On Synthesizing Computable Skolem Functions for First Order Logic

Supratik Chakraborty
Department of Computer Science and Engineering, IIT Bombay, India

S. Akshay
Department of Computer Science and Engineering, IIT Bombay, India

Abstract

Skolem functions play a central role in the study of first order logic, both from theoretical and practical perspectives. While every Skolemized formula in first-order logic makes use of Skolem constants and/or functions, not all such Skolem constants and/or functions admit effectively computable interpretations. Indeed, the question of whether there exists an effectively computable interpretation of a Skolem function, and if so, how to automatically synthesize it, is fundamental to their use in several applications, such as planning, strategy synthesis, program synthesis etc.

In this paper, we investigate the computability of Skolem functions and their automated synthesis in the full generality of first order logic. We first show a strong negative result, that even under mild assumptions on the vocabulary, it is impossible to obtain computable interpretations of Skolem functions. We then show a positive result, providing a precise characterization of first-order theories that admit effective interpretations of Skolem functions, and also present algorithms to automatically synthesize such interpretations. We discuss applications of our characterization as well as complexity bounds for Skolem functions (interpreted as Turing machines).

2012 ACM Subject Classification Theory of computation → Logic and verification

Keywords and phrases Skolem functions, Automated, Synthesis, First order logic, Computability

Digital Object Identifier 10.4230/LIPIcs.MFCS.2022.30

Related Version *Full Version*: https://arxiv.org/abs/2102.07463

1 Introduction

The history of Skolem functions can be traced back to 1920, when the Norwegian mathematician, Thoralf Albert Skolem, gave a simplified proof of a landmark result in logic, now known as the *Löwenheim-Skolem* theorem. Skolem's proof made use of a key observation: *For every first order logic formula $\exists y\, \varphi(x, y)$, the choice of y that makes $\varphi(x, y)$ true (if at all) depends on x in general. This dependence can be thought of as implicitly defining a function that gives the "correct" value of y for every value of x. If F_y denotes a fresh unary function symbol, the second order sentence $\exists F_y\, \forall x \left(\exists y\, \varphi(x, y) \Rightarrow \varphi(x, F_y(x))\right)$ formalizes this idea.* Since the implication trivially holds in the other direction too, the second order sentence $\exists F_y\, \forall x \left(\exists y\, \varphi(x, y) \Leftrightarrow \varphi(x, F_y(x))\right)$ is valid.

Let $\xi_1 \equiv \exists y\, \varphi(x, y)$ and $\xi_2 \equiv \varphi(x, F_y(x))$. The fresh function symbol F_y introduced in transforming ξ_1 to ξ_2 is called a *Skolem function*. Skolem functions play an extremely important role in logic – both from theoretical and applied perspectives. While it suffices in some contexts to simply know that a Skolem function F_y exists, in other contexts, we require an effective procedure to compute $F_y(x)$ for every value of x. This motivates us to ask if Skolem functions are always computable, and whenever they are, can we algorithmically generate a halting Turing machine that computes the function? Note that we are concerned with computability at two levels here: (i) computability of the Skolem function itself, and (ii) computability of a halting Turing machine that computes the Skolem function. For clarity of

exposition, we call the Turing machine referred to in (ii) above a *computable interpretation of Skolem function*, and the problem of generating it algorithmically the *synthesis problem for Skolem functions*.

The synthesis problem for Skolem functions has been studied in detail in the propositional setting, specifically for quantified Boolean formulas (QBF) with a $\forall^*\exists^*$ quantifier prefix [17, 16, 12, 18, 13, 22, 27, 5, 23, 3, 1, 21, 4, 14, 24]. Surprisingly, a similar in-depth investigation in the context of general first order logic appears lacking in the literature, despite several potential applications, viz. automatic program synthesis and repair [26, 19, 28]. Some notable works in the context of specific theories include those of Kuncak et al [19] (for linear rational arithmetic), Spielman et al [25] (unbounded bit-vector theory), Preiner et al [20] (bit-vector theory) etc. in which terms that serve as interpretations of Skolem functions in specific theories are synthesized. In [17], Jiang presented a partial approach for quantifier elimination in general first-order theories by relying on the availability of functions that can be conditionally expressed by a finite set of terms. Unfortunately, such finite conditional decomposition may not always be possible (as acknowledged in [17]), even when a computable interpretation of the Skolem function exists. The problem of quantifier-free constraint solving, i.e. finding assignments of free variables that render a quantifier-free formula true, has been investigated in depth for several theories, viz. propositional logic, theory of arrays, linear rational arithmetic, real algebraic numbers, Presburger arithmetic, regular languages of finite strings etc. If the theory also admits effective quantifier elimination, this yields an algorithm for synthesizing computable interpretations of Skolem functions. However, not all first order theories admit effective quantifier elimination, e.g. Presburger arithmetic (without divisibility predicates) or the theory of evaluated trees [10] does not. We show that for some such theories, computable interpretations of Skolem functions can be synthesized algorithmically.

Our main contributions are to ask and answer the following questions:

- Does there always exist computable interpretations of Skolem functions for a first order formula interpreted over a structure? We answer this question strongly in the negative by showing that uncomputable interpretations cannot be avoided even with one binary predicate and one existential quantifier in the formula.
- We next ask if it is possible to algorithmically decide whether computable interpretations exist for all Skolem functions, given a formula and a structure over which it is interpreted. We answer this question in the negative.
- Next, we ask if it is possible to characterize the class of structures such that effectively computable interpretations of Skolem functions can be algorithmically synthesized for all formulas interpreted over a structure in the class. We answer this by showing that decidability of the elementary diagram of a structure serves as the required necessary and sufficient condition. Using this result, we show that several important first-order theories admit synthesis of effectively computable Skolem functions, while others do not.
- For structures satisfying the condition in the above characterization, we present lower and upper complexity bounds for effectively computable interpretations of Skolem functions.
- Finally, we distinguish between synthesizing Skolem functions as halting Turing machines vs terms in the underlying logic and show that the latter is a strictly weaker notion.

Our results reveal a highly a nuanced picture of the computability landscape for synthesizing interpretations of Skolem functions in first-order logic. We hope that this work will be a starting point towards further research into the design of practical algorithms (whenever possible) to synthesize Skolem functions for various first order theories. Proofs that are missing due to lack of space can be found in the full version at [2].

2 Preliminaries

Since every Turing machine with tape alphabet $\{0,1\}$ can be encoded as a natural number (we use $\mathbb{N}$ for naturals), and since every finite string over $\{0,1\}^*$ can be encoded as a natural number, we often speak of Turing machine i, denoted TM_i, running on input string j, where $i, j \in \mathbb{N}$.

We use x, y, z, etc., possibly with subscripts, to denote first order variables, $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$, etc., possibly with subscripts, to denote sequences of first order variables. We use φ, ξ, α, possibly with subscripts, to denote formulas. For a sequence $\mathbf{X}_i$, $|\mathbf{X}_i|$ denotes the count of variables in $\mathbf{X}_i$, and $x_{i,1}, \ldots x_{i,|\mathbf{X}_i|}$ denotes the variables. A *vocabulary* $\mathcal{V}$, is a set of function and/or predicate symbols, along with their respective arities. Constants are function symbols with arity 0. We assume that $\mathcal{V}$ *has finitely many predicate and function symbols, except possibly for countably infinitely many constant symbols.* We also assume that *a special binary predicate "=" (equality) is present in every vocabulary.*

We consider first order logic formulas over vocabulary $\mathcal{V}$, also called $\mathcal{V}$*-formulas*. The notion of *bound* and *free* variables is standard, $\mathcal{V}$-formulas without free variables are $\mathcal{V}$*-sentences*. A $\mathcal{V}$*-term* is either a variable or $f(t_1, \ldots t_k)$, where f is a k-ary function symbol in $\mathcal{V}$ and $t_1, \ldots t_k$ are $\mathcal{V}$-terms. When $\mathcal{V}$ is implicit from the context, we omit it. A *ground term* (resp. *ground formula*) is a term (resp. formula) without any variables. For x, a free variable in ξ, t a term in which all variables (if any) are free in ξ, $\xi[x \mapsto t]$ denotes the formula obtained by *substituting* t *for* x *in* ξ, i.e., replacing every free occurrence of x in ξ with t.

