International Journal of Innovative Computing, Information and Control
Volume **17**, Number **1**, February **2021**
 ISSN 1349-4198
pp. 1–13

SKOLEMIZATION THAT PRESERVES LOGICAL MEANINGS

Kiyoshi Akama[1] and Ekawit Nantajeewarawat[2,*]

[1]Information Initiative Center
Hokkaido University
Kita 11, Nishi 5, Kita-ku, Sapporo, Hokkaido 060-0811, Japan
akama@iic.hokudai.ac.jp

[2]School of Information, Computer and Communication Technology
Sirindhorn International Institute of Technology
Thammasat University
99 Moo 18, Km. 41 on Paholyothin Highway, Khlong Luang, Pathum Thani 12120, Thailand
*Corresponding author: ekawit@siit.tu.ac.th

Received August 2020; revised December 2020

Abstract. *Skolemization is a well-known method for removing existential quantifiers from a logical formula. Although it always yields a satisfiability-preserving transformation step, conventional Skolemization in general does not preserve the logical meaning of a source formula. We develop in this paper a theory for extending the space of first-order formulas by incorporation of function variables and show how meaning-preserving Skolemization can be achieved in an obtained extended space. A procedure for converting a logical formula into an equivalent one in an extended conjunctive normal form on the extended space is described. This work lays a theoretical foundation for solving logical problems involving existential quantifications based on meaning-preserving formula transformation.*
Keywords: Skolemization, First-order logic, Equivalent transformation, Conjunctive normal form, Query-answering problems

1. **Introduction.** Conversion of a given formula into a conjunction of clauses, called a conjunctive normal form (CNF) or a clausal normal form, is referred to as *formula decomposition* or, for short, *decomposition* in this paper. The conventional Skolemization-based decomposition (CSD) is commonly used in automated reasoning [1, 2, 3, 4, 5, 6, 7]. It involves removal of existential quantifications by Skolemization [2, 8] (named after Thoralf Albert Skolem), i.e., by replacement of an existentially quantified variable with a Skolem term, which is determined by a relevant part of a formula prenex.

Recently, query-answering problems (QA problems) have gained wide attention [7, 9, 10, 11, 12, 13]. A problem in this class is concerned with finding the set of all ground instances of a given query atom that are logical consequences of a given formula. Equivalent transformation (ET) of formulas is essential and very useful for solving many kinds of logical problems [14, 15], including QA problems. In ET-based problem solving, a logical formula representing a given problem is successively transformed into a simpler but logically equivalent formula. Correctness of computation is readily guaranteed by any combinations of equivalent transformations, which yield many kinds of correct algorithms for solving logical problems.

Since conventional Skolemization does not generally preserve the logical meaning of a given formula, it cannot be used in an ET-based problem-solving process. The formula

DOI: 10.24507/ijicic.17.01.1

resulting from Skolemization is not necessarily equivalent to the original one. Only the satisfiability property of a formula is preserved. The resulting formula is equisatisfiable with the original formula [16], i.e., the satisfiability of the resulting formula and that of the original formula are equivalent. For instance, the first-order formula $\exists x : p(x)$ and its corresponding Skolemized form $p(c)$, where c is a Skolem constant, are both satisfiable, but they are not logically equivalent, e.g., the latter formula logically entails $p(c)$ while the former one does not.

Decomposition that preserves the set of all models of a formula is called *meaning-preserving decomposition* (MPD). It can be used in an ET-based problem-solving process safely. Our objective here is to develop MPD. However, as long as we consider first-order logic, there is no formula decomposition that preserves logical meanings in general. Our idea is (i) to develop a theory for extending a space of logical formulas by introduction of function variables and (ii) to achieve MPD by introducing meaning-preserving Skolemization in the obtained extended space.

The conventional first-order logic cannot solve many logical problems correctly whatever procedures are taken, which is a representational and transformational limitation. Introduction of function variables means extension of first-order logic, which overcomes the limitation of representation and transformation. The theory of MPD leads us to a new logic, where more logical problems can be solved with guarantee of correctness, compared to the conventional logic based on first-order formulas [17, 18, 19]. Extension of first-order logic along with MPD in this paper overcomes the limitation of the conventional logic.

The paper is organized as follows. Section 2 formalizes a class of QA problems and explains the necessity of meaning-preserving decomposition and logical-space extension. Section 3 takes a Tax-cut problem as an example, and compares conventional Skolemization and meaning-preserving Skolemization, showing the change of the meaning of the problem. After introducing function variables, Section 4 formulates an extended logical space, and presents an extended conjunctive normal form, called existentially quantified conjunctive normal form (ECNF). Section 5 proposes an algorithm for meaning-preserving conversion of a formula into a formula in an ECNF on the extended logical space, which gives a general solution method for all QA problems on first-order formulas. Section 6 concludes the paper.

The notation that follows holds thereafter. Given a partial mapping f, $dom(f)$ denotes the domain of f (i.e., $dom(f) = \{x \mid \langle x, y\rangle \in f\}$). Given a set A, $pow(A)$ denotes the power set of A.

2. Need for Meaning-Preserving Skolemization.

Conventional Skolemization and meaning-preserving Skolemization are compared. The conventional Skolemization-based decomposition does not preserve logical meanings. In order to solve QA problems, space-extension of first-order formulas and introduction of meaning-preserving decomposition are necessary.

2.1. Query-answering problems.

A *query-answering problem* (*QA problem*) is a pair $\langle K, q\rangle$, where K is a logical formula and q is an atomic formula (atom). The *answer* to a QA problem $\langle K, q\rangle$, denoted by $answer(K, q)$, is defined by

$$answer(K, q) = \{g \mid (g \text{ is a ground instance of } q) \ \&\ (K \models g)\},$$

i.e., the set of all ground instances of q that follow logically from K. When K consists of only definite clauses, problems in this class are problems that have been discussed in logic programming [10]. When K is a conjunction of axioms and assertions in Description Logics [11], QA problems are usually called *question-answering problems* [12]. Our target

in this paper is the class of QA problems on full first-order formulas, which is a superset of the two problem classes stated above.

