Conjunction-Based Clauses for Equivalent Transformation of Query-Answering Problems

Kiyoshi Akama and Ekawit Nantajeewarawat

Abstract—This paper introduces a class of conjunction-based clauses with function variables and their semantics, with an aim to provide a larger problem-transformation space that seamlessly supports both top-down computation and bottom-up computation. A representative set of the collection of all models of a set of conjunction-based clauses is formulated. Two types of equivalent transformation on conjunction-based clauses, i.e., unfolding and forwarding, are presented and their application is illustrated. The presented work provides a foundation for constructing a correct method for solving query-answering problems.

Index Terms—Query-answering problem, equivalent transformation, conjunction-based clause, representative set, forwarding transformation.

I. INTRODUCTION

A query-answering (QA) problem is concerned with finding the set of all ground instances of a given query atom that are logical consequences of a given logical formula. Equivalent transformation (ET) of formulas is essential and very useful for solving many kinds of logical problems [1], 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 combination of equivalent transformations. Many kinds of correct algorithms for solving logical problems can be devised based on the ET principle.

Meaning-preserving Skolemization [2] necessitates incorporation of function variables. This paper introduces a class of extended clauses, called conjunction-based clauses, which may contain occurrences of function variables, and establishes their semantics. A representative set of the collection of all models of a set of conjunction-based clauses is formulated, based on which preservation of the intersection of all these models can be discussed. Two types of transformation on conjunction-based clauses, i.e., unfolding transformation and forwarding transformation, are presented. Transformation of the first type corresponds to top-down (goal-directed) computation, while that of the second type can be naturally regarded as bottom-up computation. Application of them to simplification of a QA problem is illustrated.

To begin with, Section 2 formulates a class of QA problems, describes a general scheme for solving them using ET, and recalls the class of extended clauses introduced in [2]. Section 3 formulates conjunction-based clauses and defines their semantics. Section 4 defines a representative set of the collection of all models of a set of conjunction-based clauses. Section 5 presents unfolding transformation and forwarding transformation on conjunction-based clauses. Section 6 illustrates their application. Section 7 provides concluding remarks.

II. SOLVING QUERY-ANSWERING PROBLEMS BY EQUIVALENT TRANSFORMATION

A. Query-Answering (QA) Problems

A query-answering problem (QA problem) is a pair \((K, a)\), where \(K\) is a logical formula and \(a\) is an atomic formula (atom). The answer to a QA problem \((K, a)\), denoted by \(\text{ans}(K, a)\), is defined as the set of all ground instances of \(a\) 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 [6]. In the class of QA problems discussed in [8], \(K\) is a conjunction of axioms and assertions in Description Logics [3]. Recently, QA problems have gained wide attention, owing partly to emerging applications in systems involving integration between formal ontological background knowledge and instance-level rule-oriented components, e.g., interaction between Description Logics and Horn rules [5, 7] in the Semantic Web’s ontology-based rule layer.

B. Solving QA Problems by Equivalent Transformation

Using the set of all models of \(K\), denoted by \(\text{Models}(K)\), the answer to a QA problem \((K, a)\) can be equivalently represented as

\[
\text{ans}(K, a) = (\forall \text{Models}(K)) \cap \text{rep}(a),
\]

where \(\forall \text{Models}(K)\) is the intersection of all models of \(K\) and \(\text{rep}(a)\) is the set of all ground instances of \(a\). Calculating \(\forall \text{Models}(K)\) directly may require high computational cost. To reduce the cost, \(K\) is transformed into a simplified formula \(K’\) such that \((\forall \text{Models}(K)) \cap \text{rep}(a)\) is preserved and \((\forall \text{Models}(K’)) \cap \text{rep}(a)\) can be determined at a low cost.

By meaning-preserving Skolemization [2] and moving constraint atoms from left sides to right sides, the logical formula \(K\) is converted into a set \(C_s\) of extended clauses, each of which takes the form

\[
a_1, \ldots, a_n \leftarrow b_1, \ldots, b_p, f_1, \ldots, f_p,
\]

where \(a_1, \ldots, a_n\) are usual atoms, each of \(b_1, \ldots, b_p\) is a usual atom or a constraint atom, and \(f_1, \ldots, f_p\) are func-atoms, which are introduced as follows: Given any \(n\)-ary function constant

Manuscript received March 11, 2012; revised May 2, 2012.

Kiyoshi Akama is with the Information Initiative Center, Hokkaido University, Japan (e-mail: akama@iic.hokudai.ac.jp).