A $\mathcal{V}$*-structure* $\mathfrak{M}$ consists of a *universe* $U^{\mathfrak{M}}$ of elements and an interpretation of every predicate and function symbol in $\mathcal{V}$ over $U^{\mathfrak{M}}$. The interpretation of the special predicate "=" is always the identity relation, and we write $t_1 = t_2$ instead of $= (t_1, t_2)$ for notational convenience. We denote the interpretation of a predicate symbol P (resp. function symbol f) in $\mathfrak{M}$ as $P^{\mathfrak{M}}$ (resp. $f^{\mathfrak{M}}$). In general, an interpretation of a predicate or function symbol may be well-defined but not computable. We say a $\mathcal{V}$-structure $\mathfrak{M}$ is *computable* if $U^{\mathfrak{M}}$ is countable and if $P^{\mathfrak{M}}$ (resp. $f^{\mathfrak{M}}$) is computable for all predicate symbol P (resp. function symbol f) in $\mathcal{V}$. In other words, there exists a halting Turing machine for computing the interpretations $P^{\mathfrak{M}}$ (resp. $f^{\mathfrak{M}}$). *Throughout this paper, we assume that all* $\mathcal{V}$*-structures are computable.* This is motivated by practical applications of Skolem functions; additionally, non-computable $\mathcal{V}$-structures may make it difficult (even impossible) to obtain computable interpretations of Skolem functions in most cases. A computable $\mathcal{V}$ structure can be finitely represented, e.g. by using a single bit to encode whether the universe is finite or countably infinite, and by giving a natural number encoding of each Turing machine that computes an interpretation of a predicate or function symbol. If there are countably infinite constant symbols, we assume that interpretations of all of them can be collectively encoded by a single Turing machine that computes a mapping from $\mathbb{N}$ (index of constant symbol) to $\mathbb{N}$ (index of element in universe). If a $\mathcal{V}$-formula $\xi(\mathbf{Z})$ evaluates to true when interpreted over $\mathfrak{M}$ and with $\mathbf{Z}$ set to $\sigma \in \left(U^{\mathfrak{M}}\right)^{|\mathbf{Z}|}$, we say that $\mathfrak{M}$ is a *model* of $\xi(\sigma)$ and denote it by $\mathfrak{M} \models \xi(\sigma)$. An *expansion* of a vocabulary $\mathcal{V}$ is a vocabulary $\mathcal{V}'$ such that $\mathcal{V} \subseteq \mathcal{V}'$. Given a $\mathcal{V}$-structure $\mathfrak{M}$ and a $\mathcal{V}'$-structure $\mathfrak{M}'$, where $\mathcal{V}'$ is an expansion of $\mathcal{V}$, $\mathfrak{M}'$ is an *expansion* of $\mathfrak{M}$ if (i) $U^{\mathfrak{M}'} = U^{\mathfrak{M}}$, and (ii) all predicate/function symbols in $\mathcal{V}$ are interpreted identically in $\mathfrak{M}$ and $\mathfrak{M}'$.

For a quantifier $Q \in \{\exists, \forall\}$ and sequence of variables $\mathbf{X}_i = (x_{i,1}, \ldots x_{i,|\mathbf{X}_i|})$, we use $Q\mathbf{X}_i$ to denote the block of quantifiers $Qx_{i,1} \ldots Qx_{i,|\mathbf{X}_i|}$. Every first order logic formula can be effectively transformed to a semantically equivalent *prenex normal form*, in which all quantifiers appear to the left of the quantifier-free part of the formula. Henceforth,

we assume all first order formulas are in prenex normal form, unless stated otherwise. Let $\xi(\mathbf{Z}) \equiv \forall \mathbf{X}_1 \exists \mathbf{Y}_1 \cdots \forall \mathbf{X}_q \exists \mathbf{Y}_q \, \varphi(\mathbf{Z}, \mathbf{X}_1, \mathbf{Y}_1, \ldots \mathbf{X}_q, \mathbf{Y}_q)$ be such a formula, where $\mathbf{Z}$ is a sequence of free variables, and φ is quantifier-free. We say that $\forall \mathbf{X}_1 \exists \mathbf{Y}_1 \cdots \forall \mathbf{X}_q \exists \mathbf{Y}_q$ is the *quantifier prefix* of the formula, and it has q $\forall^* \exists^*$ blocks. The quantifier-free part, i.e. φ, is called the *matrix* of the formula. Note that in case the leading (leftmost) quantifier in ξ is existential, $\mathbf{X}_1$ may be considered to be an empty sequence, and similarly, if the trailing (rightmost) quantifier in ξ is universal. Every variable $y_{i,j}$ that is existentially quantified in the quantifier prefix is called an an *existential variable* in ξ. The notion of *universal variables* is analogously defined. The quantifier prefix imposes a total order on the quantified variables in ξ. We say that a variable u is *to the left* (resp. *right*) of variable v in the quantifier prefix iff Qu appears to the left (resp. right) of $Q'v$ in the quantifier prefix, where $Q, Q' \in \{\exists, \forall\}$.

Skolemization. Given a formula ξ in prenex normal form, *Skolemization* refers to the process of transforming ξ to a new formula $\xi^\star$ via the following steps: (i) for every existential variable $y_{i,j}$, substitute $F_{y_{i,j}}(\mathbf{Z}, \mathbf{X}_1, \ldots \mathbf{X}_i)$ for $y_{i,j}$ in φ, where $F_{y_{i,j}}$ is a new function symbol of arity $|\mathbf{Z}| + \sum_{j=1}^{i} |\mathbf{X}_j|$, and (ii) remove all existential quantifiers from the quantifier prefix of ξ. The functions $F_{y_{i,j}}$ introduced above are called *Skolem functions*. In case ξ has no free variables and the leading quantifier is existential, the Skolem functions for variables in the leftmost existential quantifier block have no arguments (i.e. they are nullary functions). Such functions are also called *Skolem constants*. The sentence $\xi^\star$ is said to be in *Skolem normal form* if the matrix of $\xi^\star$ is in conjunctive normal form. The key guarantee of Skolemization is as follows: for every existential variable $y_{i,j}$, let $\xi^\star_{y_{i,j}}$ denote the formula obtained by Skolemizing all existential variables to the left of $y_{i,j}$ in the quantifier prefix. Formally, $\xi^\star_{y_{i,j}}$ is obtained by (i) substituting the Skolem function $F_{y_{k,l}}$ for every existential variable $y_{k,l}$ to the left of $y_{i,j}$ in the quantifier prefix, and (ii) removing all quantifiers to the left of and including $\exists y_{i,j}$ from the quantifier prefix. Note that $\xi^\star_{y_{i,j}}$ has free variables in $\mathbf{Z}, \mathbf{X}_1, \ldots \mathbf{X}_i, y_{i,j}$. Skolemization guarantees that for every $\mathcal{V}$-structure $\mathfrak{M}$ over which ξ is interpreted, there always exists an expansion $\mathfrak{M}^\star$ of $\mathfrak{M}$ that provides an interpretation of $F_{y_{i,j}}$ for all existential variables $y_{i,j}$ such that the following holds for every $i \in \{1, \ldots q\}$ and $j \in \{1, \ldots |\mathbf{Y}_i|\}$:

$$\forall \mathbf{Z} \forall \mathbf{X}_1 \ldots \forall \mathbf{X}_i \left(\exists y_{i,j} \, \xi^\star_{y_{i,j}} \;\Leftrightarrow\; \xi^\star_{y_{i,j}}[y_{i,j} \mapsto F_{y_{i,j}}(\mathbf{Z}, \mathbf{X}_1, \ldots \mathbf{X}_i)] \right) \tag{1}$$

▶ **Example 1.** Consider $\xi(z) \equiv \exists y \forall x \exists u \forall v \exists w \, \varphi(z, x, y, u, v, w)$. Skolemizing gives $\xi^\star \equiv \forall x \forall v \, \varphi(z, x, F_y(z), F_u(z, x), v, F_w(z, x, v))$, where $F_y(z), F_u(z, x)$ and $F_w(z, x, v)$ are Skolem functions for y, u and w respectively. Using the notation introduced above, we have

- $\xi^\star_y(z, y) \;\equiv\; \forall x \exists u \forall v \exists w \, \varphi(z, x, y, u, v, w)$
- $\xi^\star_u(z, x, u) \;\equiv\; \forall v \exists w \, \varphi(z, x, F_y(z), u, v, w)$
- $\xi^\star_w(z, x, v, w) \;\equiv\; \varphi(z, x, F_y(z), F_u(z, x), v, w)$

By virtue of Skolemization, we know that for every structure $\mathfrak{M}$ over which ξ is interpreted, there exists an expansion $\mathfrak{M}^\star$ that interprets F_y, F_u and F_w such that the following hold.

