since there exists an extension of the assignment to $\bar{z}$ such that ψ evaluates to $true$, it is straightforward to construct a homomorphism from $\tau \cup \pi$ to D. However, there is also a homomorphism from σ to D by construction. Since $\tau \cup \pi$ and σ do not have any variables in common, there is a homomorphism from the whole of Q to D. This contradicts the fact Q does not hold in D. Therefore our assumption was wrong, and Q is indeed equivalent to Q' under Σ, as required.

For the backward direction of the proof, suppose that there exists a positive query Q' of size at most k equivalent to Q relative to Σ. We need to prove that φ is valid.

We make use of the following claims.

Claim 2. *Consider any Q' that is equivalent to Q relative to the constraints Σ (regardless of size). For each $i = 1, \ldots, n$, $Q' \subseteq Q_i$, where*

$$Q_i = \exists \bar{v}_\sigma \; (\sigma \wedge (\mathsf{XAssigned}(t_{x_i}) \vee \mathsf{XAssigned}(f_{x_i}))).$$

Proof. Assume, for the sake of contradiction, that this is not the case, and there is a number i with $Q' \not\subseteq Q_i$. Convert Q' to a UCQ and consider a CQ Q'' in this UCQ such that $Q'' \not\subseteq Q_i$. Let D be the database that is the result of replacing each variable v in Q'' by a fresh constant a_v. Let D' be one of the outputs of the disjunctive chase of D with Σ. On the one hand, D' satisfies both Q' and Σ by construction. On the other hand , we will argue that D' does not satisfy Q. Assume, for the sake of contradiction, that D' satisfies Q. Then, D' contains YAssigned and ZAssigned atoms, which require $\mathsf{XAssignedAll}()$ and, in turn, $\mathsf{XAssignedUntil}_i()$ in D' (because none of these are in D, so should be generated by Σ). But this is only possible, because $\mathsf{XAssignedUntil}_i()$ is not in D and can be obtained only by the corresponding rule in (6). But this rule cannot fire, because it requires Q_i, which is not present in D and is not obtainable by any rules in Σ. So, we conclude that D' does not satisfy Q.

Since D' does not satisfy Q, this means that Q is not equivalent to Q under Σ. So, our assumption was wrong and $Q' \subseteq Q_i$ indeed holds for all i. □

Claim 3. *Suppose Q' is a positive query equivalent to Q relative to Σ, and also that Q' has size at most k. Then Q' has the following form (up to renaming of variables):*

$$\exists \bar{v}_\sigma \, (\sigma \wedge \mathsf{XAssigned}(u_{x_1}) \wedge \cdots \wedge \mathsf{XAssigned}(u_{x_n})),$$

where each u_{x_i}, $i = 1, \ldots, n$, is either t_{x_i} or f_{x_i}.

Proof. Consider again the UCQ that is obtained by flattening Q', and any CQ Q'' in this UCQ. Consider also an output $\bar{Q}''$ from disjunctive chasing of Q'' with Σ. Since Q is equivalent to Q' under Σ, there exists a homomorphism h from Q to $\bar{Q}''$. Since Q contains σ, we have $h(\sigma) \subseteq \bar{Q}''$. However, Σ does not mention any predicates of σ in the heads, so $h(\sigma) \subseteq Q''$. Since Q is equivalent to Q' under Σ, we also know that there exists a homomorphism h' from Q' to any instance $\bar{Q}$ in the disjunctive chase of Q. In particular, $h'(h(\sigma)) \subseteq \bar{Q}$. Again, since Σ does not create atoms over predicates of σ, we have that $h'(h(\sigma)) \subseteq Q$. The conjunction σ has a separate unary predicate for each of its variables, which are not mentioned in τ and π. That is, they are not mentioned in the other parts of Q. So, the composition of h' and h is the identity on the variables of σ, and, in particular, h is injective on these variables.

Overall, we have that each CQ Q'' in the UCQ representing Q' has a subformula (possibly re-arranging the order of existential quantifiers) that is isomorphic to σ. By Claim 2, each such CQ has a homomorphism from at least one of

$$\sigma \wedge \mathsf{XAssigned}(t_{x_i}) \quad \text{and} \quad \sigma \wedge \mathsf{XAssigned}(f_{x_i})$$

for each $i = 1, \ldots, n$.