2.2. **Formalization of QA problems.** Let q be an atom with a predicate p. The answer to a QA problem $\langle K, q\rangle$ can be equivalently represented as

$$\mathit{answer}(K, q) = \varphi\left(\bigcap \mathit{Models}(K')\right),$$

where

- q' is an atom obtained from q by replacing the predicate p in q with a new predicate p',
- $Q = (\forall x_1, \ldots, x_n : (q \rightarrow q'))$, where $x_1, \ldots, x_n$ are the variables occurring in q, i.e., Q is the universal closure of $(q \rightarrow q')$,
- $K' = K \wedge Q$,
- $\mathit{Models}(K')$ is the set of all models of K',
- $\bigcap \mathit{Models}(K')$ is the intersection of all models of K',
- φ is a mapping that associates with any arbitrarily given set G of ground atoms the set $\varphi(G)$ of ground atoms obtained from G by replacing every occurrence of the predicate p' with p.

2.3. **Conventional Skolemization.** It is well-known that, in the traditional conversion of first-order formulas into CNFs, the conventional Skolemization results in a conversion step that does not preserve the logical meaning of a given formula. For example, the formula

$$\forall x, \exists y : p(x, y) \tag{1}$$

is Skolemized to $\forall x : p(x, f(x))$, where f is a new function constant, called a Skolem function. It is obvious that $\forall x, \exists y : p(x, y)$ and $\forall x : p(x, f(x))$ have different meanings. Given any arbitrary ground term t_x, the former formula states the existence of a ground term t_y such that $p(t_x, t_y)$ is true, while the latter formula states not only the existence of such a ground term t_y but also that one such t_y is $f(t_x)$. The set $\{p(t, 3) \mid t \text{ is a ground term}\}$, for example, is a model of the former formula but is not a model of the second one. Except for a Skolemization step, all transformation steps in formula conversion into a CNF are basically equivalent transformation. They include, for example, the implication law $(p \rightarrow q \equiv \neg p \vee q)$ and the De Morgan's laws.

2.4. **Introduction of meaning-preserving Skolemization.** The basic idea of meaning-preserving Skolemization is to use existentially quantified function variables instead of function symbols. For example, Formula (1) is transformed into

$$\exists h, \forall x : p(x, h(x)), \tag{2}$$

where h is a function variable. Intuitively, h is an unknown function that associates with any arbitrarily given ground term t_x a ground term $h(t_x)$ such that $p(t_x, h(t_x))$ is true. An alternative form of (2) is

$$\exists h, \forall x, y : (p(x, y) \vee (h(x) \neq y)), \tag{3}$$

which is intuitively equivalent to (2).

2.5. **Need for an extended space.** Let $\mathcal{L}_1$ be the set of all conventional first-order formulas. The conventional Skolemization-based decomposition (CSD) does not preserve the set of all models of some first-order formula, i.e.,

$$\exists E \in \mathcal{L}_1 : \mathit{Models}(\mathrm{CSD}(E)) \neq \mathit{Models}(E),$$

where for any first-order formula E, CSD(E) denotes the formula obtained by applying CSD to E. We want to develop new decomposition, called meaning-preserving decomposition (MPD), that preserves the set of all models of any first-order formula, i.e.,

$$\forall E \in \mathcal{L}_1 : \mathit{Models}(\mathrm{MPD}(E)) = \mathit{Models}(E),$$

where for any first-order formula E, MPD(E) is the formula obtained by applying MPD to E. Let $\mathrm{CSD}(\mathcal{L}_1) = \{\mathrm{CSD}(E) \mid E \in \mathcal{L}_1\}$ and $\mathrm{MPD}(\mathcal{L}_1) = \{\mathrm{MPD}(E) \mid E \in \mathcal{L}_1\}$. We know that $\mathrm{CSD}(\mathcal{L}_1) \subseteq \mathcal{L}_1$. However, we cannot realize MPD that satisfies the following conditions:

- Model-preservation, i.e., $\forall E \in \mathcal{L}_1 : \mathit{Models}(\mathrm{MPD}(E)) = \mathit{Models}(E)$.
- Closedness in $\mathcal{L}_1$, i.e., $\mathrm{MPD}(\mathcal{L}_1) \subseteq \mathcal{L}_1$.

We need an extended space $\mathcal{L}_2$ that includes not only usual terms, atoms, and formulas in $\mathcal{L}_1$, but also function variables, quantifications on function variables, and formulas containing them. This paper constructs $\mathcal{L}_2$ and proposes MPD that satisfies the following conditions:

- Model-preservation, i.e., $\forall E \in \mathcal{L}_1 : \mathit{Models}(\mathrm{MPD}(E)) = \mathit{Models}(E)$.
- Space extension to $\mathcal{L}_2$, i.e., $\mathrm{MPD}(\mathcal{L}_1) \subseteq \mathcal{L}_2$.

2.6. **Limitations of conventional logic and logical computation.** Proof problems historically constitute the most important problem class in logical problem solving [1, 2, 8, 16]. Resolution provides us a refutation proof procedure for logical formulas. Based on the resolution proof method, automated theorem proving has been extensively investigated [3, 4, 5, 6, 20].

Resolution is sound and complete, i.e., for any clause set Cs, $\mathit{Models}(Cs) = \emptyset$ if and only if $Cs \vdash \square$, where $Cs \vdash \square$ means that there is a deduction of Cs that results in an empty clause. CSD maps a first-order formula to a set of clauses preserving satisfiability. Let F be a first-order formula and $\mathrm{CSD}(F) = Cs$. Then $\mathit{Models}(F) = \emptyset$ if and only if $\mathit{Models}(Cs) = \emptyset$, and hence $\mathit{Models}(F) = \emptyset$ if and only if $Cs \vdash \square$. This proof method can be looked at as a satisfiability-based method.

Being inspired by the resolution proof method, Prolog and a theory of SLD-resolution were invented for solving QA problems [7, 9, 10, 13]. Soundness and completeness theorems were established [21, 22]. Skolemization was still assumed in formula decomposition, and QA problems on clauses were considered. However, CSD does not necessarily preserve the answers to QA problems. A general theory for solving QA problem on full first-order formulas cannot be constructed if we stay in the current first-order formula space, i.e., $\mathcal{L}_1$.