Ekawit Nantajeewarawat is with the Sirindhorn International Institute of Technology, Thammasat University, Thailand.
or n-ary function variable \( f \), an expression
\[
func(f, t_1, \ldots, t_n, t_{n+1}).
\]
where the \( t_i \) are usual terms, is considered as an atom of a new type, called a \( func \)-atom. When \( f \) is a function constant and the \( t_i \) are all ground, the truth value of this atom is evaluated to be true iff \( f(t_1, \ldots, t_n) = t_{n+1} \). Let ECL denote the set of all extended clauses.

Given \( Cs \subseteq ECL \), the set of all models of \( Cs \) is denoted by \( Models(Cs) \). A QA problem \( \langle Cs, a \rangle \) such that \( Cs \subseteq ECL \) is called a QA problem on ECL.

III. CONJUNCTION-BASED CLAUSES AND CONVERSION FROM EXTENDED CLAUSES

An atom conjunction is a formula of the form \([ a_1, \ldots, a_m ]\), where \( a_1, \ldots, a_m \) are usual atoms. A conjunction-based clause \( C \) is a formula of the form
\[
c_1, \ldots, c_m \leftarrow b_1, \ldots, b_n, f_1, \ldots, f_p,
\]
where \( c_1, \ldots, c_m \) are atom conjunctions, each of \( b_1, \ldots, b_n \) is a usual atom or a constraint atom, and \( f_1, \ldots, f_p \) are \( func \)-atoms. The sets \( \{ c_1, \ldots, c_m \} \) and \( \{ b_1, \ldots, b_n, f_1, \ldots, f_p \} \) are called the left-hand side and the right-hand side, respectively, of \( C \), denoted by \( lhs(C) \) and \( rhs(C) \), respectively. When \( m = 1 \), \( C \) is called a conjunction-based definite clause, \( c_1 \) is called the head of \( C \), denoted by head\((C)\), and \( rhs(C) \) is also called the body of \( C \), denoted by body\((C)\). When the conjunction-based clause \( C \) above contains no usual variable and no function variable, it determines a formula \( L(C) \), given by
\[
L(C) = (conj(c_1) \lor \ldots \lor conj(c_m)) \lor \neg b_1 \lor \ldots \lor \neg b_n \lor \neg f_1 \lor \ldots \lor \neg f_p,
\]
where for any \( i \in \{1, \ldots, m\} \), if \( c_i = [a_1, \ldots, a_q] \), then \( conj(c_i) \) denotes \( (a_1 \land \ldots \land a_q) \).

Let CBC be the set of all conjunction-based clauses. Let FVar be the set of all function variables, FCon the set of all function constants, and Map\((FVar, FCon)\) the set of all functions from FVar to FCon. Given \( \sigma \in Map(FVar, FCon) \) and \( R \subseteq CBC \), let inst\((\sigma, R)\) be the set of conjunction-based clauses obtained from \( R \) by instantiating all function variables appearing in it into function constants using \( \sigma \). A set \( G \) of ground usual atoms is a model of a set \( R \subseteq CBC \) iff there exist \( \sigma \in Map(FVar, FCon) \) such that for any \( C \in inst(\sigma, R) \) and any ground substitution \( \theta \) for all usual variables occurring in \( C \), \( L(C|\theta) \) is true with respect to \( G \). The set \( \{ C \in CBC \} \) is denoted by \( Models(R) \).

Theorem 1. Let \( Cs \) be a set of extended clauses. Let \( R \) be the set of conjunction-based clauses obtained from \( Cs \) by converting each clause \( \langle a_1, \ldots, a_n \leftarrow b_1, \ldots, b_n, f_1, \ldots, f_p \rangle \in Cs \) into the conjunction-based clause \( \langle [a_1], \ldots, [a_n] \leftarrow b_1, \ldots, b_n, f_1, \ldots, f_p \rangle \). Then \( Models(Cs) = Models(R) \).

IV. A REPRESENTATIVE SET FOR SOLVING QA PROBLEMS

Next, the notion of a representative set of a collection of sets is introduced. The intersection of a given collection of sets can be determined in terms of the intersection of sets in its representative set (Theorem 2). Given a set \( R \) of conjunction-based clauses, a set collection, \( MM(R) \), is defined, with an important property being that \( MM(R) \) is a representative set of the set of all models of \( R \) (Theorem 3). Consequently, the answer to a QA problem concerning \( R \) can be computed through \( MM(R) \).

A. Representative Sets

A representative set is defined below:

Definition 1. Let \( G \) be a set and \( M_1, M_2 \subseteq 2^G \). \( M_1 \) is a representative set of \( M_2 \) iff \( M_1 \subseteq M_2 \) and for any \( m_2 \in M_2 \), there exists \( m_1 \in M_1 \) such that \( m_2 \supseteq m_1 \).

