On Checking Domain Independence

Volha Kerhet and Enrico Franconi

Faculty of Computer Science, Free University of Bozen-Bolzano, Italy
{kerhet,franconi}@inf.unibz.it

Abstract Domain independence is an important property of a formula that guarantees that the truth value of the formula in an interpretation remains the same regardless of the underlying domain of the interpretation. Unfortunately, checking domain independence of a first-order formula is in general undecidable. There have been several attempts to define syntactic fragments of first-order logic characterising domain independent formulas that easily can be checked, but all of them are incomplete.
We reduce the problem of checking domain independence of a general first-order logic formula to checking a standard first-order logic entailment. This method can be applied in any decidable fragment where the formulas participating in the entailment can be expressed.

1 Introduction

Domain independence [1] is an important property of a formula that guarantees that the truth value of the formula in an interpretation remains the same regardless of the underlying domain of the interpretation. The importance of checking domain independence stems from the use of first-order logic as a query language for databases. Since a query can be an arbitrary first-order formula, its answer can be infinite (since the domain is not restricted to be finite) or it may depend on the domain. For example, the query $Q(x) = \neg Student(x)$ over the database $Student(a), Student(b)$, with domain $\{a, b, c\}$ has the answer $\{x \mapsto c\}$, while with domain $\{a, b, c, d\}$ has the answer $\{x \mapsto c, x \mapsto d\}$, and if we change the domain to an infinite one, the answer will be infinite even in presence of such a finite database. Therefore, the notion of *domain independent* queries has been introduced in relational databases: indeed, the above open formula $\neg Student(x)$ turns out not to be domain independent. An example of a domain independent formula would be $\exists x.\ P(x)$, since if it turns out to be true in some interpretation of the unary predicate symbol P with a specific domain, it is also true in all the structures sharing the same interpretation for the predicate P and any other compatible domain. On the other hand, the formula $\forall x.\ P(x)$ is not a domain independent formula, since if it is true in an interpretation with some domain, it is definitely not true in any structure sharing the same interpretation for the predicate P but with a larger domain.

The problem of checking whether a FOL formula is domain independent is undecidable [2]. The well known *safe-range* syntactic fragment of FOL intro-

duced by Codd is an *equally expressive* language; indeed any safe-range formula is domain independent, and any domain independent formula can be easily transformed into a logically equivalent safe-range formula. Intuitively, a formula is safe-range if and only if its variables are bounded by positive predicates or equalities – for the exact syntactical definition see, e.g., [2]. For example, the formula $\forall x.\ G(x) \rightarrow P(x)$ is safe-range, while formula $\exists x.\ (A(x) \vee B(a))$ is not. To check whether a formula is safe-range, the formula is transformed into a logically equivalent *safe-range normal form* and its *range restriction* is computed according to a set of syntax based rules; the range restriction of a formula is a subset of its free variables, and if it coincides with the free variables then the formula is said to be safe-range.

The problem arises since the safe-range property is an incomplete characterisation of domain independence: indeed, the above non-safe-range formula $\exists x.\ (A(x) \vee B(a))$ is logically equivalent to the formula $(\exists x.\ A(x)) \vee B(a)$ which is safe-range, and therefore we can conclude that also the former formula is domain independent. In this paper, we show for the first time a sound and complete procedure to check domain independence of a formula, by reducing this problem to an entailment problem in first-order logic.

All the proofs of the theorems presented in this paper can be found in the technical report [3].

2 Preliminaries

Let $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ be a classical function-free first-order language with equality over a signature $\Sigma = (\mathbb{C}, \mathbb{P})$, where $\mathbb{C}$ is a set of *constants* and $\mathbb{P}$ is a set of *predicates* with associated arities.The arity of a predicate P we denote as $\text{AR}(P)$. We denote as $\sigma(\phi)$ the signature of the formula ϕ, that is all the predicates and constants occurring in ϕ. We denote the set of all variables appearing in ϕ as $\text{VAR}(\phi)$, and the set of the free variables appearing in ϕ as $\text{FREE}(\phi)$; we may use for ϕ the notation $\phi(\bar{x})$, where $\bar{x} = \text{FREE}(\phi)$ is the (possibly empty) set of free variables of the formula. A formula in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ is in *prenex normal form*, if it is written as a string of quantifiers followed by a quantifier-free part. Every formula is equivalent to a formula in prenex normal form and can be converted into it in polynomial time [4].