The ET-based approach, on the other hand, is ideal for guarantee of correctness of computation. The most general ET is answer-preserving transformation. Model preservation and model-intersection preservation are sufficient conditions for answer-preservation. As will be proposed in Section 5, MPD is realized as a model-preserving mapping in the space $\mathcal{L}_2$. So far, many problems that cannot be solved in the space $\mathcal{L}_1$ with resolution have been discovered and solved with the correctness of computation being guaranteed strictly [14, 15, 17, 19, 23].

3. **Conventional and New Decomposition by Example.** CSD and MPD are compared by using an example problem, called a "Tax-cut" problem, which is simplified from the original problem in [24]. MPD preserves the answer to this example problem, while CSD does not. The precise definition of MPD will be determined by the algorithm given in Section 5.

3.1. **A simplified Tax-cut problem.** The "Tax-cut" problem is to find all persons who can have discounted tax, with the knowledge that:

1) Any person who has two children or more can get discounted tax.
2) Peter has a child.
3) Peter has a child named Paul.

This problem can be represented in first-order logic using the formulas F_1-F_3 below, where *neq* stands for "not equal".

F_1: $\forall x, y, z : ((\mathit{hasChild}(x, y) \wedge \mathit{hasChild}(x, z) \wedge \mathit{neq}(y, z)) \rightarrow \mathit{TaxCut}(x))$

F_2: $\exists x : (\mathit{hasChild}(\mathit{Peter}, x))$

F_3: $\mathit{hasChild}(\mathit{Peter}, \mathit{Paul})$

In this section, we show that

- the correct answer to the "Tax-cut" problem is the empty set, which is obtained from the model-based definition of the answer (Section 3.2),
- the answer obtained by the conventional Skolemization-based decomposition (CSD) is not the empty set (Section 3.3), and
- the answer obtained by meaning-preserving decomposition (MPD) is the empty set (Section 3.4).

It is concluded that revision of formula decomposition from the conventional decomposition to meaning-preserving decomposition is necessary.

3.2. **The correct answer to the problem.** The "Tax-cut" problem is a query-answering (QA) problem $\langle K, q\rangle$, where $K = F_1 \wedge F_2 \wedge F_3$ and $q = \mathit{TaxCut}(x)$.

Let $\mathcal{G}_U$ be the set of all ground user-defined atoms. A model of a first-order formula is a set of ground user-defined atoms with respect to which the formula is true. Let $Q = (\forall x : \mathit{TaxCut}(x) \rightarrow \mathit{TaxCut}'(x))$. Note that F_3 implies F_2. Hence $\mathit{Models}(K \wedge Q) = \mathit{Models}(F_1 \wedge F_3 \wedge Q)$. Then there exist infinitely many models of $K \wedge Q$ and

$$\{S_0\} \ \subseteq \ \mathit{Models}(K \wedge Q) \ \subseteq \ \{S_0\} \cup M_1,$$

where

- $S_0 = \{\mathit{hasChild}(\mathit{Peter}, \mathit{Paul})\}$, and
- $M_1 = \{S_0 \cup G \mid G \subseteq \mathcal{G}_U\}$.

We use the following proposition for minimal models.

Proposition 3.1. *Let G be a set. Assume that $S \subseteq G$, $Md \subseteq \mathit{pow}(G)$, and $Ms \subseteq \mathit{pow}(G)$. If $\{S\} \subseteq Md \subseteq \{S\} \cup Ms$ and $\bigcap Ms \supseteq S$, then $\bigcap Md = S$.*

Proof: Since $\bigcap(\{S\} \cup Ms) = S \cap (\bigcap Ms) = S$, we have $S \supseteq \bigcap Md \supseteq S$. Hence $\bigcap Md = S$. □

It follows, from Proposition 3.1, that $\bigcap \mathit{Models}(K \wedge Q) = S_0$. Hence

$$\mathit{answer}(K, q) = \varphi\left(\bigcap \mathit{Models}(K \wedge Q)\right) = \varphi(S_0) = \{\},$$

where φ is a mapping that associates with any arbitrarily given set G of ground atoms the set $\varphi(G)$ defined by $\{\, \mathit{TaxCut}(x) \mid \mathit{TaxCut}'(x) \in G\}$.

3.3. **Conventional decomposition.** From the first-order formulas F_1, F_2, F_3, and Q, we obtain the following clauses C_1, C_2, C_3, and C_4 by using the conventional Skolemization-based decomposition (CSD), where f_1 is a Skolem function:

C_1: $\mathit{TaxCut}(x) \leftarrow \mathit{hasChild}(x, y), \mathit{hasChild}(x, z), \mathit{neq}(y, z)$

C_2: $\mathit{hasChild}(\mathit{Peter}, f_1) \leftarrow$

C_3: $\mathit{hasChild}(\mathit{Peter}, \mathit{Paul}) \leftarrow$

C_4: $\mathit{TaxCut}'(x) \leftarrow \mathit{TaxCut}(x)$

The transformations $F_1 \Rightarrow C_1$, $F_3 \Rightarrow C_3$, and $Q \Rightarrow C_4$ are equivalent transformations. However, the transformation of F_2 into C_2 is not an equivalent transformation. The atom $\mathit{hasChild}(\mathit{Peter}, f_1)$ is true in every model of C_2, while it is false in some model of F_2. By using a set of clauses, the conjunction of the clauses is represented. There exist many models of $\{C_1, C_2, C_3, C_4\}$, and

$$\{S_0 \cup S_1 \cup S_2\} \subseteq \mathit{Models}(\{C_1, C_2, C_3, C_4\}) \subseteq \{S_0 \cup S_1 \cup S_2\} \cup M_2,$$

where

- $S_0 = \{\mathit{hasChild}(\mathit{Peter}, \mathit{Paul})\}$,
- $S_1 = \{\mathit{TaxCut}(\mathit{Peter}), \mathit{TaxCut}'(\mathit{Peter})\}$,
- $S_2 = \{\mathit{hasChild}(\mathit{Peter}, f_1)\}$, and
- $M_2 = \{S_0 \cup S_1 \cup S_2 \cup G \mid G \subseteq \mathcal{G}_U\}$.