Theorem 2. Let \( G \) be a set. For any \( M_1, M_2 \subseteq 2^G \), if \( M_1 \) is a representative set of \( M_2 \), then \( \cap M_1 = \cap M_2 \).

B. Representative Set for All Models of a Conjunction-Based-Clause Set

Given a set \( R \subseteq CBC \), \( MM(R) \) is defined below. The following notations are used:

- Let \( CBC_{atv} \) be the set of all conjunction-based clauses with no occurrence of any function variable, GCBC the set of all conjunction-based clauses that consist only of ground usual atoms, and GAC the set of all ground atom conjunctions.
- Given \( R \subseteq CBC_{atv} \), let \( ginst(R) \) be defined as a subset of GCBC as follows:
  1) Let \( R_1 \) be the set of ground conjunction-based clauses obtained from \( R \) by \( R_1 = \{ C \in R \} \) & \( \emptyset \) is a ground substitution for all usual variables occurring in \( C \).
  2) Let \( R_2 \) be the set of ground conjunction-based clauses obtained from \( R_1 \) by removing each conjunction-based clause whose right-hand side contains at least one false constraint atom or at least one false \( func \)-atom.
  3) Then let \( ginst(R) \) be the set of ground conjunction-based clauses obtained from \( R_2 \) by removing all true constraint atoms and all true \( func \)-atoms from the right-hand side of each conjunction-based clause in \( R_2 \).

- Let \( SEL \) be the set of all mappings from GCBC to \( GAC \lor \{ \bot \} \) such that for any \( sel \in SEL \) and any \( C \in GCBC \), the following conditions are satisfied:
  1) If \( lhs(C) = \emptyset \), then \( sel(C) = \bot \).
  2) If \( lhs(C) \neq \emptyset \), then \( sel(C) \in lhs(C) \).

- Let \( GCBDC \) be the set consisting of every conjunction-based definite clause whose body contains only ground usual atoms and whose head is either a ground atom conjunction or \( \bot \).

- Given a mapping \( sel \in SEL \) and \( R \subseteq GCBDC \), let \( edc(sel, R) \) be defined as a subset of GCBDC by
\[
edc(sel, R) = \{ edc(sel, C) \mid C \in R \},
\]
where for each conjunction-based clause \( C \in R \), \( edc(sel, C) \) is the conjunction-based definite clause obtained from \( C \) as follows:


6
1) \text{head}(edc}(sel, C) = sel(C)
2) \text{body}(edc}(sel, C) = rhs(C)

- Given \( C = \{a_1, \ldots, a_n \leftarrow b_1, \ldots, b_m\} \in \text{GCBDC} \), let \( dc(C) = \{a_i \leftarrow b_j \mid 1 \leq i \leq n, 1 \leq j \leq m\} \). Given \( D \subseteq \text{GCBDC} \), let \( dc(D) = \bigcup_{C \subseteq D} dc(C) \).

\textbf{Definition 2.} Let \( R \subseteq \text{CBC} \). A collection \( \text{MM}(R) \) of ground-atoms set is defined by

\[ \text{MM}(R) = \{M(D) \mid (\sigma \in \text{Map}(\text{FVar}, \text{FCon})) \& (sel \in \text{SEL}) \&
\]
\[ (D = dc(edc(sel, \text{ginst}(\text{inst}(\sigma, R)))) \} \& (\perp \notin M(D)) \}, \]

where for any set \( D \) of definite clauses, \( M(D) \) denotes the minimal model of \( D \).

\textbf{Theorem 3.} For any set \( R \) of conjunction-based clauses, \( \text{MM}(R) \) is a representative set of \( \text{Models}(R) \).

V. \textbf{EQUIVALENT TRANSFORMATION OF QA PROBLEMS}

A QA problem \( \langle R, a \rangle \) such that \( R \subseteq \text{CBC} \) is called a QA problem on CBC. Given a QA problem \( \langle R, a \rangle \) on CBC, \( R \) may be further transformed equivalently in the CBC space into another subset of CBC for problem simplification. Unfolding, forwarding, and other transformation rules may be used.