- $\forall z \big(\exists y \, \xi^\star_y(z, y) \Leftrightarrow \xi^\star_y[y \mapsto F_y(z)] \big)$
- $\forall z \forall x \left(\exists u \, \xi^\star_u(z, x, u) \Leftrightarrow \xi^\star_u[u \mapsto F_u(z, x)] \right)$
- $\forall z \forall x \forall v \left(\exists w \xi^\star_w(z, x, v, w) \Leftrightarrow \xi^\star_w[w \mapsto F_w(z, x, v)] \right)$

Let $\mathcal{V}^\star$ be the expansion of $\mathcal{V}$ obtained by adding all Skolem function and constant symbols in $\xi^\star$ to $\mathcal{V}$. In general, a $\mathcal{V}$-structure $\mathfrak{M}$ over which $\xi(\mathbf{Z})$ is interpreted can be expanded to a $\mathcal{V}^\star$-structure by adding interpretations of Skolem functions for all existential variables in $\xi(\mathbf{Z})$. However, not every such expansion of $\mathfrak{M}$ may model the sentence (1) above for every existential variable $y_{i,j}$. Skolemization guarantees that there exists at least

one "correct" expansion $\mathfrak{M}^\star$ of $\mathfrak{M}$ that does so. We call the interpretation of Skolem functions in such a "correct" expansion as an $\mathfrak{M}$-*interpretation* of the Skolem functions. There may be multiple "correct" expansions of $\mathfrak{M}$, and hence multiple $\mathfrak{M}$-interpretations of Skolem functions. Skolemization guarantees the existence of at least one $\mathfrak{M}$-interpretation of all Skolem functions/constants; however, it doesn't tell us whether these are computable interpretations, and if so, can we algorithmically synthesize the interpretation as a halting Turing machine? These are two central questions that concern us in this paper.

Sometimes, given a $\mathcal{V}$-formulas ξ, we can find an $\mathfrak{M}$-interpretation of Skolem functions that works in the same way for all computable structures $\mathfrak{M}$ over which ξ is interpreted (modulo differences in interpreting predicates and functions). Formally, suppose there exists a halting Turing machine M_ξ with access to oracles that compute the interpretations of predicates and functions in $\mathfrak{M}$, and suppose M_ξ computes an $\mathfrak{M}$-interpretation of Skolem functions for all existential variables in ξ, and for all $\mathcal{V}$-structures $\mathfrak{M}$. Then, we say that ξ admits a *uniform representation* of $\mathfrak{M}$-interpretations of Skolem functions.

Model theory. We use $\mathcal{V}(\mathfrak{M})$ to denote the expansion of $\mathcal{V}$ obtained by adding a fresh constant symbol c_e for every element $e \in U^{\mathfrak{M}}$, if not already present in $\mathcal{V}$. Clearly, if $U^{\mathfrak{M}}$ and $\mathcal{V}$ are countable, so is $\mathcal{V}(\mathfrak{M})$. We use $\mathfrak{M}_C$ to denote the expansion of $\mathfrak{M}$ to a $\mathcal{V}(\mathfrak{M})$-structure that interprets the additional constants in $\mathcal{V}(\mathfrak{M})$ in the natural way, i.e. c_e is interpreted to have the value e, for all $e \in U^{\mathfrak{M}}$. The *elementary diagram* of $\mathfrak{M}$, denoted $\mathcal{ED}(\mathfrak{M})$, is the set of all $\mathcal{V}(\mathfrak{M})$-sentences ξ such that $\mathfrak{M}_C \models \xi$. The *diagram* of $\mathfrak{M}$, denoted $\mathcal{D}(\mathfrak{M})$, is the set of all literals in $\mathcal{ED}(\mathfrak{M})$, i.e. the set of all atomic ground formulas that hold in $\mathfrak{M}_C$. Clearly, $\mathcal{D}(\mathfrak{M}) \subseteq \mathcal{ED}(\mathfrak{M})$. A set Γ of $\mathcal{V}$-sentences is called a $\mathcal{V}$-*theory* if it is consistent, i.e. there exists a $\mathcal{V}$-structure that serves as a model for every sentence in Γ. Given a $\mathcal{V}$-structure $\mathfrak{M}$, the set of all first order $\mathcal{V}$-sentences ξ such that $\mathfrak{M} \models \xi$ is called the *theory of* $\mathfrak{M}$, denoted $Th(\mathfrak{M})$. Note that both $\mathcal{ED}(\mathfrak{M})$ and $\mathcal{D}(\mathfrak{M})$ are $\mathcal{V}$-theories, and $\mathcal{ED}(\mathfrak{M}) = Th(\mathfrak{M}_C)$, where $Th(\mathfrak{M}_C)$ is the $\mathcal{V}(\mathfrak{M})$-theory of $\mathfrak{M}_C$. We say that a $\mathcal{V}$-theory Γ is *decidable* iff there exists a Turing machine that takes as input an arbitrary $\mathcal{V}$-sentence ξ and always halts and correctly reports whether $\xi \in \Gamma$ or not. If $\mathfrak{M}$ is a computable structure, it follows immediately that $\mathcal{D}(\mathfrak{M})$ is a decidable theory, but $\mathcal{ED}(\mathfrak{M})$ is not necessarily so.

A $\mathcal{V}$-theory Γ is said to admit *quantifier elimination* if for every $\mathcal{V}$-formula $\xi(\mathbf{Z})$ with free variables $\mathbf{Z}$, there exists a semantically equivalent quantifier-free $\mathcal{V}$-formula $\xi^{\#}(\mathbf{Z})$ such that the sentence $\forall \mathbf{Z}\left(\xi(\mathbf{Z}) \Leftrightarrow \xi^{\#}(\mathbf{Z})\right)$ is in Γ. If, in addition, there exists a Turing machine that takes an arbitrary $\mathcal{V}$-formula (ξ) as input and computes its quantifier-eliminated form $(\xi^{\#})$ and halts, we say that Γ admits *effective quantifier elimination*[1]. For a $\mathcal{V}$-structure $\mathfrak{M}$, we say that $Th(\mathfrak{M})$ admits *effective constraint solving* if there exists a Turing machine that takes a $\mathcal{V}$-formula $\xi(\mathbf{Z})$ with free variables $\mathbf{Z}$ as input and halts after reporting one of two things: (i) a $|\mathbf{Z}|$-tuple σ of elements from $U^{\mathfrak{M}}$ such that $\mathfrak{M} \models \xi(\sigma)$, or (ii) no such $|\mathbf{Z}|$-tuple of elements from $U^{\mathfrak{M}}$ exists. Note that the formula $\xi(\mathbf{Z})$ may have quantifiers in general. In case the above Turing machine exists only if $\xi(\mathbf{Z})$ is quantifier-free, we say that $Th(\mathfrak{M})$ admits *effective quantifier-free constraint solving*. Clearly, if $Th(M)$ admits effective quantifier elimination and effective quantifier-free constraint solving, then it also admits effective constraint solving.

[1] There is a technique, popularly called "Morleyization", that trivially makes a theory admit effective quantifier elimination by expanding the vocabulary to include a separate predicate symbol for each $\mathcal{V}$-formula. For purposes of this paper, we disallow expansion of the vocabulary (and hence "Morleyization") during effective quantifier elimination.

3 An illustrative example

Consider the vocabulary $\mathcal{V} = \{P, c, d\}$, where P is a binary predicate symbol, and c and d are constants, and the first-order $\mathcal{V}$-sentence $\xi \equiv \forall x \exists y P(x, y) \wedge \big(P(x, c) \vee P(x, d)\big)$. We will use $\varphi(x, y)$ to denote the matrix of the above formula, i.e. $P(x, y) \wedge \big(P(x, c) \vee P(x, d)\big)$. On Skolemizing ξ we get $\xi^\star \equiv \forall x \varphi(x, F_y(x))$, where F_y is a fresh unary Skolem function symbol. Let $\mathfrak{M}$ be a computable $\mathcal{V}$-structure. We now ask if there exists an algorithm $\mathcal{A}^{[F]}$ that serves as a computable interpretation of $F_y : U^{\mathfrak{M}} \to U^{\mathfrak{M}}$. A careful examination of ξ and $\xi^\star$ reveals that such an algorithm indeed exists. Specifically, the algorithm (represented informally as an imperative "program" for ease of understanding) "**input(x); if** $\mathtt{P}^{\mathfrak{M}}(\mathtt{x}, \mathtt{c}^{\mathfrak{M}})$ **then return** $c^{\mathfrak{M}}$ **else return** $d^{\mathfrak{M}}$" takes as input $x \in U^{\mathfrak{M}}$ and returns either $c^{\mathfrak{M}}$ or $d^{\mathfrak{M}}$ depending on whether $P^{\mathfrak{M}}(x, c^{\mathfrak{M}})$ evaluates to true or false. If we let this algorithm interpret F_y in the expansion $\mathfrak{M}^\star$ of $\mathfrak{M}$, then it is not hard to see that we indeed have $\mathfrak{M}^\star \models \forall x \big(\exists y \varphi(x, y) \Leftrightarrow \varphi(x, F_y(x))\big)$.

However, is this always possible? Consider the $\mathcal{V}$-formula $\alpha \equiv \forall x \exists y\, P(x, y)$ instead of ξ, whose Skolemized version is $\alpha^\star \equiv \forall x\, P(x, F_y(x))$. As we show in Section 5, it is impossible to obtain a computable $\mathfrak{M}$-interpretation of the Skolem function $F_y(x)$ in this case for all $\mathcal{V}$-structures $\mathfrak{M}$.

