# 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 $\langle K, a\rangle$, where $K$ is a logical formula and $a$ is an atomic formula (atom). The *answer* to a QA problem $\langle K, a\rangle$, denoted by $ans(K, a)$, 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 $Models(K)$, the answer to a QA problem $\langle K, a\rangle$ can be equivalently represented as

$$ans(K, a) = (\cap Models(K)) \cap rep(a),$$

where $\cap Models(K)$ is the intersection of all models of $K$ and $rep(a)$ is the set of all ground instances of $a$. Calculating $\cap Models(K)$ directly may require high computational cost. To reduce the cost, $K$ is transformed into a simplified formula $K'$ such that $(\cap Models(K)) \cap rep(a)$ is preserved and $(\cap Models(K')) \cap rep(a)$ 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 $Cs$ of extended clauses, each of which takes the form

$$a_1, \ldots, a_m \leftarrow b_1, \ldots, b_p, \mathbf{f}_1, \ldots, \mathbf{f}_q,$$

where $a_1, \ldots, a_m$ are usual atoms, each of $b_1, \ldots, b_p$ is a usual atom or a constraint atom, and $\mathbf{f}_1, \ldots, \mathbf{f}_q$ 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

$$\mathbf{c}_1, \ldots, \mathbf{c}_m \leftarrow b_1, \ldots, b_n, \mathbf{f}_1, \ldots, \mathbf{f}_p,$$

where $\mathbf{c}_1, \ldots, \mathbf{c}_m$ are atom conjunctions, each of $b_1, \ldots, b_n$ is a usual atom or a constraint atom, and $\mathbf{f}_1, \ldots, \mathbf{f}_p$ are *func*-atoms. The sets $\{\mathbf{c}_1, \ldots, \mathbf{c}_m\}$ and $\{b_1, ..., b_n, \mathbf{f}_1, ..., \mathbf{f}_p\}$ 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*, $\mathbf{c}_1$ 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 $\mathrm{L}(C)$, given by

$$\mathrm{L}(C) = (conj(\mathbf{c}_1) \vee \ldots \vee conj(\mathbf{c}_m) \vee \neg b_1 \vee \ldots \vee \neg b_n \vee \ldots \vee \neg f_1 \vee \ldots \vee \neg f_p),$$

where for any $i \in \{1, \ldots, m\}$, if $\mathbf{c}_i = [a_1, \ldots, a_q]$, then $conj(\mathbf{c}_i)$ denotes $(a_1 \wedge \ldots \wedge a_q)$.

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 exists $\sigma \in Map$(FVar, FCon) such that for any $C \in inst(\sigma, R)$ and any ground substitution $\theta$ for all usual variables occurring in $C$, $\mathrm{L}(C\theta)$ is true with respect to $G$. The set of all models of a set $R \subseteq$ 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 $(a_1, \ldots, a_m \leftarrow b_1, \ldots, b_n, \mathbf{f}_1, \ldots, \mathbf{f}_p) \in Cs$ into the conjunction-based clause $([a_1], \ldots, [a_m] \leftarrow b_1, \ldots, b_n, \mathbf{f}_1, \ldots, \mathbf{f}_p)$. 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, $\mathbf{MM}(R)$, is defined, with an important property being that $\mathbf{MM}(R)$ 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 $\mathbf{MM}(R)$.

### 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 below provides a basis for computing the intersection of the set of all models of a clause set using its representative set.

**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, $\mathbf{MM}(R)$ is defined below. The following notations are used:

- Let $\mathrm{CBC}_{\mathrm{nfv}}$ 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 $R \subseteq \mathrm{CBC}_{\mathrm{nfv}}$, 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 $R_1 = \{C\theta \mid (C \in R)$ & ($\theta$ 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*-atoms.
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 $\cup \{\perp\}$ such that for any $sel \in$ SEL and any $C \in$ GCBC, the following conditions are satisfied:

1) If $lhs(C) = \emptyset$, then $sel(C) = \perp$.
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 $\perp$.
- Given a mapping $sel \in$ SEL and $R \subseteq$ GCBC, 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:

1) $head(edc(sel, C)) = sel(C)$
2) $body(edc(sel, C)) = rhs(C)$

- Given $C = ([a_1, \ldots, a_m] \leftarrow b_1, \ldots, b_n) \in$ GCBDC, let $dc(C) = \{(a_i \leftarrow b_1, \ldots, b_n) \mid 1 \leq i \leq m\}$. Given $D \subseteq$ GCBDC, let $dc(D) = \cup_{C \in D}\, dc(C)$.

**Definition 2.** Let $R \subseteq$ CBC. A collection **MM**($R$) of ground-atom sets is defined by

**MM**($R$) = {M($D$) | ($\sigma \in Map$(FVar, FCon)) & ($sel \in$ SEL) &

($D = dc(edc(sel, ginst(inst(\sigma, R))))$) & ($\bot \notin$ M($D$))},

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

**Theorem 3.** For any set $R$ of conjunction-based clauses, **MM**($R$) is a representative set of *Models*($R$).

## V. Equivalent Transformation of QA Problems

A QA problem $\langle R, a\rangle$ such that $R \subseteq$ 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.