Let Q'_{min} be a minimal (in the number of atoms) positive existential first order sentence equivalent to Q'. Since σ has a separate unary predicate for each of its variables, it does not have a non-trivial automorphism. Therefore, the body of Q'_{min} has, up to renaming of variables, the form $\sigma \wedge \eta$, such that, for each i, η implies at least one of $\mathsf{XAssigned}(t_{x_i})$, $\mathsf{XAssigned}(f_{x_i})$, $\mathsf{VTrue}_{x_i}(t'_{x_i}) \wedge \mathsf{XAssigned}(t'_{x_i})$, and $\mathsf{VFalse}_{x_i}(f'_{x_i}) \wedge \mathsf{XAssigned}(f'_{x_i})$, where t_{x_i} and f_{x_i} are the variables of σ, while t'_{x_i} and f'_{x_i} are some other variables.

However, we know that the number of atoms in Q' and, therefore, Q'_{min} is bounded by the number of atoms in σ plus n, the size of $\bar{x}$. So, the number of atoms in η is bounded by n, which is only possible if $\eta = \mathsf{XAssigned}(u_{x_1}) \wedge \cdots \wedge \mathsf{XAssigned}(u_{x_n})$ where u_{x_i} is either t_{x_i} or f_{x_i}. Therefore, Q' is equal, up to renaming of variables, to Q'_{min}, and has the form as required in the claim. □

By Claim 3, there exists a uniquely defined assignment of variables $\bar{x}$ corresponding to Q'. Next we prove that for any extension of this assignment to variables $\bar{y}$ there exists an extension to $\bar{z}$ such that ψ holds under the overall assignment. Suppose, for the sake of contradiction, that this is not the case. Thus there is an extension to $\bar{y}$ such that ψ evaluates to $false$ for all extensions to $\bar{z}$. Let D be the database obtained from Q' by instantiating each variable v by a fresh constant a_v. Let D' be the extension of D with the atoms

$\mathsf{XAssignedUntil}_i()$,	for all $i = 1, \ldots, n$,
$\mathsf{XAssignedAll}()$,	
$\mathsf{YAssigned}(a_{t_y})$,	for all $y \in \bar{y}$ assigned to $true$ by the extension,
$\mathsf{YAssigned}(a_{f_y})$,	for all $y \in \bar{y}$ assigned to $false$ by the extension,
$\mathsf{ZAssigned}(a_{t_z}), \mathsf{ZAssigned}(a_{f_z})$,	for each $z \in \bar{z}$.

By construction, D' satisfies Q' and all the rules in Σ, except, possibly, (5). Next we prove that D' does not satisfy Q. Note that if D' does not satisfy Q then (a) D' satisfies all Σ, including (5) (b) since D' satisfies Q', queries Q and Q' are not equivalent, which is a contradiction.

Suppose, for the sake of contradiction, that D' satisfies Q. That is, there is a homomorphism h from the body of Q to D'. By construction,

$$\begin{array}{ll} h(w_\gamma) = a_{v^j_\gamma}, & \text{for each clause } \gamma \text{ in } \psi \text{ and some } j = 1, \ldots, 7, \\ h(w_s) = a_{t_s}, & \text{for each } s \in \bar{x} \cup \bar{y} \text{ assigned to } true, \\ h(w_s) = a_{f_s}, & \text{for each } s \in \bar{x} \cup \bar{y} \text{ assigned to } false, \end{array}$$

and, for each $z \in \bar{z}$,

$$\text{either} \quad h(w_z) = a_{t_z} \quad \text{or} \quad h(w_z) = a_{f_z}.$$

Consider the extension of the assignment of $\bar{x} \cup \bar{y}$ to $\bar{z}$ according to h. By construction, it satisfies ψ, which contradicts our assumption that ψ evaluates to $false$ for all extensions to $\bar{z}$. Therefore, our second assumption was wrong and D' does not satisfy Q (and, therefore, satisfies Σ). As we already said, this contradicts the fact that Q and Q' are equivalent under Σ. Therefore, our first assumption was also wrong, and indeed for any extension of the assignment of $\bar{x}$ defined by Q' to $\bar{y}$ there is an extension to $\bar{z}$ such that ψ holds. This means that φ is valid, as required. □

Correctness of Optimized Huang

We prove Proposition 3:

> The Optimized Huang Algorithm produces an interpolant φ for $\Sigma_1 \models \Sigma_2$, and any negated literal in φ occurs negated in both Σ_1 and Σ_2.

Proof. One can show by induction that if a literal occurs in a proof node n with provenance LEFT, then it is the result of unifying a literal in Σ_1, while if it occurs with provenance RIGHT it is the result of unifying a literal in $\neg\Sigma_2$. From this we can easily see that every relational symbol occurring in φ is present in both Σ_1 and Σ_2, and that if it occurs negated it must be negated in both Σ_1 and Σ_2.