There are several observations that one can now make. Clearly, algorithm $\mathcal{A}^{[F]}$ described above is specific to the formula ξ; a different formula would have required a different algorithm to be designed for its Skolem function(s). Interestingly, algorithm $\mathcal{A}^{[F]}$ also requires access to the interpretations of c, d and P in the $\mathcal{V}$-structure $\mathfrak{M}$ on which ξ is interpreted. Since we are given an effectively computable interpretation of P in $\mathfrak{M}$, there exists an algorithm $\mathcal{A}^{[P]}$ to compute $P^{\mathfrak{M}}$. Algorithm $\mathcal{A}^{[F]}$ effectively uses $\mathcal{A}^{[P]}$ as a sub-routine to compute the value of $F_y(x)$ for every $x \in U^{\mathfrak{M}}$. Note that if the interpretation of P (in perhaps a different $\mathcal{V}$-structure $\mathfrak{M}'$) was not effectively computable, the "program" above would not serve as an effectively computable interpretation of F_y. This underlines the importance of effectively computable structures in the synthesis of Skolem functions.

It is easy to see that "**input(x); if** $\mathtt{P}^{\mathfrak{M}}(\mathtt{x}, \mathtt{c}^{\mathfrak{M}})$ **then return** $c^{\mathfrak{M}}$ **else return** $d^{\mathfrak{M}}$" *uniformly* serves as a computable interpretation of F_y in every computable $\mathcal{V}$-structure $\mathfrak{M}$ over which ξ is interpreted. Regardless of the actual structure $\mathfrak{M}$, a computable $\mathfrak{M}$-interpretation of F_y is obtained by invoking algorithms to compute interpretations of P, c, d in $\mathfrak{M}$ as sub-routines. Thus we get a uniform representation of an $\mathfrak{M}$-interpretation of F_y.

Finally, the interpretation of Skolem function F discussed above is represented as an algorithm, and not as a $\mathcal{V}$-term. Is it possible to obtain a $\mathcal{V}$-term that uniformly represents an $\mathfrak{M}$-interpretation of F_y in this case? To answer this, first observe that there are only two terms, viz. c and d, that can be formed using $\mathcal{V}$. If one of these terms serves as a uniform $\mathfrak{M}$-interpretation of F_y, choose a structure $\mathfrak{M}$ as follows: $U^{\mathfrak{M}} = \{a_0, a_1\}, c^{\mathfrak{M}} = a_0, d^{\mathfrak{M}} = a_1, P^{\mathfrak{M}}(a_0, a_0) = P^{\mathfrak{M}}(a_1, a_1) = \mathsf{false}$ and $P^{\mathfrak{M}}(a_0, a_1) = P^{\mathfrak{M}}(a_1, a_0) = \mathsf{true}$. Clearly $\mathfrak{M} \models \forall x \exists y \varphi(x, y)$. However, with $F_y(x) = c$ (or $F_y(x) = d$), we have $MM^\star \not\models \forall x \big(\exists y \varphi(x, y) \;\Leftrightarrow\; \varphi(x, F(x))\big)$. Thus even when an effectively computable interpretation of a Skolem function exists, it may not be representable as a term over $\mathcal{V}$.

4 Problem statement

We now formulate the primary questions that we wish to address in this paper.

1. Given a vocabulary $\mathcal{V}$, a $\mathcal{V}$-formula $\xi(\mathbf{Z})$ in prenex normal form and a computable $\mathcal{V}$-structure $\mathfrak{M}$, the SkolemExist problem asks if there exists a *computable* $\mathfrak{M}$-interpretation of Skolem functions for all existential variables in ξ. We have already seen in Section 3 that there are positive instances of SkolemExist. We ask if there are negative instances as well, i.e. there is no computable $\mathfrak{M}$-interpretation of Skolem functions.

2. Next, we ask if SkolemExist is decidable.
3. We then consider special cases where either the formula $\xi(\mathbf{Z})$ or structure $\mathfrak{M}$ is fixed, and ask if it is possible to characterize the class of problems where the SkolemExist problem has a positive answer.
4. In cases where the SkolemExist problem has a positive answer, we ask the following:
   a. Does there exist an algorithm to synthesize computable $\mathfrak{M}$-interpretations of Skolem functions? We call this problem SkolemSynthesis and consider two variants of it, where either (i) $\mathcal{V}$ and $\xi(\mathbf{Z})$ are fixed and $\mathfrak{M}$ is the input of SkolemSynthesis, or (ii) $\mathcal{V}$ and $\mathfrak{M}$ is fixed and ξ is the input of SkolemSynthesis.
   b. Is it possible to obtain finite uniform representations of $\mathfrak{M}$-interpretations of Skolem functions, and if so, can we obtain these as $\mathcal{V}$-terms?
   c. In case SkolemExist has a positive answer, can we give bounds on the worst-case running time of computable $\mathfrak{M}$-interpretations of Skolem functions?

Note that SkolemSynthesis is not meaningful in cases where SkolemExist has a negative answer. Hence, we don't try to answer SkolemSynthesis in negative instances of SkolemExist. Moreover, all the above problems except the last one is trivial if the universe $U^{\mathfrak{M}}$ is finite. Therefore, we focus mostly on structures with countably infinite universe.

5 Hardness of SkolemExist and SkolemSynthesis

We have already seen a positive instance (i.e. problem instance with positive answer) of SkolemExist in Section 3. The following lemma shows that SkolemExist always has a positive answer if all Skolem functions are Skolem constants. In the following, we use $(\mathcal{V}, \mathfrak{M}, \xi)$ to denote an instance of SkolemExist, where $\mathcal{V}$ is a vocabulary, $\mathfrak{M}$ is a computable $\mathcal{V}$-structure and ξ is a $\mathcal{V}$-formula.

▶ **Lemma 2.** *For every vocabulary $\mathcal{V}$, every computable $\mathcal{V}$-structure $\mathfrak{M}$ and every $\mathcal{V}$-sentence $\exists \mathbf{Y}\, \varphi(\mathbf{Y})$, where φ is a quantifier-free $\mathcal{V}$-formula with free variables in $\mathbf{Y}$, the instance $(\mathcal{V}, \mathfrak{M}, \xi)$ of SkolemExist has a positive answer.*

However, there are negative instances of SkolemExist, even with a restricted vocabulary.

▶ **Theorem 3.** *There exists a negative instance of SkolemExist where the vocabulary has a single binary predicate.*

Now that we know there are positive and negative instances of SkolemExist, we ask if SkolemExist is decidable. Unfortunately, we obtain a negative answer in general.

▶ **Theorem 4.** *SkolemExist is undecidable.*

Proof. We prove this theorem by contradiction. Suppose, if possible, there exists a halting Turing machine M that takes as inputs a vocabulary $\mathcal{V}$, a $\mathcal{V}$-formula $\xi(\mathbf{Z})$ and a computable $\mathcal{V}$-structure $\mathfrak{M}$, and decides if there exists a computable $\mathfrak{M}$-interpretation of Skolem functions for all existential variables in $\xi(\mathbf{Z})$. We show below that we can use M to effectively decide if an arbitrary Turing machine, say TM_i, halts on the empty tape.

Consider $\mathcal{V} = \{Q, a\}$, where Q is a binary predicate symbol and a is a constant symbol. For each $i \in \mathbb{N}$, define $\mathfrak{M}_i$ to be a $\mathcal{V}$-structure such that $\mathbf{U}^{\mathfrak{M}_i} = \mathbb{N}$, $a^{\mathfrak{M}_i} = i$ and $Q^{\mathfrak{M}_i}(u, v) = \mathsf{true}$ for $u, v \in \mathbb{N}$ iff the Turing machine TM_u halts on the empty tape within v steps. It is easy to see that each $\mathfrak{M}_j$ is a computable $\mathcal{V}$-structure. We also define the $\mathcal{V}$-sentence $\xi \equiv \exists s \forall u \exists x \left(\big(Q(a, s) \wedge (x = u)\big) \vee \big(\neg Q(a, u) \wedge Q(u, x)\big)\right)$. Skolemizing this formula gives $\xi^\star \equiv \forall u \left(\big(Q(a, c_s) \wedge (F_x(u) = u)\big) \vee \big(\neg Q(a, u) \wedge Q(u, F_x(u))\big)\right)$, where c_s is a Skolem constant for s, and F_x is a Skolem function for x. From the guarantee of Skolemization, the following must hold:

- $\exists s \forall u \exists x \big(\big(Q(a,s) \wedge (x = u)\big) \vee \big(\neg Q(a,u) \wedge Q(u,x)\big)\big) \;\Leftrightarrow\; \forall u \exists x \big(\big(Q(a,c_s) \wedge (x = u)\big) \vee \big(\neg Q(a,u) \wedge Q(u,x)\big)\big)$
- $\forall u \big(\exists x \big(\big(Q(a,c_s) \wedge (x = u)\big) \vee \big(\neg Q(a,u) \wedge Q(u,x)\big)\big) \Leftrightarrow \big(\big(Q(a,c_s) \wedge (F_x(u) = u)\big) \vee \big(\neg Q(a,u) \wedge Q(u,F_x(u))\big)\big)\big)$

We now consider two cases.