### A. Unfolding Transformation

Given a set $A$ of atoms, let $Rep(A) = \{a\theta \mid (a \in A)$ & $\theta$ is a substitution for usual variables) }. Let $\langle R, a\rangle$ be a QA problem on CBC. Assume that:

1) $A_q$ is a set of atoms such that $a \in Rep(A_q)$ and $A_p$ is a set of *atoms* such that $Rep(A_p) \cap Rep(A_q) = \emptyset$.
2) $D$ is a set of conjunction-based definite clauses in $R$ that satisfies the following conditions:
   - For any conjunction-based definite clause $C \in D$, $head(C)$ contains only one atom and this atom belongs to $Rep(A_p)$.
   - For any conjunction-based clause $C' \in R - D$, each atom occurring in $lhs(C')$ belongs to $Rep(A_q)$.

1) $occ$ is an occurrence of an atom $b$ in the right-hand side of a *conjunction*-based clause $C$ in $R - D$ such that $b \in Rep(A_p)$.
2) UNFOLD($R$, $D$, $occ$) is *the* set

$$(R - \{C\}) \cup (\cup\{unfold(C, C', b) \mid C' \in D\}),$$

where for each $C' \in D$, $unfold(C, C', b)$ is defined as follows, assuming that $head(C') = [b']$ 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 $unfold(C, C', b) = \emptyset$.
- If they are unifiable, then $unfold(C, C', b) = \{C''\}$, 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 $b'\rho$:
- $lhs(C'') = lhs(C)\theta$.
- $rhs(C'') = (rhs(C) - \{b\})\theta \cup rhs(C'\rho)\theta$.

Then **MM**($R$) = **MM**(UNFOLD($R$, $D$, $occ$)), and consequently, by Theorems 2 and 3, ($\cap$*Models*($R$)) $\cap$ $rep(a)$ = ($\cap$Models(UNFOLD($R$, $D$, $occ$))) $\cap$ $rep(a)$.

### B. Forwarding Transformation

Let **c** be an atom conjunction $[a_1, \ldots, a_m]$ and **c**′ an atom conjunction $[b_1, \ldots, b_n]$. Then let **c** $\oplus$ **c**′ denote the atom conjunction $[a_1, \ldots, a_m, b_1, \ldots, b_n]$. 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 $lhs(C)$ also occurs in $rhs(C)$.

- **Fwd-1**: Let **c** and **d** be atom conjunctions. Assume that

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

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

- **Fwd-2**: Let $\mathbf{c}_1, \ldots, \mathbf{c}_m$ and $\mathbf{d}_1, \ldots, \mathbf{d}_q$ be atom conjunctions. Let $e_1, \ldots, e_n$ and $a_1, \ldots, a_k$ be usual atoms. Assume that

1) $R = \{C_1, C_2\} \cup R_{\text{rest}}$, where
   - $C_1 = (\mathbf{c}_1, \ldots, \mathbf{c}_m, [e_1, \ldots, e_n] \leftarrow)$,
   - $C_2 = (\mathbf{d}_1, \ldots, \mathbf{d}_q \leftarrow a_1, \ldots, a_k)$;
2) $\theta$ is a substitution for usual variables such that each atom in $[a_1, \ldots, a_k]\theta$ occurs in $[e_1, \ldots, e_n]$;
3) $C' = (\mathbf{c}_1, \ldots, \mathbf{c}_m, ([e_1, \ldots, e_n] \oplus \mathbf{d}_1\theta), \ldots, ([e_1, \ldots, e_n] \oplus \mathbf{d}_q\theta) \leftarrow)$.

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

## VI. Example

The Oedipus problem, given in [3], is taken as an example. Oedipus killed his father, married his mother Iokaste, and had children with her, among them Polyneikes. Polyneikes 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, Iokaste, Polyneikes, and Thersandros, this problem is represented as a QA problem $\langle Cs, prob(X)\rangle$, where $Cs$ consists of the following seven clauses:

$C_{\text{I}}$: $hasChild(oe, io) \leftarrow$ $C_{\text{II}}$: $hasChild(po, io) \leftarrow$

$C_{\text{III}}$: $hasChild(po, oe) \leftarrow$ $C_{\text{IV}}$: $hasChild(th, po) \leftarrow$

$C_{\text{V}}$: $pat(oe) \leftarrow$ $C_{\text{VI}}$: $\leftarrow pat(th)$

$C_{\text{VII}}$: $pat(Z), prob(X) \leftarrow hasChild(Z, Y), hasChild(Y, X), pat(Y)$