It follows, from Proposition 3.1, that $\bigcap \mathit{Models}(\{C_1, C_2, C_3, C_4\}) = S_0 \cup S_1 \cup S_2$. Hence

$$\bigcap \mathit{Models}(K \wedge Q) \neq \bigcap \mathit{Models}(\{C_1, C_2, C_3, C_4\}).$$

Non-preservation of the model intersection above comes from the Skolemization step included in the conversion of F_2 into C_2. It follows that

$$\varphi\left(\bigcap \mathit{Models}(\{C_1, C_2, C_3, C_4\})\right) = \varphi(S_0 \cup S_1 \cup S_2) = \{\mathit{TaxCut}(\mathit{Peter})\} \neq \mathit{answer}(K, q).$$

The answer to the simplified "Tax-cut" problem is not preserved by CSD.

3.4. **Meaning-preserving decomposition.** From the first-order formulas F_1, F_2, F_3, and Q, we obtain the following clauses C_1, C_2', C_3, and C_4 by meaning-preserving decomposition (MPD), where h_1 is a function variable:

C_1: $\mathit{TaxCut}(x) \leftarrow \mathit{hasChild}(x, y), \mathit{hasChild}(x, z), \mathit{neq}(y, z)$

C_2': $\mathit{hasChild}(\mathit{Peter}, x) \leftarrow \mathit{func}(h_1, x)$

C_3: $\mathit{hasChild}(\mathit{Peter}, \mathit{Paul}) \leftarrow$

C_4: $\mathit{TaxCut}'(x) \leftarrow \mathit{TaxCut}(x)$

The transformations $F_1 \Rightarrow C_1$, $F_2 \Rightarrow C_2'$, $F_3 \Rightarrow C_3$, and $Q \Rightarrow C_4$ are all equivalent transformations. Let s be a term the existence of which is guaranteed by h_1 in C_2', i.e., $s = h_1()$. There are two cases:

- $s = \mathit{Paul}$. Then C_2' and C_3 are equivalent. *Peter*'s child is only *Paul*. Both $\mathit{TaxCut}(\mathit{Peter})$ and $\mathit{TaxCut}'(\mathit{Peter})$ are false.
- $s \neq \mathit{Paul}$. *Peter* has two children. Both $\mathit{TaxCut}(\mathit{Peter})$ and $\mathit{TaxCut}'(\mathit{Peter})$ are true.

Let $\mathcal{G}_T$ be the set of all ground terms. There exist infinitely many models of $\{C_1, C_2', C_3, C_4\}$ and

$$\{S_0\} \subseteq \mathit{Models}(\{C_1, C_2', C_3, C_4\}) \subseteq \{S_0\} \cup M_3 \cup M_3',$$

where

- $S_0 = \{\mathit{hasChild}(\mathit{Peter}, \mathit{Paul})\}$,
- $S_1 = \{\mathit{TaxCut}(\mathit{Peter}), \mathit{TaxCut}'(\mathit{Peter})\}$,
- for any $s \in \mathcal{G}_T$, $S_3(s) = \{\mathit{hasChild}(\mathit{Peter}, s)\}$,
- $M_3 = \{S_0 \cup S_1 \cup S_3(s) \cup G \mid (\mathit{Paul} \neq s \in \mathcal{G}_T)\ \&\ (G \subseteq \mathcal{G}_U)\}$, and
- $M_3' = \{S_0 \cup S_3(s) \cup G \mid (\mathit{Paul} = s \in \mathcal{G}_T)\ \&\ (G \subseteq \mathcal{G}_U)\}$.

It follows, from Proposition 3.1, that $\bigcap \mathit{Models}(\{C_1, C_2', C_3, C_4\}) = S_0$. Hence

$$\varphi\left(\bigcap \mathit{Models}(\{C_1, C_2', C_3, C_4\})\right) = \varphi(S_0) = \{\} = \mathit{answer}(K, q).$$

The answer to the simplified "Tax-cut" problem is preserved by MPD.

4. **An Extended Space for Formula Decomposition.** A new space $\mathcal{L}_2$ is constructed based on $\mathcal{L}_1$. The notions of function variables and formula trees are introduced. An extended conjunctive normal form, called an *existentially quantified conjunctive normal form* (*ECNF*), is defined.

4.1. **Function variables.** By meaning-preserving decomposition, a subformula

$$\forall x_1, \ldots, \forall x_n, \exists y : \beta,$$

where $n \geq 0$, is transformed into

$$\exists h, \forall x_1, \ldots, \forall x_n, \forall y : (\beta \vee \neg \mathit{func}(h, x_1, \ldots, x_n, y)),$$

where h is a function variable and *func* is a built-in predicate. The expression

$$\mathit{func}(h, x_1, \ldots, x_n, y)$$

is called a *func*-atom, the meaning of which is $h(x_1, \ldots, x_n) = y$. If h is instantiated into a function constant h_c, and $x_1, \ldots, x_n$ and y become ground terms $t_1, \ldots, t_n$ and s, respectively, then the meaning of this *func*-atom becomes $h_c(t_1, \ldots, t_n) = s$ and its truth value is determined. Let *Var* denote the set of all usual variables, *FVar* the set of all function variables, and *FCon* the set of all function constants. Formulas in $\mathcal{L}_2$ are constructed using logical connectives (i.e., $\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$), quantifications on usual variables, and quantifications on function variables.