- Suppose TM_i halts on the empty tape after $p \in \mathbb{N}$ steps. Then $\mathfrak{M}_i \models Q(a,p)$. In this case, by choosing the $c_s^{\mathfrak{M}_i} = p$ and by choosing $F_x^{\mathfrak{M}_i}(u) = u$ for all $u \in \mathbb{N}$, both the above guarantees of Skolemization are easily seen to hold. Clearly, the Skolem functions have computable interpretations in this case.
- If TM_i doesn't halt on the empty tape, then $\mathfrak{M}_i \models \forall u \, \neg Q(a,u)$. In this case, we choose an arbitary value, say 0, for s. However, for the guarantee of Skolemization to hold, we must have the following: for every $u \in \mathbb{N}$, if $\exists x Q(u,x)$ holds (i.e. TM_u halts on the empty tape), then $Q(u, F_x^{\mathfrak{M}_i}(u))$ must also hold (i.e. TM_u must also halt in $F_x^{\mathfrak{M}_i}(u)$ steps). Clearly, such an interpretation $F_x^{\mathfrak{M}_i}$ is not computable, as otherwise it can be used to decide the halting problem.

The above reasoning shows that there exist computable $\mathfrak{M}_i$-interpretations of all Skolem functions of existential variables in ξ iff TM_i halts on empty tape. Thus, if we feed the instance $(\mathcal{V}, \mathfrak{M}_i, \xi)$ as input to the supposed Turing machine M that decides SkolemExist, we can decide if TM_i halts on the empty tape, for every $i \in \mathbb{N}$. This gives a decision procedure for the halting problem on the empty tape – an impossibility! ◀

It is easy to see that the proof of Theorem 4 can be repeated with $\xi \equiv \forall u \exists s \exists x \big(\big(Q(a,s) \wedge (x = u)\big) \vee \big(\neg Q(a,u) \wedge Q(u,x)\big)\big)$ as well. This gives the following interesting result.

▶ **Theorem 5.** *If the vocabulary contains a binary predicate and a constant,* SkolemExist *is undecidable for the quantifier prefix classes* $\exists\forall\exists$ *and* $\forall\exists\exists$*. However it is decidable for the class* $\exists^+\forall^*$*.*

The second part of the above Theorem follows from an easy generalization of the proof of Lemma 2. This leaves only the case of $\forall\exists$ quantifier prefix, for which the decidability of SkolemExist remains open. We consider the case of the vocabulary having only monadic predicates later in Theorem 7.

The above negative results motivate us to consider special cases of SkolemExist and SkolemSynthesis, where either the $\mathcal{V}$-formula $\xi(\mathbf{Z})$ or the $\mathcal{V}$-structure $\mathfrak{M}$ is fixed.

Fixing the formula. The proof of Theorem 4 is quite damning: even if we allow the possibility of a potentially different algorithm, say $A_{\mathcal{V},\xi}$, for deciding SkolemExist for each combination of $\mathcal{V}$ and ξ, we cannot hope to have an algorithm $A_{\mathcal{V},\xi}$ for every $(\mathcal{V},\xi)$ pair. This is because in the proof of Theorem 4, we had indeed kept the vocabulary and formula fixed. This leaves only a few questions to be investigated if we fix the vocabulary and formula. If we consider $\mathcal{V}$ and ξ as fixed, the $\mathcal{V}$-structure $\mathfrak{M}$ is the only input to our problems of interest. The following theorem shows that SkolemSynthesis cannot be answered positively in this case even under fairly strong conditions.

Recall from Lemma 2 that SkolemExist has a positive answer if all Skolem functions are Skolem constants. Hence, by choosing ξ to be a $\mathcal{V}$-sentence with only existential quantifiers, we are guaranteed that all problem instances are positive instances of SkolemExist.

▶ **Theorem 6.** *There exists a vocabulary* $\mathcal{V}$*, a* $\mathcal{V}$*-sentence* ξ *and a family of* $\mathcal{V}$*-structures* $\mathcal{F} = \{\mathfrak{M}_i \mid i \in \mathbb{N}\}$*, such that* $(\mathcal{V}, \mathfrak{M}_i, \xi)$ *is a positive instance of* SkolemExist *for all* $i \in \mathbb{N}$*, yet there is no uniform representation of* $\mathfrak{M}_i$*-interpretations of the Skolem constants. Additionally, the* SkolemSynthesis *problem has a negative answer for the class of problem instances* $\{(\mathcal{V}, \xi, \mathfrak{M}_i) \mid i \in \mathbb{N}\}$*.*

It is interesting to ask now if there is a characterization of $\mathcal{V}$-formulas, such that for each $\mathcal{V}$-formula satisfying this characterization, the SkolemExist and SkolemSynthesis problems have positive answers for all $\mathcal{V}$-structures. The proof of Theorem 3 tells us that we must disallow binary predicates and $\forall\exists$ blocks in the quantifier prefix, which severely restricts the vocabulary and formulas. What happens if we allow a relational vocabulary with only monadic predicates (Löwenheim class with equality) [9]?

▶ **Theorem 7.** *Let the vocabulary $\mathcal{V}$ contain only monadic predicates and equality. Then SkolemExist has a positive answer, but not so for SkolemSynthesis.*

Proof. With k monadic predicates, the universe can be partitioned into 2^k equivalence classes based on predicate valuations. By an argument (based on Ehrenfeucht-Fraisse games) similar to that used to prove small-model property of Löwenheim class (see [9]), if a prenex formula ξ has quantifier rank r, the range of each Skolem function can be restricted to $\leq r.2^k$ elements. Using an argument similar to that in proof of Lemma 2, there exists a TM that enumerates the required set, say S, of $\leq r.2^k$ elements. Since elements of an equivalence class can only be distinguished using $=$, for each Skolem function of arity p, we must search for its correct interpretation over all $S^p \to S$ mappings. Since there are finitely many such mappings, we can enumerate the TMs computing these mappings, and one of them must effectively serve as the correct interpretation for the Skolem function under consideration. Since ξ has finitely many existential variables, it follows that there exists computable interpretations of all Skolem functions in ξ.

To see why SkolemSynthesis has a negative answer in general even with one monadic predicate P and one existential quantifier, consider $\mathcal{V} = \{P\}$, $\xi \equiv \exists x P(x)$, and a structure $\mathfrak{M}_i$ having universe $\mathbb{N}$ and $P^{\mathfrak{M}_i}(x) =$ true iff TM_i halts on empty tape within x steps. If there exists an algorithm to synthesize computable M_i-interpretations of the Skolem constant for x in ξ, we can use it to decide if TM_i halts on empty tape – an impossibility! Thus, we must disallow even monadic predicates if we want to characterize V-formulas that admit positive answer to SkolemSynthesis for all $\mathcal{V}$-structures. ◀

Fixing the structure. We now fix the structure $\mathfrak{M}$ (and vocabulary $\mathcal{V}$) and take the formula ξ as the only input of our problems of interest. Since the structure $\mathfrak{M}$ is fixed, we use the notation $\mathcal{U}$ for $U^{\mathfrak{M}}$ henceforth. Theorem 3 already shows that even when the structure is fixed, the SkolemExist problem has a negative instance. However, the $\mathcal{V}$-structure used in that proof may appear hand-crafted. This leads us to ask if there is a "natural" vocabulary $\mathcal{V}$ and $\mathcal{V}$-structure $\mathfrak{M}$, such that SkolemExist has a negative instance when considering $\mathcal{V}$-formulas. It turns out that this is indeed the case, and we show it by appealing to the classical Matiyasevich-Robinson-Davis-Putnam (MRDP) theorem [11].

▶ **Proposition 8.** *Skolem functions for the first order theory of natural numbers over the vocabulary $\{\times, +, 0, 1\}$ do not admit computable interpretations.*

Finally, in the setting of a fixed $\mathcal{V}$-structure $\mathfrak{M}$, even if SkolemExist is answered in the positive for all $\mathcal{V}$-formulas in a class Ξ, the SkolemSynthesis problem may have a negative answer for the class of problem instances $\{(\mathcal{V}, \mathfrak{M}, \xi) \mid \xi \in \Xi\}$.

▶ **Theorem 9.** *There exists a vocabulary $\mathcal{V}$, a $\mathcal{V}$-structure $\mathfrak{M}$ and a class of $\mathcal{V}$-sentences $\Xi = \{\xi_i \mid i \in \mathbb{N}\}$ s.t., $(\mathcal{V}, \mathfrak{M}, \xi)$ is a positive instance of SkolemExist for all $\xi_i \in \Xi$, yet SkolemSynthesis has a negative answer for the class of instances $\{(\mathcal{V}, \mathfrak{M}, \xi_i) \mid i \in \mathbb{N}\}$.*

6 Necessary & sufficient condition for synthesizing Skolem functions

Given these strong negative results is there hope for proving existence and synthesizability of computable interpretations for Skolem functions. Indeed, there do exist many natural theories where computable interpretations of Skolem functions exist and can indeed be synthesized, e.g., Boolean case, Presburger arithmetic etc. So, what determines when a $\mathcal{V}$-theory admits effective synthesis of computable interpretations of Skolem functions for all $\mathcal{V}$-formulas? Our first positive result is a surprising characterization of a *necessary and sufficient* condition for algorithmic synthesis of computable interpretations of Skolem functions.