For a clause φ in which each negated or non-negated literal is annotated with LEFT or RIGHT, we let φ_{LEFT} be the disjunction of all LEFT-labelled literals in φ. φ_{LEFT} is defined to be False if there are no such literals. Let φ_{RIGHT} be defined similarly.

We will prove the following invariant that is derived from Bonacina and Johansson [Bonacina and Johansson, 2011]:

For a proof node with formula α and interpolant φ, we have

- $\Sigma_1 \models \varphi \vee \alpha_{\text{LEFT}}$
- $\Sigma_2 \wedge \varphi \models \alpha_{\text{RIGHT}}$

In particular, if $\alpha = \bot$ then $\Sigma_1 \models \varphi$ and $\varphi \models \neg\Sigma_2$ as required.

The base cases are clear for both items. We prove the inductive cases, fixing a non-root node c with parents $p_1 = r_1 \vee \alpha(\rho_1 \dots \rho_n)$ and $p_2 = r_2 \vee \neg\alpha(\tau_1 \dots \tau_n)$ having interpolants φ_1 and φ_2 respectively. Thus $c = \sigma(r_1) \vee \sigma(r_2)$ for some unifier σ.

- We consider the case where provenance is LEFT for $\alpha(\rho_1 \dots \rho_n)$ and $\neg\alpha(\tau_1 \dots \tau_n)$. The interpolant is then $\sigma(\varphi_1) \vee \sigma(\varphi_2)$.
  For the first item, we must show that $\Sigma_1 \models \sigma(\varphi_1) \vee \sigma(\varphi_2) \vee (\sigma(r_1) \vee \sigma(r_2))_{\text{LEFT}}$.
  So assume we have an instance I that satisfies Σ_1, $\neg\sigma(\varphi_1)$, and $\neg\sigma(\varphi_2)$. We know $I \models \sigma(r_1 \vee \alpha(\rho_1 \dots \rho_n))_{\text{LEFT}} = \sigma(r_1)_{\text{LEFT}} \vee \sigma(\alpha(\rho_1 \dots \rho_n))$ by induction (the equality being since α is LEFT). Similarly we know $I \models \sigma(r_2)_{\text{LEFT}} \vee \neg\sigma(\alpha(\tau_1 \dots \tau_n))$ by induction. Since one of $\sigma(\alpha(\rho_1 \dots \rho_n), \neg\sigma(\alpha(\tau_1 \dots \tau_n))$ holds we get $I \models \sigma(r_1 \vee r_2)_{\text{LEFT}}$, as required.
  For the second item, we must show $\Sigma_2 \wedge (\varphi_1 \vee \varphi_2) \models \sigma(r_1 \vee r_2)_{\text{RIGHT}}$. So assume $I \models \Sigma_2$ and, without loss of generality φ_1 (the case where $I \models \varphi_2$ is symmetric). By induction $I \models \sigma(r_1 \vee \alpha(\rho_1 \dots \rho_n))_{\text{RIGHT}} = \sigma(r_1)_{\text{RIGHT}}$. Thus $I \models \sigma(r_1 \vee r_2)_{\text{RIGHT}}$ as required.
- Now consider the case where the provenance is RIGHT for both $\alpha(\rho_1 \dots \rho_n)$ and $\neg\alpha(\tau_1 \dots \tau_n)$. The interpolant is $\sigma(\varphi_1) \wedge \sigma(\varphi_2)$.
  For the first item, we must show that $\Sigma_1 \models \sigma(\varphi_1 \wedge \varphi_2) \vee \sigma(r_1 \vee r_2)_{\text{LEFT}}$. If I does not satisfy $\sigma(\varphi_1 \wedge \varphi_2)$, it must fail one of $\sigma(\varphi_1), \sigma(\varphi_2)$. Without loss of generality, assume I does not satisfy $\sigma(\varphi_1)$. By induction $\Sigma_1 \models \varphi_1 \vee (r_1 \vee \alpha(\rho_1 \dots \rho_n))_{\text{LEFT}}$, and so $I \models (r_1)_{\text{LEFT}}$. Thus $I \models \sigma(r_1 \vee r_2)_{\text{LEFT}}$, as required.