\textbf{A. Unfolding Transformation}

- Given a set \( A \) of atoms, let \( \text{Rep}(A) = \{a \theta \mid (a \in A) \& \theta \text{ is a substitution for usual variables}\} \). Let \( \langle R, a \rangle \) be a QA problem on CBC. Assume that:
  1) \( A_p \) is a set of atoms such that \( a \in \text{Rep}(A_p) \) and \( A_p \) is a set of atoms such that \( \text{Rep}(A_p) \cap \text{Rep}(A) = \emptyset \).
  2) \( D \) is a set of conjunction-based definite clauses in \( R \) that satisfies the following conditions:
     - For any conjunction-based definite clause \( C \in D \), \( \text{head}(C) \) contains only one atom and this atom belongs to \( \text{Rep}(A_p) \).
     - For any conjunction-based clause \( C' \in R - D \), each atom occurring in \( \text{lhs}(C') \) belongs to \( \text{Rep}(A_p) \).
  1) \( \text{occ} \) is an occurrence of an atom \( b \) in the right-hand side of a conjunction-based clause \( C \) in \( R - D \) such that \( b \in \text{Rep}(A_p) \).

\[ \text{UNFOLD}(R, D, \text{occ}) \] is the set

\[ \{R - \{C\} \cup (\cup_{\text{unfolding}(C, C', b)} \mid C' \in D)\} \],

where for each \( C' \in D, \text{unfolding}(C, C', b) \) is defined as follows, assuming that \( \text{head}(C') = \{b'\} \) and \( \rho \) is a renaming substitution for usual variables such that \( C \) and \( C'\rho \) have no usual variable in common:

- If \( b \) and \( b'\rho \) are not unifiable, then \( \text{unfolding}(C, C', b) = \emptyset \).
- If they are unifiable, then \( \text{unfolding}(C, C', b) = \{C''\} \), where \( C'' \) is the conjunction-based clause obtained from \( C \) and \( C'\rho \) as follows, assuming that \( \theta \) is the most general unifier of \( b \) and \( b'\rho \):

- \( \text{lhs}(C'') = \text{lhs}(C)\theta \).

- \( \text{rhs}(C'') = \text{rhs}(C) - \{b\}\theta \cup \text{rhs}(C'\rho)\theta \).

Then \( \text{MM}(R) = \text{MM} \{\text{UNFOLD}(R, D, \text{occ})\} \), and consequently, by Theorems 2 and 3, \( \text{MM}(\text{UNFOLD}(R, D, \text{occ})) \cap \text{rep}(a) = (\text{MM}(\text{UNFOLD}(R, D, \text{occ})) \cap \text{rep}(a)) \).

B. \textbf{Forwarding Transformation}

Let \( e \) be an atom conjunction \( \{a_1, \ldots, a_n\} \) and \( c' \) an atom conjunction \( \{b_1, \ldots, b_m\} \). Then let \( e \oplus c' \) denote the atom conjunction \( \{a_1, \ldots, a_n, b_1, \ldots, b_m\} \). Assume that \( R \) is a set of range-restricted conjunction-based clauses, i.e., for each conjunction-based clause \( C \in R \), each usual variable that occurs in \( \text{lhs}(C) \) also occurs in \( \text{rh}(C) \).

\textbf{Fwd-1:} Let \( c \) and \( d \) be atom conjunctions. Assume that

1) \( R = \{C_1, C_2\} \cup R_{\text{rest}}, \) where \( C_1 = (e \leftarrow) \) and \( C_2 = (d \leftarrow); \)
2) \( C' = (e \oplus d \leftarrow). \)

Then \( \text{MM}(R) = \text{MM}(\{C', C_1, C_2\} \cup R_{\text{rest}}) \), and it follows from Theorems 2 and 3 that for any usual atom \( a \), \( (\text{MM}(\text{Models}(R)) \cap \text{rep}(a) = (\text{MM}(\text{Models}(\{C', C_1, C_2\} \cup R_{\text{rest}})) \cap \text{rep}(a). \)

\textbf{Fwd-2:} Let \( c_1, \ldots, c_n \) and \( d_1, \ldots, d_q \) be atom conjunctions. Let \( e_1, \ldots, e_s \) and \( a_1, \ldots, a_l \) be usual atoms. Assume that

1) \( R = \{C_1, C_2\} \cup R_{\text{rest}}, \) where \( C_1 = (c_1, \ldots, c_n \oplus e_1, \ldots, e_s), \)
2) \( C_2 = (d_1, \ldots, d_q \leftarrow a_1, \ldots, a_l); \)
3) \( \theta \) is a substitution for usual variables such that each atom in \( \{a_1, \ldots, a_l\} \theta \) occurs in \( \{e_1, \ldots, e_s\} \).