▶ **Theorem 10.** *Let $\mathfrak{M}$ be a computable $\mathcal{V}$-structure for vocabulary $\mathcal{V}$. The* SkolemSynthesis *problem for $\mathcal{V}$-formulas, i.e. for problem instances $\{(\mathcal{V}, \mathfrak{M}, \xi) \mid \xi$ is a $\mathcal{V}$-formula$\}$, has a positive answer iff $\mathcal{ED}(\mathfrak{M})$ is decidable.*

Proof. ($\Longleftarrow$) Let $\xi(\mathbf{Z})$ be a $\mathcal{V}$-formula with free variables $\mathbf{Z}$, where $\xi(\mathbf{Z}) \equiv \forall\mathbf{X}_1\exists\mathbf{Y}_1 \ldots \forall\mathbf{X}_n\exists\mathbf{Y}_n$
$\xi_n(\mathbf{Z}, \mathbf{X}_1, \mathbf{Y}_1, \ldots, \mathbf{X}_n, \mathbf{Y}_n)$, where $\mathbf{X}_1, \ldots, \mathbf{X}_n$ are n sequences of universally quantified variables, $\mathbf{Y}_1, \ldots \mathbf{Y}_n$ are sequences of existentially quantified variables, $\mathbf{Z}$ is a sequence of free variables and ξ_n is quantifier-free. We will show that there is an algorithm, that for every $i \in \{1, \ldots, n\}$, takes as input a $(|\mathbf{Z}|+|\mathbf{X}_1|+\ldots+|\mathbf{X}_i|)$ tuple of values from the universe $\mathcal{U}$, say, $\mu \in \mathcal{U}^{|\mathbf{Z}|}, \sigma_1 \in \mathcal{U}^{|\mathbf{X}_1|}, \ldots, \sigma_i \in \mathcal{U}^{|\mathbf{X}_i|}$ and halts after computing a $(|\mathbf{Y}_1|+\ldots+|\mathbf{Y}_i|)$-dimensional vector of values, $\mathbf{F}_1(\mu, \sigma_1) \in \mathcal{U}^{|Y_1|}, \ldots \mathbf{F}_i(\mu, \sigma_1, \ldots \sigma_i) \in \mathcal{U}^{|\mathbf{Y}_i|}$ where for each $1 \leq j \leq i$, $\mathbf{F}_j$ is a $|\mathbf{Y}_j|$-dimensional vector of Skolem functions, each of arity $|\mathbf{Z}| + |\mathbf{X}_1| + \ldots + |\mathbf{X}_j|$.

The proof is by induction on i. For $i = 1$, let $\xi(\mathbf{Z}) \equiv \forall\mathbf{X}_1\exists\mathbf{Y}_1\, \xi_1(\mathbf{Z}, \mathbf{X}_1, \mathbf{Y}_1)$, where ξ_1 has one less number of quantifier alternations than ξ. On Skolemizing, we get $\xi^\star(\mathbf{Z}) \equiv \forall\mathbf{X}_1\, \xi_1(\mathbf{Z}, \mathbf{X}_1, \mathbf{F}_1(\mathbf{Z}, \mathbf{X}_1))$, where $\mathbf{F}_1$ is a $|\mathbf{Y}_1|$-dimensional vector of Skolem functions each of arity $|\mathbf{Z}| + |\mathbf{X}_1|$. We now design a Turing machine (or algorithm) M_1 that takes any $|\mathbf{Z}|+|\mathbf{X}_1|$-tuple of elements from $\mathcal{U}$, say (μ, σ_1), as input and halts after computing $\mathbf{F}_1(\mu, \sigma_1)$:

(a) It first determines if $\exists\mathbf{Y}_1\, \xi_1(\mu, \sigma_1, \mathbf{Y}_1)$ holds, using the decision procedure for $\mathcal{ED}(\mathfrak{M})$.

(b) If the answer to the above question is "Yes", the machine M_1 recursively enumerates $|\mathbf{Y}_1|$-tuples of elements of $\mathcal{U}$, and for each tuple ν thus enumerated, it checks if $\xi_1(\mu, \sigma_1, \nu)$ evaluates to true. Again the decidability of $\mathcal{ED}(\mathfrak{M})$ ensures that this check can also be effectively done. The machine M_1 outputs the first (in recursive enumeration order) element of $\mathcal{U}^{|\mathbf{Y}_1|}$, for which $\xi_1(\mu, \sigma_1, \nu)$ is true as $\mathbf{F}_1(\mu, \sigma_1)$, and halts.

(c) If the answer is "No", i.e. there is no $\nu \in \mathcal{U}^{|\mathbf{Y}_1|}$ s.t. $\xi_1(\mu, \sigma_1, \nu)$ is true, M_1 outputs the first (in recursive enumeration order) tuple of $\mathcal{U}^{|\mathbf{Y}_1|}$ as $\mathbf{F}_1(\mu, \sigma_1)$, and halts.

It is easy to verify that the vector of functions $\mathbf{F}_1$ computed by M_1 satisfies $\forall\mathbf{X}_1 \left(\exists\mathbf{Y}_1\, \xi_1(\mathbf{Z}, \mathbf{X}_1, \mathbf{Y}_1) \Leftrightarrow \xi_1(\mathbf{Z}, \mathbf{X}_1, \mathbf{F}_1(\mathbf{Z}, \mathbf{X}_1))\right)$ for every valuation of the free variables $\mathbf{Z}$ in $\mathcal{U}^{|\mathbf{Z}|}$, i.e., we have a (correct) $\mathfrak{M}$-interpretation of Skolem function $\mathbf{F}_1$. This completes the base case for $i = 1$.

For the general case of $i \geq 1$, we write $\xi(\mathbf{Z})$ as $\forall\mathbf{X}_1\exists\mathbf{Y}_1 \ldots \forall\mathbf{X}_i\exists\mathbf{Y}_i\xi_i(\mathbf{Z}, \mathbf{X}_1, \mathbf{Y}_1, \ldots \mathbf{X}_i, \mathbf{Y}_i)$, where ξ_i is a formula with i less $\forall^*\exists^*$ blocks than ξ. By induction hypothesis, we know that there exists a Turing machine M_i that takes as input any values for free variables $\mathbf{Z}$ and universally quantified variables $\mathbf{X}_1, \ldots \mathbf{X}_i$ and outputs values for $\mathbf{Y}_1, \ldots \mathbf{Y}_i$ so that they correspond to outputs of (correct) $\mathfrak{M}$-interpretations of vectors of Skolem functions $\mathbf{F}_1, \ldots \mathbf{F}_i$.

We need to show the existence of a computable interpretation of the vector of Skolem functions $\mathbf{F}_{i+1}$ for $\mathbf{Y}_{i+1}$. Thus, we are given a $(|\mathbf{Z}| + |\mathbf{X}_1| + \ldots + |\mathbf{X}_i| + |\mathbf{X}_{i+1}|)$-tuple of values from $\mathcal{U}$, and we need to show how to define a Turing machine M_{i+1} that

takes this vector as input and halts after computing values of vectors of Skolem functions $\mathbf{F}_1(\mu,\sigma_1),\dots\mathbf{F}_{i+1}(\mu,\sigma_1,\dots\sigma_i,\sigma_{i+1})$. Let $(\mu,\sigma_1,\dots,\sigma_i,\sigma_{i+1})$ be the given set of input values. The Turing machine M_{i+1} first simulates M_i on input $(\mu,\sigma_1,\dots,\sigma_i)$. This returns i vectors of values $\nu_1=\mathbf{F}_1(\mu,\sigma_1)\in\mathcal{U}^{|Y_1|},\dots\nu_i=\mathbf{F}_i(\mu,\sigma_1,\dots\sigma_i)\in\mathcal{U}^{|Y_i|}$ such that each of $\mathbf{F}_1,\dots,\mathbf{F}_i$ is a Skolem function vector. Plugging in all these values in ξ_i gives a sentence $\hat{\xi}_i=\xi_i[\mathbf{Z}\mapsto\mu,\mathbf{X}_1\mapsto\sigma_1,\mathbf{Y}_1\mapsto\nu_1,\dots,\mathbf{X}_i\mapsto\sigma_i,\mathbf{Y}_i\mapsto\nu_i]$. Observe that $\hat{\xi}_i\equiv\forall\mathbf{X}_{i+1}\exists\mathbf{Y}_{i+1}\hat{\xi}_{i+1}(\mathbf{X}_{i+1},\mathbf{Y}_{i+1})$ for some formula $\hat{\xi}_{i+1}(\mathbf{X}_{i+1},\mathbf{Y}_{i+1})$. We can then apply the same argument as in the base case above and conclude.