Let $\bar{x}$ and $\mathbb{S}$ be two countable sets, first of which is some set of variables. We define a *substitution* $\Theta^{\mathbb{S}}_{\bar{x}}$ to be a total function $\bar{x} \mapsto \mathbb{S}$ assigning an element of the set $\mathbb{S}$ to each variable in $\bar{x}$, including the empty substitution ϵ when $\bar{x} = \emptyset$. We can see substitution as (at most) a countable set of assignments of elements from $\mathbb{S}$ to elements from $\bar{x}$. That is, if $\bar{x} = \{x_1, x_2, \ldots\}$, then $\Theta^{\mathbb{S}}_{\bar{x}} := \{x_1 \rightarrow s_1, x_2 \rightarrow s_2, \ldots\}$, where $s_1, s_2, \ldots$ are elements from $\mathbb{S}$ assigned to corresponding variables from $\bar{x}$ by $\Theta^{\mathbb{S}}_{\bar{x}}$.

As usual, an *interpretation* $\mathcal{M} = \langle \Delta^{\mathcal{M}}, \cdot^{\mathcal{M}} \rangle$ includes a non-empty set – the domain $\Delta^{\mathcal{M}}$ – and an interpretation function $\cdot^{\mathcal{M}}$ defined over constants and predicates of the signature. We use standard definitions of validity, satisfiability and entailment of a formula. An *extension* of $\phi(\bar{x})$ in interpretation $\mathcal{M} = \langle \Delta^{\mathcal{M}}, \cdot^{\mathcal{M}} \rangle$,

denoted $(\phi(\bar{x}))^{\mathcal{M}}$, is the set of substitutions which satisfy ϕ in $\mathcal{M}$. That is,

$$(\phi(\bar{x}))^{\mathcal{M}} = \{\Theta^{\Delta}_{\bar{x}} \mid \mathcal{M}, \Theta^{\Delta}_{\bar{x}} \models \phi(\bar{x})\}.$$

If ϕ is closed, then the extension depends on whether ϕ holds in $\mathcal{M}$ or not. Thus, for a closed formula ϕ, $(\phi)^{\mathcal{M}} = \epsilon$, if $\mathcal{M} \models \phi$, and $(\phi)^{\mathcal{M}} = \emptyset$, if $\mathcal{M} \not\models \phi$.

Hereafter let σ' be a subset of $\sigma = \mathbb{P} \cup \mathbb{C}$.

Two interpretations $\mathcal{M}_1 = \langle \Delta_1, \cdot^{\mathcal{M}_1} \rangle$ and $\mathcal{M}_2 = \langle \Delta_2, \cdot^{\mathcal{M}_2} \rangle$ are *σ'-compatible* iff they agree on the interpretations of all the predicates and constants in σ'. That is, for any predicate P and constant c in σ' the following holds:

$$\begin{aligned} P^{\mathcal{M}_2} &= P^{\mathcal{M}_1} \\ c^{\mathcal{M}_2} &= c^{\mathcal{M}_1} \end{aligned}$$

Definition 1. *(**Domain Independence**) Let $\mathcal{M}_1 = \langle \Delta_1, \cdot^{\mathcal{M}_1} \rangle$ and $\mathcal{M}_2 = \langle \Delta_2, \cdot^{\mathcal{M}_2} \rangle$ be any two σ-compatible interpretations, and ϕ be a formula in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$. ϕ is domain independent, if and only if*

$$(\phi)^{\mathcal{M}_1} = (\phi)^{\mathcal{M}_2}$$

The *semantic active domain* of sub-signature σ' in an interpretation $\mathcal{M} = \langle \Delta^{\mathcal{M}}, \cdot^{\mathcal{M}} \rangle$, denoted $adom(\sigma', \mathcal{M})$, is the set of all elements of the domain $\Delta^{\mathcal{M}}$ occurring in interpretations of predicates and constants from σ' in $\mathcal{M}$:

$$adom(\sigma', \mathcal{M}) := \bigcup_{P \in \sigma' \cap \mathbb{P}} \bigcup_{(a_1, \ldots, a_n) \in P^{\mathcal{M}}} \{a_1, \ldots, a_n\} \cup \bigcup_{c \in \sigma' \cap \mathbb{C}} \{c^{\mathcal{M}}\}.$$

3 Checking Domain Independence

In this section we show how to prove domain independence of a formula ϕ by proving a validity of a first-order logic entailment.