4.2. **Formula trees.** We regard a formula in $\mathcal{L}_2$ as a tree, called a formula tree. Given a formula α in $\mathcal{L}_2$, the *formula tree* of α, denoted by $\mathit{FT}(\alpha)$, is a binary tree defined inductively as follows: If α is an atomic formula (atom), then $\mathit{FT}(\alpha)$ is a one-vertex binary tree whose root is α. If $\alpha = \neg\beta$, then $\mathit{FT}(\alpha)$ is a binary tree that has $\neg$ as its root and $\mathit{FT}(\beta)$ as its only immediate subtree. If $\alpha = \beta \wedge \gamma$ (respectively, $\beta \vee \gamma$, $\beta \rightarrow \gamma$, $\beta \leftrightarrow \gamma$), then $\mathit{FT}(\alpha)$ is a binary tree that has $\wedge$ (respectively, $\vee$, $\rightarrow$, $\leftrightarrow$) as its root, $\mathit{FT}(\beta)$ as its left immediate subtree, and $\mathit{FT}(\gamma)$ as its right immediate subtree. If $\alpha = \forall v : \beta$ (respectively, $\exists v : \beta$), where $v \in \mathit{Var}$, then $\mathit{FT}(\alpha)$ is a binary tree that has $\forall v$ (respectively, $\exists v$) as its root and $\mathit{FT}(\beta)$ as its only immediate subtree. If $\alpha = \forall v_h : \beta$ (respectively, $\exists v_h : \beta$), where $v_h \in \mathit{FVar}$, then $\mathit{FT}(\alpha)$ is a binary tree that has $\forall v_h$ (respectively, $\exists v_h$) as its root and $\mathit{FT}(\beta)$ as its only immediate subtree.

Each vertex of a formula tree is either an atom or an element of $\mathit{LCon} \cup \mathit{QVar} \cup \mathit{QFVar}$, where

- $\mathit{LCon} = \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$,
- $\mathit{QVar} = \{\forall v \mid v \in \mathit{Var}\} \cup \{\exists v \mid v \in \mathit{Var}\}$, and
- $\mathit{QFVar} = \{\forall v_h \mid v_h \in \mathit{FVar}\} \cup \{\exists v_h \mid v_h \in \mathit{FVar}\}$.

A $\forall v$-vertex and an $\exists v$-vertex in *QVar* are also called a $\forall$*Var-vertex* and an $\exists$*Var-vertex*, respectively. A $\forall v_h$-vertex and an $\exists v_h$-vertex in *QFVar* are also called a $\forall$*FVar-vertex* and an $\exists$*FVar-vertex*, respectively.

4.3. **Existentially quantified conjunctive normal form (ECNF).** A formula α in $\mathcal{L}_2$ is said to be in an *existentially quantified conjunctive normal form* (*ECNF*) if and only if α is a closed formula and every path from the root to a leaf of the formula tree of α consists of

1) zero or more $\exists FVar$-vertices, followed by
2) zero or more $\wedge$-vertices, followed by
3) zero or more $\forall Var$-vertices, followed by
4) zero or more $\vee$-vertices, followed by
5) either (i) a leaf vertex representing a usual atom or (ii) a $\neg$-vertex followed by a leaf vertex representing a usual atom or a *func*-atom.

A formula in an ECNF, also called an ECNF itself, is similar to a usual CNF in that it contains a conjunction of clauses, each of which is a disjunction of literals. There are, however, two main differences:

1) A formula in an ECNF contains existential quantifications on function variables; it has the form
$$\exists v_{h1}, \ldots, \exists v_{hn} : \beta,$$
where the v_{hi} are function variables and β has the same form as a usual CNF except that the negations of *func*-atoms may appear in β, i.e., β is a conjunction of disjunctions of (i) usual atoms, (ii) negated usual atoms, and (iii) negated *func*-atoms.
2) While usual Skolem functions may appear in usual atoms, function variables can appear only in *func*-atoms.

4.4. **An extended clause space.** Given usual atoms $a_1, \ldots, a_m$, $b_1, \ldots, b_n$ and *func*-atoms $\mathbf{f}_1, \ldots, \mathbf{f}_p$, a disjunction
$$a_1 \vee \cdots \vee a_m \vee \neg b_1 \vee \cdots \vee \neg b_n \vee \neg \mathbf{f}_1 \vee \cdots \vee \neg \mathbf{f}_p$$
contained in an ECNF is often written as
$$a_1, \ldots, a_m \leftarrow b_1, \ldots, b_n, \mathbf{f}_1, \ldots, \mathbf{f}_p.$$
Formulas in this class are similar to usual clauses, and are called *extended clauses*. The set of all extended clauses is denoted by $\mathrm{ECLS_F}$. A conjunction of a finite or infinite number of extended clauses is used for knowledge representation and also for computation. As usual, such a conjunction is usually dealt with by regarding it as a set of (extended) clauses. The *extended clause space* in this paper is the power set of $\mathrm{ECLS_F}$.

Let *Cs* be a set of extended clauses. Implicit existential quantifications of function variables and implicit clause conjunction are assumed in *Cs*. Function variables in *Cs* are all existentially quantified and their scope covers all clauses in *Cs*. With occurrences of function variables, clauses in *Cs* are connected through shared function variables. After instantiating all function variables in *Cs* into function constants, clauses in the instantiated set are totally separated.

5. **An Algorithm for Meaning-Preserving Decomposition.** An algorithm for transforming a formula in $\mathcal{L}_1$ into an equivalent ECNF in $\mathcal{L}_2$ is proposed. Based on the algorithm, a transformation scheme for solving a QA problem is given and its correctness is proved.

5.1. **The proposed conversion algorithm.** Assume that

- the initial set $\textsc{Ini} = \mathcal{L}_1$, which is a subset of $\mathcal{L}_2$, and
- the target set $\textsc{Fin}$ is the set of all ECNFs in $\mathcal{L}_2$.

Let a formula α in $\textsc{Ini}$ be given as input. Regard α as a formula in the space $\mathcal{L}_2$ and let $\mathbf{T} = FT(\alpha)$. To transform α into a formula in $\textsc{Fin}$, $\mathbf{T}$ is changed in the space $\mathcal{L}_2$ successively by the steps described below. Figure 1 depicts an outline of the procedure.