Note that the values of $\mathbf{Z},\mathbf{X}_1,\dots\mathbf{X}_n$ used above were arbitrary tuples from $\mathcal{U}^{|\mathbf{Z}|}$, $\mathcal{U}^{|\mathbf{X}_1|},\dots\mathcal{U}^{|\mathbf{X}_n|}$. Hence lifting the notation introduced in sentence (1) of Section 2 to talk about vector of variables $\mathbf{Y}_{i+1}$ instead of single variables $y_{i,j}$, we conclude that the vector of functions $\mathbf{F}_{i+1}$ computed by Turing machine M_{i+1} satisfies $\forall\mathbf{Z}\forall\mathbf{X}_1\dots\forall\mathbf{X}_{i+1}$ $\big(\exists\mathbf{Y}_{i+1}\xi^\star_{\mathbf{Y}_{i+1}}\ \Leftrightarrow\ \xi^\star_{\mathbf{Y}_{i+1}}[\mathbf{Y}_{i+1}\mapsto\mathbf{F}_{i+1}(\mathbf{Z},\mathbf{X}_1,\dots\mathbf{X}_{i+1})]\big)$. Thus, $\mathbf{F}_{i+1}$ gives a (correct) $\mathfrak{M}$-interpretation of a vector of Skolem functions for $\mathbf{Y}_{i+1}$, which completes the proof.

($\Rightarrow$) In the other direction, we will show that if there exists a halting Turing machine, say M, that synthesizes computable $\mathfrak{M}$-interpretations of Skolem functions for all existential variables in all $\mathcal{V}$-formulas, then $\mathcal{ED}(\mathfrak{M})$ must be decidable. We assume that there are at least two elements in $\mathcal{U}$; let's call them d_1, d_2. To show that $\mathcal{ED}(\mathfrak{M})$ is decidable, we need to show a decision procedure for the $\mathcal{V}(\mathfrak{M})$-theory of $\mathfrak{M}_C$. Consider any $\mathcal{V}(\mathfrak{M})$-sentence φ. This sentence may have finitely many constants that are not in $\mathcal{V}$. Let us say $c_1,\dots c_k$ are these constants, for some non-negative integer k. We introduce k fresh variables $y_1,\dots y_k$ and define φ' to be the formula obtained by taking φ and replacing each occurrence of the constant c_i by variable y_i respectively. Note that φ' is a $\mathcal{V}$-formula. We introduce 3 more fresh variables x,z_1,z_2 and consider the sentence $\psi\equiv\forall y_1\dots\forall y_k\forall z_1\forall z_2\exists x\big(((x=z_1)\wedge\varphi')\vee((x=z_2)\wedge\neg\varphi')\big)$. We can easily rewrite this formula in prenex normal form, but doing so still leaves x as the leftmost existentially quantified variable. We now feed this prenex normal form of ψ as the input to Turing machine M (that synthesizes computable interpretations of Skolem functions for all existential variables in all $\mathcal{V}$-formulas). Let the computable interpretation of the Skolem function for x, as output by M, be the Turing machine M_x. Therefore, M_x takes as inputs values of $y_1,\dots y_k,z_1,z_2$ (as these are the only universally quantified variables to the left of x in the quantifier prefix) and it always halts after computing a value for x. Now we run the Turing machine M_x with the inputs: c_i as the value for y_i for $i\in\{1,\dots k\}$, d_1 as the value for z_1 and d_2 as the value for z_2. Let the value computed by M_x with these inputs be t. We then check if $t=d_1$. If yes, we conclude that $\mathfrak{M}_C\models\varphi$, else $\mathfrak{M}_C\not\models\varphi$. Thus, we have a decision procedure for $\mathcal{V}(\mathfrak{M})$-theory of $\mathfrak{M}_C$, i.e., $\mathcal{ED}(\mathfrak{M})$ is decidable. ◀

The construction in the first part of the above proof shows that there exists a Turing machine that runs in time polynomial in the length of ξ and in the length of a decision procedure for $\mathcal{ED}(\mathfrak{M})$, and generates a computable interpretation (i.e. code for a halting Turing machine) that computes $\mathfrak{M}$-interpretations of all Skolem functions in ξ. Further, since a positive answer to SkolemSynthesis implies a positive answer to SkolemExist, Theorem 10 also gives a sufficient condition for SkolemExist to have a positive answer.

7 Applications and complexity

We now look at some consequences of the above characterization. We first ask if we can algorithmically synthesize computable interpretations of Skolem functions in some well-known theories in first-order logic. We start with a lemma.

▶ **Lemma 11.** *Let $\mathfrak{M}$ be a computable $\mathcal{V}$-structure with universe $\mathcal{U}$. Suppose for every element $e \in \mathcal{U}$, there exists an effectively computable uni-variate $\mathcal{V}$-formula $\alpha_e(x)$ such that $\alpha_e(x)$ is* true *iff $x = e$. Then $Th(\mathfrak{M})$ is decidable iff $\mathcal{ED}(\mathfrak{M})$ is decidable.*

One may wonder if decidability of $Th(\mathfrak{M})$ automatically implies decidability of $\mathcal{ED}(\mathfrak{M})$. However, this is not true in general (see [2] for more details), emphasizing the need for Lemma 11. From Lemma 11 and Theorem 10 we now have,

▶ **Corollary 12.** *For the following theories, both* SkolemExist *and* SkolemSynthesis *have positive answers, and we can effectively synthesize computable $\mathfrak{M}$-interpretations for Skolem functions for a $\mathcal{V}$-formula: 1. Presburger arithmetic; 2. Linear rational arithmetic (LRA); 3. Theory of real algebraic numbers; 4. Theory of dense linear orders without endpoints.*

For the theory of natural numbers with addition, multiplication and order, we have seen in Proposition 8 that SkolemExist has a negative answer, which of course implies that SkolemSynthesis cannot have a positive answer. Using Lemma 11 and Theorem 10 we get a direct proof for the latter fact. To see this note that the premise of Lemma 11 holds for this theory as for Presburger arithmetic. Hence, $\mathcal{ED}(\mathfrak{M})$ is decidable iff $Th(\mathfrak{M})$ is decidable. But we know from the MRDP theorem [11] that the latter is indeed undecidable. Thus, from Theorem 10, we obtain that SkolemSynthesis has negative instances in this theory.

We remark that the above discussion can also be seen as an alternate proof of the fact that the elementary diagram is undecidable, since Theorem 10 is a characterization.

Complexity bounds on $\mathfrak{M}$-interpretations. When it applies, the proof of Theorem 10 gives us a construction of a Turing machine M that takes a formula ξ as input and outputs a computable $\mathfrak{M}$-interpretation (i.e. code for another Turing machine, say M') of Skolem functions for all existential variables in ξ. What bounds can we give on the worst case running time of M' (Problem 4.c in Section 4)? We start with a lower bound that follows from the second part of the proof of Theorem 10.

▶ **Theorem 13.** *Let $\mathfrak{M}$ be a computable $\mathcal{V}$-structure with a decidable $\mathcal{ED}(\mathfrak{M})$. The worst case running time of any computable $\mathfrak{M}$-interpretation of Skolem functions for a $\mathcal{V}$-formula is at least as much as that of a decision procedure for $\mathcal{ED}(\mathfrak{M})$.*

This shows for instance that for Presburger arithmetic, there exists formulas for which any computable $\mathfrak{M}$-interpretation of Skolem functions will take at least (alternating) double exponential time [8, 15] Next, for upper bounds, the computable $\mathfrak{M}$-interpretation of Skolem functions, as detailed in the proof of Theorem 10, relies on enumeration. Hence, it does not help in giving complexity upper bounds. However, if a theory admits effective constraint solving (see Sec. 2 for a definition), then we can do better.

▶ **Theorem 14.** *Let $\mathfrak{M}$ be a $\mathcal{V}$-structure such that $\mathcal{ED}(\mathfrak{M})$ is decidable. Suppose $\mathcal{ED}(\mathfrak{M})$ admits effective constraint solving with worst-case time complexity $T(n)$ and the solution is represented as a tuple of domain elements requiring at most $S(n)$ bits. Then we can synthesize $\mathfrak{M}$-interpretations of Skolem functions for $\mathcal{V}$-formulas of size n, such that the running time and output size of the $\mathfrak{M}$-interpretations are bounded by recursive functions of $T(n)$ and $S(n)$.*

As an example, if $S(n)$ is linear, i.e., $S(n) \leq C.n$ for a constant $C > 0$, then we get $time(n) \leq k.T(k.(\max\{C, 1\})^k.n)$ and $size(n) \leq k.(\max\{C, 1\})^k.n$. Finally, one way to obtain an algorithm for effective constraint solving is by using effective quantifier elimination

repeatedly and then using quantifier-free constraint solving. Thus, we could further bound the complexity as functions of the complexity for effective quantifier elimination and that of quantifier-free constraint solving. This can be applied, for example, for LRA, theory of reals etc. Significantly, there are first-order theories that do not admit effective quantifier elimination but admit effective constraint solving, e.g., theory of evaluated trees [10]. In such cases, we can still use our approach to synthesize Skolem functions.