  For the second item, we must show that $\Sigma_2 \wedge \sigma(\varphi_1 \wedge \varphi_2) \models \sigma(r_1 \vee r_2)_{\text{RIGHT}}$, and thus assume $I \models \Sigma_2 \wedge \sigma(\varphi_1 \wedge \varphi_2)$. By induction $I \models \sigma(r_1 \vee \alpha(\rho_1 \dots \rho_n))_{\text{RIGHT}} = \sigma(r_1)_{\text{RIGHT}} \vee \sigma(\alpha(\rho_1 \dots \rho_n))$. Also by induction $I \models \sigma(r_2 \vee \neg\alpha(\tau_1 \dots \tau_n))_{\text{RIGHT}} = \sigma(r_2)_{\text{RIGHT}} \vee \neg\sigma(\alpha(\tau_1 \dots \tau_n))$. Putting these together we can conclude that $I \models \sigma(r_1 \vee r_2)_{\text{RIGHT}}$, hence $I \models \sigma(r_1 \vee r_2)_{\text{RIGHT}}$ as required.
- We turn to the case where the provenance is LEFT for $\alpha(\rho_1 \dots \rho_n)$ and RIGHT for $\neg\alpha(\tau_1 \dots \tau_n)$. Then the interpolant is $\sigma(\varphi_1 \vee (\alpha(\rho_1 \dots \rho_n) \wedge \varphi_2))$.
  For the first item we must show that $\Sigma_1 \models \sigma(\varphi_1) \vee \sigma(\alpha(\rho_1 \dots \rho_n) \wedge \varphi_2 \vee (r_1 \vee r_2)_{\text{LEFT}})$. Suppose I satisfies $\Sigma_1 \wedge \neg\sigma(\varphi_1) \wedge \neg\sigma(\alpha(\rho_1 \dots \rho_n) \wedge \varphi_2)$. Then using the induction hypothesis, $I \models \sigma(r_1 \vee \alpha(\rho_1 \dots \rho_n))_{\text{LEFT}} = \sigma(r_1)_{\text{LEFT}} \vee \sigma(\alpha(\rho_1 \dots \rho_n))$. I must satisfy either $\neg\sigma(\alpha(\rho_1 \dots \rho_n))$ or $\neg\sigma(\varphi_2)$, by assumption. If $I \models \sigma(\varphi_2)$ then by induction $I \models \sigma(r_2 \vee \neg\alpha(\tau_1 \dots \tau_n))_{\text{LEFT}} = \sigma(r_2)_{\text{LEFT}}$, and thus $I \models \sigma(r_1 \vee r_2)_{\text{LEFT}}$ and we are done. On the contrary, if $I \models \neg\sigma(\alpha(\rho_1 \dots \rho_n))$, then since $I \models \sigma(r_1)_{\text{LEFT}} \vee \sigma(\alpha(\rho_1 \dots \rho_n))$ we deduce that $I \models \sigma(r_1)_{\text{LEFT}}$; again $I \models \sigma(r_1 \vee r_2)_{\text{LEFT}}$ and we are done.
  For the second item we need to show that $\Sigma_2 \wedge (\sigma(\varphi_1) \vee \sigma(\alpha(\rho_1 \dots \rho_n) \wedge \varphi_2)) \models \sigma(r_1 \vee r_2)_{\text{RIGHT}}$. Thus assume $I \models \Sigma_2 \wedge \sigma(\varphi_1) \vee \sigma(\alpha(\rho_1 \dots \rho_n) \wedge \varphi_2))$.

In the first case, suppose $I \models \sigma(\varphi_1)$. Then by induction $I \models \sigma(r_1 \vee \alpha(\rho_1 \dots \rho_n))_{\text{RIGHT}} = \sigma(r_1)_{\text{RIGHT}}$ and we are done. In the second case, $I \models \sigma(\alpha(\rho_1 \dots \rho_n)) \wedge \sigma(\varphi_2)$. Then by induction $I \models \sigma(r_2 \wedge \neg\alpha(\tau_1 \dots \tau_n))_{\text{RIGHT}}$. Since $I \models \sigma(\alpha(\rho_1 \dots \rho_n))$ this implies that $I \models \sigma(r_2)_{\text{RIGHT}}$, and from there we easily conclude.