We introduce a new unary predicate $Adom_{\sigma'}$. According to the idea, we consider a theory encoding that interpretation of $Adom_{\sigma'}$ in any model of the theory is never empty and equal to the semantic active domain of σ' in this model, if it is not empty. We call such a theory *σ'-active domain theory* and denote it as $\mathcal{A}_{\sigma'}$. Formally, $\mathcal{A}_{\sigma'}$ is the smallest theory satisfying the following four requirements.

1. For every n-ary predicate $P \in \sigma' \cap \mathbb{P}$,

$$\forall \bar{x} \,.\, P(\bar{x}) \rightarrow \bigwedge_{x \in \bar{x}} Adom_{\sigma'}(x) \in \mathcal{A}_{\sigma'}.$$

   In other words, $Adom_{\sigma'}$ contains any domain element that occurs in the interpretation of some predicate from σ'.

2. $Adom_{\sigma'}$ contains all the constants from σ':

$$\bigwedge_{c \in \sigma' \cap \mathbb{C}} Adom_{\sigma'}(c) \in \mathcal{A}_{\sigma'}.$$

3. Let n be a maximal arity among the predicates in σ', and let $\bar{z} = \{z_1, \ldots, z_{n-1}\}$. Then

$$\bigvee_{P \in \sigma' \cap \mathbb{P}} \exists \bar{x}.\, P(\bar{x}) \vee \bigvee_{c \in \sigma' \cap \mathbb{C}} \exists x.\, (x = c) \rightarrow (\forall x. Adom_{\sigma'}(x) \rightarrow$$
$$\exists \bar{z}.\, \bigvee_{P \in \sigma' \cap \mathbb{P}} (P(x, z_1, \ldots, z_{\mathrm{AR}(P)-1}) \vee \ldots \vee P(z_1, \ldots, z_{\mathrm{AR}(P)-1}, x)) \vee \bigvee_{c \in \sigma' \cap \mathbb{C}} (x = c)) \in \mathcal{A}_{\sigma'}.$$

It means, that every element from $Adom_{\sigma'}$ necessarily appears in at least one predicate from σ' or equal to some constant from σ', if not all the predicates from σ' are empty or σ' contains constants.

4. $Adom_{\sigma'}$ is not empty:

$$\exists x.\, Adom_{\sigma'}(x) \in \mathcal{A}_{\sigma'}.$$

The next proposition follows from the definition of $\mathcal{A}_{\sigma'}$.

Proposition 1. *An interpretation $\mathcal{M} = \langle \Delta^{\mathcal{M}}, \cdot^{\mathcal{M}} \rangle$ is a model of $\mathcal{A}_{\sigma'}$ iff*

- $Adom_{\sigma'}^{\mathcal{M}} = adom(\sigma', \mathcal{M})$, *if* $adom(\sigma', \mathcal{M}) \neq \emptyset$;
- $Adom_{\sigma'}^{\mathcal{M}} \neq \emptyset$, *if* $adom(\sigma', \mathcal{M}) = \emptyset$.

Let $\phi(\bar{x})$ be a formula in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ with free variables $\bar{x}$, $x \in \bar{x}$, and let P be some predicate. For the sake of readability we write $\exists x \in P.\, \phi(\bar{x})$ to denote the formula $\exists x.\, P(x) \wedge \phi(\bar{x})$; and we write $\forall x \in P.\, \phi(\bar{x})$ to denote the formula $\forall x.\, P(x) \rightarrow \phi(\bar{x})$.