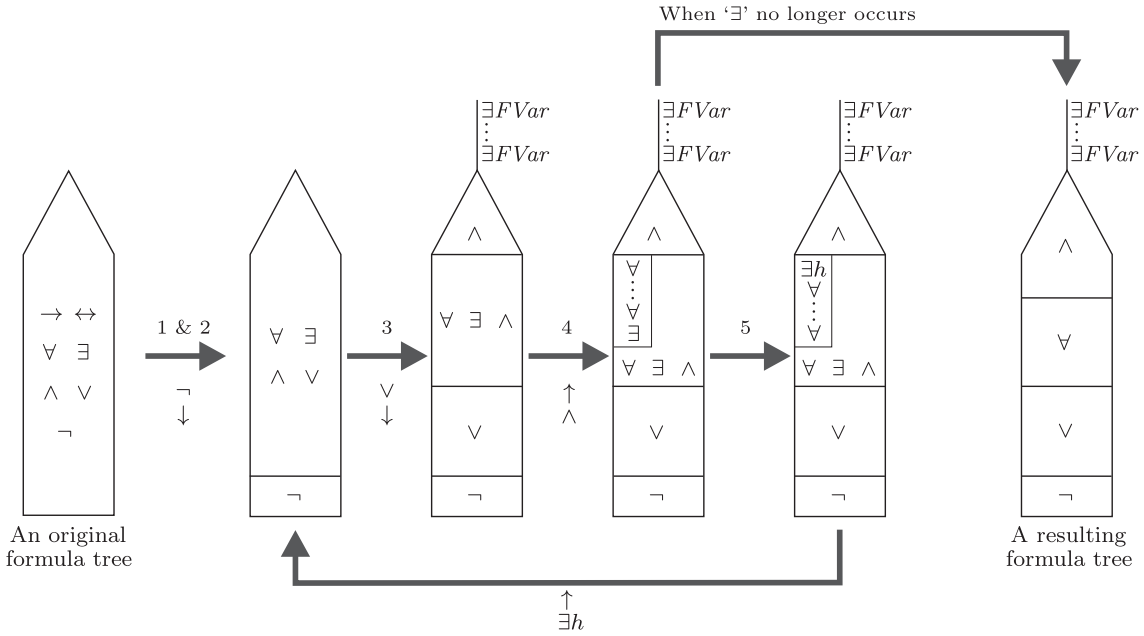

FIGURE 1. An overview of the decomposition procedure

1. *Preparation:*
   (a) Convert $\rightarrow$ and $\leftrightarrow$ equivalently into $\neg$, $\wedge$, and $\vee$, using the following logical equivalences:
   $$\begin{aligned} \beta \rightarrow \gamma &\equiv \neg\beta \vee \gamma \\ \beta \leftrightarrow \gamma &\equiv (\neg\beta \vee \gamma) \wedge (\neg\gamma \vee \beta) \end{aligned}$$
   (b) Rename quantified variables so that for any two occurrences of quantifications Qv and $Q'w$, where $v, w \in \mathit{Var}$, $v \neq w$.
2. *Move $\neg$ inwards:* Move $\neg$ inwards equivalently until each occurrence of $\neg$ immediately precedes an atom, using the following logical equivalences:
   $$\begin{aligned} \neg(\neg\beta) &\equiv \beta \\ \neg(\beta \wedge \gamma) &\equiv \neg\beta \vee \neg\gamma \\ \neg(\beta \vee \gamma) &\equiv \neg\beta \wedge \neg\gamma \\ \neg\forall x : \alpha &\equiv \exists x : \neg\alpha \\ \neg\exists x : \alpha &\equiv \forall x : \neg\alpha \end{aligned}$$
3. *Move down $\vee$-vertices:* Repeatedly move down $\vee$-vertices in the current state of $\mathbf{T}$ through $\exists\mathit{Var}$-vertices, $\forall\mathit{Var}$-vertices, and $\wedge$-vertices as far as possible using the following logical equivalences:
   $$\begin{aligned} (\exists x : \beta) \vee \gamma &\equiv \exists x : (\beta \vee \gamma) \\ (\forall x : \beta) \vee \gamma &\equiv \forall x : (\beta \vee \gamma) \\ (\beta \wedge \gamma) \vee \delta &\equiv (\beta \vee \delta) \wedge (\gamma \vee \delta) \end{aligned}$$
4. *Move up $\wedge$-vertices:* Repeatedly move up $\wedge$-vertices in the current state of $\mathbf{T}$ through $\forall\mathit{Var}$-vertices as far as possible using the following logical equivalence:
   $$\forall x : (\beta \wedge \gamma) \equiv (\forall x : \beta) \wedge (\forall x : \gamma)$$
5. If $\mathbf{T}$ includes an $\exists\mathit{Var}$-vertex, then:
   (a) *Skolemization:* In $\mathbf{T}$, select a subformula
   $$\forall x_1, \ldots, \forall x_n, \exists y : \beta,$$

where $n \geq 0$, such that there is no further universal quantification over this subformula in **T**. Transform this subformula into

$$\exists h, \forall x_1, \ldots, \forall x_n, \forall y : (\beta \vee \neg \mathit{func}(h, x_1, \ldots, x_n, y)),$$

where $h \in \mathit{FVar}$ such that h has not been used so far.

(b) *Move up an $\exists \mathit{FVar}$-vertex:* Repeatedly move up the new $\exists \mathit{FVar}$-vertex (introduced at Step 5(a)) through $\wedge$-vertices as far as possible using the following logical equivalence:

$$(\exists \mathit{FVar} : \beta) \wedge \gamma \quad \equiv \quad \exists \mathit{FVar} : (\beta \wedge \gamma)$$

(c) Go to Step 3.

6. Stop with the formula represented by the current state of **T** as the output formula.

5.2. **Model preservation of the decomposition algorithm.** Each step of the algorithm is an ET step in the space of $\mathcal{L}_2$:

- Except for the Skolemization step (Step 5(a)), a formula with α as a subformula is transformed based on a logical equivalence of the form $\alpha \equiv \beta$ into a new formula that is obtained by replacing α with β.
- Subformula replacement by the Skolemization step (Step 5(a)) is also meaning-preserving.

Since all steps are ET steps, it is obvious that the algorithm preserves the logical meaning of the input formula.