- Finally, we consider the case where the provenance is RIGHT for $\alpha(\rho_1 \dots \rho_n)$ and LEFT for $\neg\alpha(\tau_1 \dots \tau_n)$. The interpolant is $\sigma(\varphi_2) \vee (\neg\sigma(\alpha(\tau_1 \dots \tau_n)) \wedge \sigma(\varphi_1))$.
  For the first item we must show that $\Sigma_1 \models \sigma(\varphi_2) \vee (\neg\sigma(\alpha(\tau_1 \dots \tau_n)) \wedge \sigma(\varphi_1)) \vee \sigma(r_1 \vee r_2)_{\text{LEFT}}$. Towards this goal we assume $I \models \Sigma_1 \wedge \neg\sigma(\varphi_2) \wedge \neg(\neg\sigma(\alpha(\tau_1 \dots \tau_n)) \wedge \sigma(\varphi_1))$. By induction $I \models \sigma(r_2 \vee \neg\alpha(\tau_1 \dots \tau_n))_{\text{LEFT}} = \sigma(r_2)_{\text{LEFT}} \vee \neg\sigma(\alpha(\tau_1 \dots \tau_n))$. If $I \models \sigma(r_2)_{\text{LEFT}}$ we have that $I \models \sigma(r_1 \vee r_2)_{\text{LEFT}}$ and we are done. So we can assume $I \models \neg\sigma(\alpha(\tau_1 \dots \tau_n))$. Sin $I \models \neg(\neg\sigma(\alpha(\tau_1 \dots \tau_n)) \wedge \sigma(\varphi_2))$ we deduce that $I \models \neg\sigma(\varphi_2)$. Thus by induction $I \models \sigma(r_2 \vee \alpha(\tau_1 \dots \tau_n))_{\text{LEFT}} = \sigma(r_2)_{\text{LEFT}}$ and we are done.
  For the second item we must show that $\Sigma_2 \wedge \sigma[\varphi_2 \vee (\neg\alpha(\tau_1 \dots \tau_n) \wedge \varphi_1)] \models \sigma(r_1 \vee r_2)_{\text{RIGHT}}$. Again we assume I satisfies $\Sigma_2 \wedge \sigma[\varphi_2 \vee (\neg\alpha(\tau_1 \dots \tau_n) \wedge \varphi_1)]$. If $I \models \sigma(\varphi_2)$, then inductively $I \models \sigma(r_2 \vee \neg\alpha(\tau_1 \dots \tau_n))_{\text{RIGHT}} = \sigma(r_2)_{\text{RIGHT}}$ and we are done. If $I \models \neg\sigma(\alpha(\tau_1 \dots \tau_n)) \wedge \sigma(\varphi_1)$, then by induction $I \models \sigma(r_1 \vee \alpha(\tau_1 \dots \tau_n))_{\text{RIGHT}} = \sigma(r_1)_{\text{RIGHT}} \vee \sigma(\alpha(\tau_1 \dots \tau_n))$. Using $I \models \neg\sigma(\alpha(\tau_1 \dots \tau_n))$ we can conclude.

□

Propositional Optimizations

We now report about all propositional optimizations that were employed during the inductive computation of the interpolant. To do this, let α be the proposition involved in the resolution of the clause p_1, having interpolant φ_1, with the clause p_2, having interpolant φ_2. In particular, the occurrence of α is positive in p_1, i.e., $\alpha \in p_1$, and negative in p_2, i.e., $\neg\alpha \in p_2$. The obtained clause is $p = (p_1 \setminus \{\alpha\}) \cup (p_2 \setminus \{\neg\alpha\})$.

- If the provenance of α is LEFT in p_1 and that of $\neg\alpha$ is RIGHT in p_2, we have that the interpolant φ associated with p is computed as follows.
  - $\varphi = \varphi_1$, if φ_1 simplifies to True when all occurrences of α in it are replaced by True.
  - $\varphi = \alpha \wedge (\varphi_1' \vee \varphi_2')$, if the previous condition does not hold and φ_1 simplifies to False when all occurrences of α in it are replaced by False, where φ_1' and φ_2' are obtained as the simplifications of φ_1 and φ_2, respectively, after the substitution of all their occurrences α with True.
  - $\varphi = \varphi_1 \vee (\alpha \wedge \varphi_2')$, if the previous two conditions do not hold, where φ_2' is the obtained as the simplification of φ_2 after the substitution of all its occurrences α with True.
- If the provenance of α is RIGHT in p_1 and that of $\neg\alpha$ is LEFT in p_2, we have that the interpolant φ associated with p is computed as follows.
  - $\varphi = \varphi_2$, if φ_2 simplifies to True when all occurrences of α in it are replaced by False.
  - $\varphi = \neg\alpha \wedge (\varphi_1' \vee \varphi_2')$, if the previous condition does not hold and φ_2 simplifies to False when all occurrences of α in it are replaced by True, where φ_1' and φ_2' are obtained as the simplifications of φ_1 and φ_2, respectively, after the substitution of all their occurrences α with False.
  - $\varphi = \varphi_2 \vee (\neg\alpha \wedge \varphi_1')$, if the previous two condition do not hold, where φ_2' is the obtained as the simplification of φ_2 after the substitution of all its occurrences α with False.