The clause set $Cs$ is converted into a set $R$ consisting of the following conjunction-based clauses:

$C_1$: [$hasChild(oe, io)$] ← $C_2$: [$hasChild(po, io)$] ←

$C_3$: [$hasChild(po, oe)$] ← $C_4$: [$hasChild(th, po)$] ←

$C_5$: [$pat(oe)$] ← $C_6$: ← $pat(th)$

$C_7$: [$pat(Z)$], [$prob(X)$] ← $hasChild(Z, Y)$, $hasChild(Y, X)$, $pat(Y)$

The set $R$ is successively transformed as follows:

- By unfolding at $hasChild(Z, Y)$ in $C_7$ with $A_p$ = {$hasChild(X, Y)$}, $A_q$ = {$prob(X)$, $pat(X)$}, and $D$ = {$C_1$, $C_2$, $C_3$, $C_4$}, we obtain:

  $C_a$: [$pat(oe)$], [$prob(X)$] ← $hasChild(io, X)$, $pat(io)$

  $C_b$: [$pat(po)$], [$prob(X)$] ← $hasChild(io, X)$, $pat(io)$

  $C_c$: [$pat(po)$], [$prob(X)$] ← $hasChild(oe, X)$, $pat(oe)$

  $C_d$: [$pat(th)$], [$prob(X)$] ← $hasChild(po, X)$, $pat(po)$

- By unfolding at the *hasChild*-atoms in $C_a$, $C_b$, $C_c$, and $C_d$, with $A_p$ = {$hasChild(X, Y)$}, $A_q$ = {$prob(X)$, $pat(X)$}, and $D$ = {$C_1$, $C_2$, $C_3$, $C_4$}, we obtain:

  $C_e$: [$pat(po)$], [$prob(io)$] ← $pat(oe)$

  $C_f$: [$pat(th)$], [$prob(io)$] ← $pat(po)$

  $C_g$: [$pat(th)$], [$prob(oe)$] ← $pat(po)$

- The conjunction-based clauses $C_1$-$C_4$ can then be removed. The current conjunction-based-clause set is {$C_5$, $C_6$, $C_e$, $C_f$, $C_g$}.

- By **Fwd-2** with $C_5$ and $C_e$, we obtain:

  $C_h$: [$pat(oe)$, $pat(po)$], [$pat(oe)$, $prob(io)$] ←

- By **Fwd-2** with $C_h$ and $C_f$, we obtain:

  $C_i$: [$pat(oe)$, $pat(po)$, $pat(th)$], [$pat(oe)$, $pat(po)$, $prob(io)$], [$pat(oe)$, $prob(io)$] ←

- By **Fwd-2** with $C_i$ and $C_6$, we obtain:

  $C_j$: [$pat(oe)$, $pat(po)$, $prob(io)$], [$pat(oe)$, $prob(io)$] ←

- By **Fwd-2** with $C_j$ and $C_g$, we obtain:

  $C_k$: [$pat(oe)$, $pat(po)$, $prob(io)$, $pat(th)$], [$pat(oe)$, $pat(po)$, $prob(io)$, $prob$(oe)],

  [$pat(oe)$, $prob(io)$] ←

- By **Fwd-2** with $C_k$ and $C_6$, we obtain:

  $C_m$: [$pat(oe)$, $pat(po)$, $prob(io)$, $prob(oe)$], [$pat(oe)$, $prob(io)$] ←

- The conjunction-based clauses $C_6$, $C_e$, $C_f$, and $C_g$ can then be removed. The current conjunction-based-clause set is the singleton{$C_m$ }.

Obviously, $\cap$**MM**({$C_m$}) $\cap$ $rep(prob(X))$ = {$prob(io)$}. 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


[1] K. Akama and E. Nantajeewarawat, “Formalization of the Equivalent Transformation Computation Model,” *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 10, no. 3, pp. 245-259, 2006.

[2] K. Akama and E. Nantajeewarawat, “Meaning-Preserving Skolemization,” in *Proc. of the 2011 International Conference on Knowledge Engineering and Ontology Development (KEOD 2011)*, Paris, France, pp. 322-327, 2011.

[3] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, *the Description Logic Handbook*. 2nd edn. Cambridge University Press, 2007.

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

[5] I. Horrocks, P. F. Patel-schneider, S. Bechhofer, and D. Tsarkov, “OWL Rules: A Proposal and Prototype Implementation,” *Journal of Web Semantics*, pp. 23-40, 2005.

[6] J. W. Lloyd. *Foundations of Logic Programming*. 2nd edn. Springer-Verlag, 1987.

[7] B. Motik, U. Sattler, and R. Studer, “Query Answering for OWL-DL with Rules,” *Journal of Web Semantics*, pp. 41-60, 2005.

[8] S. Tessaris, *Questions and Answers: Reasoning and Querying in Description Logic*. PhD Thesis, Department of Computer Science, the University of Manchester, UK, 2001.