Theorem 5.1. *Given a formula α in* INI *as input, if the algorithm in Section 5.1 yields an output formula β in* FIN*, then α and β have the same logical meaning, i.e., $\mathit{Models}(\alpha) = \mathit{Models}(\beta)$.*

5.3. **Tax-cut example.** The background knowledge of the original "Tax-cut" problem in [24] contains the statement "Peter has a child, who is someone's mother", which is represented by the first-order formula

$$\exists x : (\mathit{hasChild}(\mathit{Peter}, x) \wedge (\exists y : \mathit{motherOf}(x, y))).$$

This formula is transformed by the proposed conversion algorithm as shown in Table 1. From the last formula in the table, we obtain the following two clauses:

- $\mathit{hasChild}(\mathit{Peter}, x) \leftarrow \mathit{func}(h_1, x)$, and
- $\mathit{motherOf}(x, y) \leftarrow \mathit{func}(h_1, x), \mathit{func}(h_2, x, y)$,

where h_1 and h_2 are function variables that are introduced in the conversion process.

5.4. **Solving QA problems by equivalent transformation.** Using MPD on $\mathcal{L}_2$ and $\mathrm{ECLS_F}$, a transformation scheme for solving a QA problem $\langle K, q \rangle$ is obtained.

1) $\langle K, q \rangle$ is transformed into a first-order formula $K \wedge Q$ as described in Section 2.2.
2) The first-order formula $K \wedge Q$ is converted by MPD into a formula Q' in $\mathcal{L}_2$.
3) The formula Q' is converted into a set Cs_1 of extended clauses in $\mathrm{ECLS_F}$.
4) The clause set Cs_1 is successively transformed into a simpler but logically equivalent set Cs_n of extended clauses in $\mathrm{ECLS_F}$.
5) The answer to the problem is determined by $\mathit{ans}(Cs_n, \varphi)$, where ans is a partial mapping, called an answer mapping, that gives the correct answer to a problem in its domain at a small computation cost.

By the above transformation scheme, the QA problem $\langle K, q \rangle$ is transformed into a problem $\langle Cs_1, \varphi \rangle$, which is further transformed into a problem $\langle Cs_n, \varphi \rangle$, the answer to which is $\varphi\,(\bigcap \mathit{Models}(Cs_n))$. We give a sufficient condition for correctness of the transformation scheme by the following theorem.

TABLE 1. Application of the conversion algorithm

Step	Transformation	Formula
		$\exists x : (\mathit{hasChild}(\mathit{Peter}, x) \wedge (\exists y : \mathit{motherOf}(x, y)))$
5(a)	Skolemization	$\exists h_1 : (\forall x : ((\mathit{hasChild}(\mathit{Peter}, x) \wedge (\exists y : \mathit{motherOf}(x, y))) \vee \neg \mathit{func}(h_1, x)))$
3	Move $\vee$ down	$\exists h_1 : (\forall x : ((\mathit{hasChild}(\mathit{Peter}, x) \vee \neg \mathit{func}(h_1, x)) \wedge ((\exists y : \mathit{motherOf}(x, y)) \vee \neg \mathit{func}(h_1, x))))$
3	Move $\vee$ down	$\exists h_1 : (\forall x : ((\mathit{hasChild}(\mathit{Peter}, x) \vee \neg \mathit{func}(h_1, x)) \wedge (\exists y : (\mathit{motherOf}(x, y) \vee \neg \mathit{func}(h_1, x)))))$
4	Move $\wedge$ up	$\exists h_1 : ((\forall x : (\mathit{hasChild}(\mathit{Peter}, x) \vee \neg \mathit{func}(h_1, x))) \wedge (\forall x : (\exists y : (\mathit{motherOf}(x, y) \vee \neg \mathit{func}(h_1, x)))))$
5(a)	Skolemization	$\exists h_1 : ((\forall x : (\mathit{hasChild}(\mathit{Peter}, x) \vee \neg \mathit{func}(h_1, x))) \wedge (\exists h_2 : (\forall x : (\forall y : ((\mathit{motherOf}(x, y) \vee \neg \mathit{func}(h_1, x)) \vee \neg \mathit{func}(h_2, x, y))))))$
5(b)	Move $\exists h_2$ upward	$\exists h_1 : (\exists h_2 : (\forall x : (\mathit{hasChild}(\mathit{Peter}, x) \vee \neg \mathit{func}(h_1, x))) \wedge (\forall x : (\forall y : ((\mathit{motherOf}(x, y) \vee \neg \mathit{func}(h_1, x)) \vee \neg \mathit{func}(h_2, x, y)))))$

Theorem 5.2. *If there is a sequence of problems* $\langle Cs_1, \varphi\rangle, \langle Cs_2, \varphi\rangle, \ldots, \langle Cs_n, \varphi\rangle$ *such that*

- $Cond_1 : \varphi(\bigcap Models(K \wedge Q)) = \varphi(\bigcap Models(Cs_1))$,
- $Cond_2 : \varphi(\bigcap Models(Cs_i)) = \varphi(\bigcap Models(Cs_{i+1}))$ *for any* $i \in \{1, 2, \ldots, n-1\}$, *and*
- $Cond_3 : \langle Cs_n, \varphi\rangle \in dom(ans)$,

then $\varphi(\bigcap Models(K \wedge Q)) = ans(Cs_n, \varphi)$.

Proof: By the first and the second conditions,

$$\varphi\left(\bigcap Models(K \wedge Q)\right) = \varphi\left(\bigcap Models(Cs_1)\right) = \cdots = \varphi\left(\bigcap Models(Cs_n)\right).$$

Since ans is an answer mapping and $\langle Cs_n, \varphi\rangle \in dom(ans)$, it holds that

$$\varphi\left(\bigcap Models(Cs_n)\right) = ans(Cs_n, \varphi).$$

It follows that $\varphi\left(\bigcap Models(K \wedge Q)\right) = ans(Cs_n, \varphi)$. □

The transformation scheme is called the *ET-based problem-solving method*, or simply the *ET-based method*, since it takes the meaning-preserving strategy in the sense that it satisfies $Cond_1$ and $Cond_2$ in Theorem 5.2. We use MPD since applying MPD to $K \wedge Q$ to obtain Cs_1 is a sufficient condition for $Cond_1$. MPD is necessary for the guarantee of correctness of computation in solutions of logical problems including proof problems and QA problems. As shown in Section 3, the "Tax-cut" problem can be converted correctly by MPD satisfying $Cond_1$ into a formula Q', to which further transformation satisfying $Cond_2$ can be applied in the space of the power set of $\mathrm{ECLS_F}$ in order to obtain the correct answer. On the other hand, CSD fails to satisfy $Cond_1$ and no further transformation satisfying $Cond_2$ is useful to reach the correct answer.