Details on Experiments

We are interested in generating test problems for reformulation consisting of an input query Q, a theory Σ in some particular format (e.g. FullDisTGD) and a set of target relations $\mathcal{V}$. To the best of our knowledge, no such testbed exists. The challenge is that we want to generate $Q, \Sigma, \mathcal{V}$ so that: we know Q has a Σ-reformulation over $\mathcal{V}$, but it is non-trivial for a reasoning system to find such a reformulation. For example, if we randomly generated a theory Σ, then surely it would rarely be the case that a reformulation existed. Conversely there are many simple recipes for "re-arranging" Q: we could simply rename relations in Q, added the renamed relations to $\mathcal{V}$, and added equivalences between the original relations and the renamed ones in Σ, and we would guarantee that a reformulation exists. But the simple nature of our encoding might make this atypically easy for a reasoning system.

Our testbed is based on two components. First, we have a *target reformulation generator* TargetReformGen, that generates a random sentence Q' in some desired target language (e.g. CQ, UCQ) over some vocabulary $\mathcal{V}$.

Second, we have a *theory generator* TheoryGen that inputs a target Q' and generates Q, Σ such that Q is equivalent to Q' over Σ.

Our testbed is then formed by first randomly generating target reformulations using TargetReformGen and then feeding each into TheoryGen to get the final reformulation problem.

The generator TargetReformGen produces three types of sentences: CNF, DNF and sentences with a given alternation depth AltDepth. The generator works for both first-order logic and propositional logic. We focus on the parameters that are relevant for the propositional case below. In the propositional CNF case, the generator is parametrised by

1. the total number of conjuncts,
2. the number of literals in each conjunct,
3. the probability of reusing the same relation name in different conjuncts from Q',
4. the probability of having negative literals and, finally,
5. a seed for producing random numbers.

Given these parameters, TargetReformGen first produces a vocabulary $\mathcal{V}$ and then a sentence Q' using relations from $\mathcal{V}$. TargetReformGen works similarly for the DNF case. In the alternating case, TargetReformGen produces AltDepth $+ 1$ CNF and DNF subsentences and "glues" them using conjunction or disjunction such that the alternation depth of the resulting sentence Q' is AltDepth. TargetReformGen is parametrised by (i) the alternation depth AltDepth, (ii) the number of literals in each subsentence, (iii) the type of the first subsentence of Q' (CNF or DNF) and, finally, by the last three arguments also used in the CNF and DNF cases.

Our generator TheoryGen will generate new relations in TheoryDepth layers, and then rules going in each direction between layer i and layer $i + 1$. In the most basic configuration, the rules will match the structure of the input reformulation φ, with disjunctive rules matching disjunction in φ and conjunctive rules matching conjunction in φ.

For example, if our target reformulation is $\varphi = (T_1 \wedge T_2) \vee T_3$, we might add:

$$\begin{aligned} Q &\to R_1 \vee R_2 \\ R_1 &\to T_1 \wedge T_2 \\ R_2 &\to T_3 \\ T_3 &\to R'_2 \\ T_1 \wedge T_2 &\to R'_1 \\ R'_1 &\to Q \\ R'_2 &\to Q \end{aligned}$$

Clearly, the rules ensure that φ is a reformulation of Q. Additional parameters allow the generated rules to be more complex:

- For each "layer" of alternation, we can arrange that its generation occurs not in one steps (e.g. $R_2 \to T_2$) but as a sequence of n steps of TGDs, where n is a parameter.
- We can generate additional "noise" rules, with heads and bodies randomly chosen. Since these are added in addition to the "core rules" above, they will not impact the fact that φ is a reformulation of Q, but will make this harder to detect. The number of such rules is determined by an additional configuration parameter.

In the experiments regarding CNF reformulations, we had both the numbers of conjuncts and literals ranging from 2 to 6. The probability of reusing the same relation was set either to 0 or to $3/4$. Similarly, the probability of using negated literals was either 0 or $5/8$. The same parameters were used for the DNF and alternating reformulations. In particular, for the latter, the nesting of conjunctions and disjunctions varied from 2 to 5.