Then \( \text{MM}(\text{Models}(\{C', C_1, C_2\} \cup R_{\text{rest}})) \cap \text{rep}(a) = (\text{MM}(\text{Models}(\{C', C_1, C_2\} \cup R_{\text{rest}})) \cap \text{rep}(a). \)

VI. \textbf{Example}

The Oedipus problem, given in [3], is taken as an example. Oedipus killed his father, married his mother Lokaste, and had children with her, among them Polynieikes. Polynieikes also had children, among them Thersandros, and Thersandros is not a patricide. The problem is to find “a person who has a patricide child who has a non-patricide child.” Assuming that “oe,” “io,” “po,” and “th” stand, respectively, for Oedipus, Lokaste, Polynieikes, and Thersandros, this problem is represented as a QA problem \( \langle C_S, \text{prob}(X) \rangle, \) where \( C_S \) consists of the following seven clauses:

- \( C_1: \text{hasChild}(\text{oe}, \text{io}) \leftarrow C_6: \text{hasChild}(\text{po}, \text{io}) \leftarrow C_8: \text{hasChild}(\text{th}, \text{po}) \)
- \( C_2: \text{hasChild}(\text{po}, \text{oe}) \leftarrow C_3: \text{hasChild}(\text{th}, \text{po}) \leftarrow \)
- \( C_5: \text{hasChild}(\text{Z}, \text{Y}) \leftarrow \text{hasChild}(\text{Z}, \text{Y}) \leftarrow \text{hasChild}(\text{Y}, \text{X}) \), \text{pat}(\text{th}) \)

The clause set \( C_S \) is converted into a set \( R \) consisting of the following conjunction-based clauses:
C₁: [hasChild(oe, io)] ← C₂: [hasChild(po, io)]
←
C₃: [hasChild(po, oe)] ← C₄: [hasChild(th, po)]
←
C₅: [pat(oe)] ← C₆: ← pat(th)
C₇: [pat(Z), [prob(X)] ← hasChild(Z, Y), hasChild(Y, X), pat(Y)

The set R is successively transformed as follows:

- By unfolding at hasChild(Z, Y) in C₇ with \( A_p = \{ \text{hasChild}(X, Y) \}, A_q = \{ \text{prob}(X), \text{pat}(X) \}, \) and \( D = \{ C_1, C_2, C_3, C_4 \}, \) we obtain:
  \( C_w: [\text{pat}(oe)], [\text{prob}(X)] ← \text{hasChild}(io, X), \text{pat}(io) \)
  \( C_v: [\text{pat}(po)], [\text{prob}(X)] ← \text{hasChild}(io, X), \text{pat}(io) \)
  \( C_v: [\text{pat}(po)], [\text{prob}(X)] ← \text{hasChild}(io, X), \text{pat}(io) \)
  \( C_v: [\text{pat}(th)], [\text{prob}(X)] ← \text{hasChild}(po, X), \text{pat}(po) \)

- By unfolding at the hasChild-atoms in \( C_w, C_v, C_v, \) and \( C_v \), with \( A_p = \{ \text{hasChild}(X, Y) \}, A_q = \{ \text{prob}(X), \text{pat}(X) \}, \) and \( D = \{ C_1, C_2, C_3, C_4 \}, \) we obtain:
  \( C_v: [\text{pat}(po)], [\text{prob}(io)] ← \text{pat}(oe) \)
  \( C_v: [\text{pat}(th)], [\text{prob}(io)] ← \text{pat}(po) \)
  \( C_v: [\text{pat}(th)], [\text{prob}(oe)] ← \text{pat}(po) \)

- The conjunction-based clauses \( C_1-C_4 \) can then be removed. The current conjunction-based-clause set is \( \{ C_5, C_6, C_7, C_8 \}. \)

- By \textbf{Fwd-2} with \( C_5\) and \( C_6 \), we obtain:
  \( C_{w_0}: [\text{pat}(oe), \text{pat}(po), \text{prob}(io), \text{prob}(oe)] \)
  \( [\text{pat}(oe), \text{prob}(io)] ← \)

- The conjunction-based clauses \( C_6, C_7, C_8 \), and \( C_9 \) can then be removed. The current conjunction-based-clause set is the singleton \( \{ C_m \}. \)

Obviously, \( \cap \text{MM}((C_m)) \cap \text{rep}([\text{prob}(X)]) = \{ [\text{prob}(io)] \}. \) Thus Iokaste is the only answer to this problem.

VII. CONCLUDING REMARKS

Conventional Skolemization imposes restrictions on solving QA problems in the first-order domain. Development of a correct and efficient solver for a large class of QA problems demands meaning-preserving Skolemization, which converts a given first-order formula into a set of extended clauses possibly containing function variables. This paper has proposed a class of conjunction-based clauses with function variables and has established their semantics. This class of formulas forms a space for equivalent transformation that allows a combination of top-down computation through unfolding transformation and bottom-up computation through forwarding transformation. It provides a basis for construction of more general and more efficient QA-problem solvers.

REFERENCES