8 Expressing Skolem functions as terms

Whenever Skolem functions are computable, one can further ask: *Can Skolem functions be represented as terms?* Notice that in the Boolean setting, the notions of terms, functions and formulas are often conflated (as noted by Jiang [17] as well). Note that there are theories without any terms, for which Skolem functions can still be synthesized as halting Turing machines. For instance, the theory of (countable) dense linear order without endpoints does not admit any terms. Yet, from Corollary 12, we know that we can effectively synthesize computable Skolem functions for this theory. In fact, we can show a stronger result, viz, even when the theory admits terms, we may not be able to interpret a Skolem function as a term. To see this, consider the Presburger formula: $\forall y \forall z \exists x(((x = y) \vee (x = z)) \wedge ((x \geq y) \wedge (x \geq z)))$. The unique Skolem function for x is $\max(y, z)$, which can be written as an imperative program as: "**input(y,z); if** $\mathbf{y \geq z}$ **then return** y **else return** z". This is a uniform representation (see Sec. 2) of a computable $\mathfrak{M}$-interpretation of the Skolem function for x. However, this function cannot be written as a term in Presburger arithmetic. Indeed, any term of y, z that uses only $+$, 0, 1 must be linear, while max is a non-linear function. Thus, we have

▶ **Proposition 15.** *There exist first order theories for which Skolem functions can be effectively computed, but they cannot be expressed as terms.*

As described in [17], if Skolem functions in a first order theory can be represented using a *finite set of conditional terms* (like in the case of $\max(y, z)$ above), the theory admits effective quantifier elimination. However, we already have first order theories, e.g. the theory of evaluated trees, that don't admit quantifier elimination, but admit effective synthesis of computable interpretations of Skolem functions. In such cases, Skolem functions can't be represented as a finite set of conditional terms either.

Note that there is a related notion of deskolemization in proof theory (see e.g., [6], [7]) in which proofs of Skolemized formulas are related to the proofs of corresponding formulas without Skolem functions. However, this does not necessarily yield computable interpretations of Skolem functions as terms.

9 Conclusion

The study of algorithmic computation of Skolem functions is highly nuanced. We explored what it means for Skolem functions for first order logic to be computable and synthesizable. Defining computable interpretations of Skolem functions as Turing machines, we showed that they may not always exist and checking if they exist is undecidable in general. However, when we fix a computable structure, we gave a precise characterization of when they exist and show several applications for specific theories. While we have made some preliminary progress regarding complexity issues, the question of synthesizing succinct interpretations is still open as is the question of when Skolem functions can be represented as terms in the logic. We hope that the theoretical framework set up here will lead to research towards implementable synthesis of Skolem functions for first order logic.

References

1. S. Akshay, J. Arora, S. Chakraborty, S. Krishna, D. Raghunathan, and S. Shah. Knowledge compilation for Boolean functional synthesis. In *Formal Methods in Computer Aided Design, FMCAD 2019, San Jose, CA, USA*, pages 161–169. IEEE, 2019.
2. S. Akshay and S. Chakraborty. On synthesizing Skolem functions for first order logic formulae. *CoRR*, abs/2102.07463, 2021. arXiv:2102.07463.
3. S. Akshay, S. Chakraborty, S. Goel, S. Kulal, and S. Shah. What's hard about Boolean functional synthesis? In *Computer Aided Verification – 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part I*, volume 10981 of *Lecture Notes in Computer Science*, pages 251–269. Springer, 2018.
4. S. Akshay, S. Chakraborty, S. Goel, S. Kulal, and S. Shah. Boolean functional synthesis: hardness and practical algorithms. *Form Methods Syst Des.*, 57(1):53–86, 2021.
5. S. Akshay, S. Chakraborty, A. K. John, and S. Shah. Towards parallel boolean functional synthesis. In *TACAS 2017 Proceedings, Part I*, pages 337–353, 2017. doi:10.1007/978-3-662-54577-5_19.
6. J. Avigad. Eliminating definitions and Skolem functions in first-order logic. *ACM Trans. Comput. Log.*, 4(3):402–415, 2003. doi:10.1145/772062.772068.
7. M. Baaz, S. Hetzl, and D. Weller. On the complexity of proof deskolemization. *J. Symb. Log.*, 77(2):669–686, 2012. doi:10.2178/jsl/1333566645.
8. L. Berman. The complexity of logical theories. *Theoretical Computer Science*, 11(1):71–77, 1980. doi:10.1016/0304-3975(80)90037-7.
9. E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer, 1997.
10. T. Dao and K. Djelloul. Solving first-order constraints in the theory of the evaluated trees. In *Proceedings of the Constraint Solving and Contraint Logic Programming 11th Annual ERCIM International Conference on Recent Advances in Constraints*, CSCLP'06, pages 108–123. Springer-Verlag, 2006.
11. M. Davis, Y. Matijasevic, and J. Robinson. Hilbert's tenth problem. Diophantine equations: positive aspects of a negative solution. In *Proceedings of symposia in pure mathematics*, volume 28, pages 323–378, 1976.
12. K. Fazekas, M. J. H. Heule, M. Seidl, and A. Biere. Skolem function continuation for quantified Boolean formulas. In *International Conference on Tests and Proofs (TAP)*, volume 10375 of *Lecture Notes in Computer Science*, pages 129–138. Springer, 2017.
13. D. Fried, L. M. Tabajara, and M. Y. Vardi. BDD-based boolean functional synthesis. In *Computer Aided Verification – 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II*, pages 402–421, 2016.
14. P. Golia, S. Roy, and K. S. Meel. Manthan: A data-driven approach for boolean function synthesis. In *Computer Aided Verification – 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part II*, volume 12225 of *Lecture Notes in Computer Science*, pages 611–633. Springer, 2020.
15. C. Haase. A survival guide to presburger arithmetic. *ACM SIGLOG News*, 5(3):67–82, 2018.
16. M. Heule, M. Seidl, and A. Biere. Efficient Extraction of Skolem Functions from QRAT Proofs. In *Formal Methods in Computer-Aided Design , FMCAD 2014, Lausanne, Switzerland*, pages 107–114, 2014.
17. J.-H. R. Jiang. Quantifier elimination via functional composition. In *Proc. of CAV*, pages 383–397. Springer, 2009.
18. A. John, S. Shah, S. Chakraborty, A. Trivedi, and S. Akshay. Skolem functions for factored formulas. In *Formal Methods in Computer-Aided Design, FMCAD 2015*, pages 73–80, 2015.
19. V. Kuncak, M. Mayer, R. Piskac, and P. Suter. Complete functional synthesis. In *Proceedings of the 2010 ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2010, Toronto, Ontario, Canada, June 5-10, 2010*, pages 316–329. ACM, 2010.

20 M. Preiner, A. Niemetz, and A. Biere. Counterexample-guided model synthesis. In *TACAS (1)*, volume 10205 of *Lecture Notes in Computer Science*, pages 264–280, 2017.

21 M. N. Rabe. Incremental determinization for quantifier elimination and functional synthesis. In *Computer Aided Verification – 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part II*, pages 84–94, 2019.

22 M. N. Rabe and S. A. Seshia. Incremental determinization. In *Theory and Applications of Satisfiability Testing – SAT 2016 – 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings*, pages 375–392, 2016. doi:10.1007/978-3-319-40970-2_23.

23 M. N. Rabe, L. Tentrup, C. Rasmussen, and S. A. Seshia. Understanding and extending incremental determinization for 2QBF. In *Computer Aided Verification – 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part II*, pages 256–274, 2018.

24 Preey Shah, Aman Bansal, S. Akshay, and Supratik Chakraborty. A normal form characterization for efficient boolean skolem function synthesis. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 – July 2, 2021*, pages 1–13. IEEE, 2021.

25 A. Spielmann and V. Kuncak. Synthesis for unbounded bit-vector arithmetic. In *Automated Reasoning – 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*, volume 7364 of *Lecture Notes in Computer Science*, pages 499–513. Springer, 2012.

26 S. Srivastava, S. Gulwani, and J. S. Foster. From program verification to program synthesis. *SIGPLAN Not.*, 45(1):313–326, 2010. doi:10.1145/1707801.1706337.

27 L. M. Tabajara and M. Y. Vardi. Factored boolean functional synthesis. In *Formal Methods in Computer Aided Design, FMCAD 2017, Vienna, Austria, October 2-6, 2017*, pages 124–131, 2017.

28 S. Verma and S. Roy. Debug-localize-repair: a symbiotic construction for heap manipulations. *Formal Methods Syst. Des.*, 58(3):399–439, 2021. doi:10.1007/s10703-021-00387-z.