6. **Conclusions.** $\mathcal{L}_1$ is the set of all first-order formulas, while $\mathcal{L}_2$ is a new formula space obtained by extending first-order logic with function variables. An algorithm for transforming a formula in $\mathcal{L}_1$ into an equivalent ECNF in $\mathcal{L}_2$ is proposed. For traditional

transformation of first-order formulas into CNFs, it is well-known that the conventional Skolemization-based decomposition (CSD) is not meaning-preserving, i.e., $\exists E \in \mathcal{L}_1 : \mathit{Models}(\mathrm{CSD}(E)) \neq \mathit{Models}(E)$. Meaning-preserving decomposition (MPD) is proposed in this paper such that

- $\mathrm{MPD}(\mathcal{L}_1) \subseteq \mathcal{L}_2$, and
- $\forall E \in \mathcal{L}_1 : \mathit{Models}(\mathrm{MPD}(E)) = \mathit{Models}(E)$.

For solving many kinds of logical problems, including QA problems, MPD can be used to transform a first-order formula into a set of extended clauses in $\mathrm{ECLS_F}$, which is further transformed equivalently in the space of the power set of $\mathrm{ECLS_F}$. MPD, which is based on meaning-preserving Skolemization, is of fundamental importance to guarantee the correctness of logical problem solving. A general method of logical problem solving will be established on the basis of MPD together with the ET-method, avoiding the representational and computational limitation on first-order logic. By the research of control to search for solution paths in the extended space $\mathrm{ECLS_F}$, more logical problems will be solved practically, compared to the logical computation on first-order logic.

Acknowledgment. This work was partially supported by (i) JSPS KAKENHI Grant Numbers 25280078 and 26540110, (ii) the Center of Excellence in Intelligent Informatics, Speech and Language Technology and Service Innovation (CILS), Thammasat University, and (iii) the Intelligent Informatics and Service Innovation (IISI) Center, Sirindhorn International Institute of Technology (SIIT), Thammasat University. The authors also gratefully acknowledge the helpful comments and suggestions from the reviewers.

REFERENCES

[1] J. A. Robinson, A machine-oriented logic based on the resolution principle, *Journal of the ACM*, vol.12, pp.23-41, 1965.
[2] M. Fitting, *First-Order Logic and Automated Theorem Proving*, 2nd Edition, Springer-Verlag, 1996.
[3] C. Walther, A mechanical solution of Schubert's steamroller by many-sorted resolution, *Artificial Intelligence*, vol.26, no.2, pp.217-224, 1985.
[4] F. J. Pelletier, Seventy-five problems for testing automatic theorem provers, *Journal of Automated Reasoning*, vol.2, no.2, pp.191-216, 1986.
[5] M. Stickel, Schubert's steamroller problem: Formulations and solution, *Journal of Automated Reasoning*, vol.2, no.2, pp.89-104, 1986.
[6] T. C. Wang and W. W. Bledsoe, Hierarchical deduction, *Journal of Automated Deduction*, vol.3, no.1, pp.35-77, 1987.
[7] R. A. Kowalski, Predicate logic as a programming language, *Proc. of the 6th IFIP Congress 1974*, Stockholm, Sweden, pp.569-574, 1974.
[8] K. Doets, *From Logic to Logic Programming*, The MIT Press, 1994.
[9] R. A. Kowalski, Algorithm = logic + control, *Communications of the ACM*, vol.22, pp.424-435, 1979.
[10] J. W. Lloyd, *Foundations of Logic Programming*, 2nd Edition, Springer-Verlag, 1987.
[11] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi and P. F. Patel-Schneider, *The Description Logic Handbook*, 2nd Edition, Cambridge University Press, 2007.
[12] S. Tessaris, *Questions and Answers: Reasoning and Querying in Description Logic*, Ph.D. Thesis, Department of Computer Science, The University of Manchester, UK, 2001.
[13] J. Minker, *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann Publishers Inc., 1988.
[14] K. Akama and E. Nantajeewarawat, Function-variable elimination and its limitations, *Proc. of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, KEOD, Lisbon, Portugal, vol.2, pp.212-222, 2015.
[15] K. Akama and E. Nantajeewarawat, Model-intersection problems with existentially quantified function variables: Formalization and a solution schema, *Proc. of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, KEOD, Porto, Portugal, vol.2, pp.52-63, 2016.

[16] C. L. Chang and R. C. T. Lee, *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, 1973.

[17] K. Akama and E. Nantajeewarawat, Unfolding existentially quantified sets of extended clauses, *Proc. of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, KEOD, Porto, Portugal, vol.2, pp.96-103, 2016.

[18] K. Akama, E. Nantajeewarawat and T. Akama, Computation control by prioritized ET rules, *Proc. of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, KEOD, Seville, Spain, vol.2, pp.84-95, 2018.

[19] K. Akama, E. Nantajeewarawat and T. Akama, Logical problem solving framework, *Proc. of the 11th Asian Conference on Intelligent Information and Database Systems*, Yogyakarta, Indonesia, pp.28-40, 2019.

[20] R. Manthey and F. Bry, SATCHMO: A theorem prover implemented in Prolog, *Proc. of the 9th International Conference on Automated Deduction*, Argonne, IL, pp.415-434, 1988.

[21] M. Gelfond and V. Lifschitz, The stable model semantics for logic programming, *Proc. of International Logic Programming Conference and Symposium*, pp.1070-1080, 1988.

[22] M. Gelfond and V. Lifschitz, Classical negation in logic programs and disjunctive databases, *New Generation Computing*, vol.9, pp.365-386, 1991.

[23] K. Akama and E. Nantajeewarawat, Solving query-answering problems with constraints for function variables, *Proc. of the 10th Asian Conference on Intelligent Information and Database Systems*, Dong Hoi City, Vietnam, pp.36-47, 2018.

[24] B. Motik, U. Sattler and R. Studer, Query answering for OWL-DL with rules, *Journal of Web Semantics*, vol.3, no.1, pp.41-60, 2005.