Let now $\phi(\bar{x})$ be a query in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ in prenex normal form, that is

$$\phi(\bar{x}) = \mathcal{Q}_1 y_1 \ldots \mathcal{Q}_k y_k.\, \psi(\bar{y}, \bar{x}),$$

where $\mathcal{Q}_i$ stands for either $\forall$ or $\exists$, $\bar{x} = \{x_1, \ldots, x_n\}$ is the set of all free variables of $\phi(\bar{x})$, $\bar{y} = \{y_1, \ldots, y_k\}$ is the set of all bounded (quantified) variables of $\phi(\bar{x})$, and $\psi(\bar{y}, \bar{x})$ is a quantifier free formula. Let P be an unary predicate. We will use the following denotation:

$$\phi(\bar{x})|_P := (\mathcal{Q}_1 y_1 \in P \ldots \mathcal{Q}_k y_k \in P.\, \psi(\bar{y}, \bar{x})) \wedge \bigwedge_{i=1}^{n} P(x_i).$$

We extend this denotation also for the formulas, that are not in prenex normal form, meaning that the aforementioned operation $(\cdot)|_{(\cdot)}$ is applied to the prenex normal form of the formula.

Theorem 1 (Checking Domain Independence).
Let $\phi(\bar{x})$ be a formula in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$. Then $\phi(\bar{x})$ is domain independent iff

$$\mathcal{A}_{\sigma(\phi)} \models \forall \bar{x}\, .\, \phi(\bar{x}) \leftrightarrow \phi(\bar{x})|_{Adom_{\sigma(\phi)}}. \tag{1}$$

Example 1. Let us consider the formula $\phi = \exists x.\,(A(x) \vee B(a))$ from [5] mentioned in Section 1 and apply the theorem for it. This formula is not safe-range but it is domain independent, because it is logically equivalent to $(\exists x.\,A(x)) \vee B(a)$, which is safe-range and, hence, domain independent. Such kind of formulas are of particular interest for us.

Since $adom(\sigma(\phi), \mathcal{M}) = A^{\mathcal{M}} \cup B^{\mathcal{M}} \cup \{a^{\mathcal{M}}\}$, because of the proposition **??** we have:

$$\mathcal{A}_{\sigma(\phi)} \models \forall x.\, Adom_{\sigma(\phi)}(x) \leftrightarrow A(x) \vee B(x) \vee (x = a).$$

Then $\mathcal{A}_{\sigma(\phi)} \models \phi|_{Adom_{\sigma(\phi)}} \leftrightarrow \exists x.\,(A(x) \vee B(x) \vee (x = a)) \wedge (A(x) \vee B(a))$. One can see, that

$$\models \exists x.\,(A(x) \vee B(a)) \leftrightarrow \exists x.\,(A(x) \vee B(x) \vee (x = a)) \wedge (A(x) \vee B(a)).$$

That is, $\mathcal{A}_{\sigma(\phi)} \models \phi \leftrightarrow \phi|_{Adom_{\sigma(\phi)}}$. Therefore, by applying the theorem we proved, that formula ϕ is domain independent. □

4 Conclusion

We proposed a method that allows to reduce the problem of checking domain independence of a formula to checking standard first-order logic entailment: a formula $\phi(\bar{x})$ from $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ is domain independent whenever the entailment (1) holds. Using this technique in particular makes sense when the formula to be checked for domain independence is not safe-range. Thus, in practice this method is useful and can be applied in any not safe-range decidable fragment of $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$, where $\mathcal{A}_{\sigma(\phi)}$ and $\forall \bar{x}\,.\,\phi(\bar{x}) \leftrightarrow \phi(\bar{x})|_{Adom_{\sigma(\phi)}}$ can be expressed. Discovering of such interesting fragments and application of the technique there is our future work.

We would like to thank Martín Rezk and Nhung Ngo for ideas and fruitful discussions.

References

1. Avron, A.: Constructibility and decidability versus domain independence and absoluteness. Theor. Comput. Sci. **394** (April 2008) 144–158
2. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
3. Franconi, E., Kerhet, V.: On checking domain independence (extended version). Technical Report KRDB12-3, KRDB Research Centre, Free University of Bozen-Bolzano (2012) `http://www.inf.unibz.it/krdb/pub/TR/KRDB12-3.pdf`.
4. Kleene, S.C.: Mathematical Logic. New York: Dover (2002)
5. Topor, R.W.: Domain-independent formulas and databases. Theor. Comput. Sci. **52** (1987) 281–306