MEANING-PRESERVING SKOLEMIZATION

Kiyoshi Akama[1] and Ekawit Nantajeewarawat[2]
[1]*Information Initiative Center, Hokkaido University, Hokkaido, Japan*
[2]*Computer Science, Sirindhorn International Institute of Technology, Thammasat University, Pathumthani, Thailand*

Keywords: Skolemization, Equivalent transformation, Conjunctive normal form, Question-answering problems.

Abstract: Skolemization is a well-known method for removing existential quantifiers from a logical formula. Although it always yields a satisfiability-preserving transformation step, classical Skolemization in general does not preserve the logical meaning of a source formula. We develop in this paper a theory for extending a space of logical 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.

1 INTRODUCTION

Conversion of a given formula into a conjunction of clauses, called a conjunctive normal form (CNF) or a clausal normal form, is a normalization process commonly used in automated reasoning. Such conversion involves removal of existential quantifications by Skolemization (named after Thoralf Albert Skolem), i.e., by replacement of an existentially quantified variable with a Skolem term, which is usually determined by a relevant part of a formula prenex.

Conversion into CNFs is a basic preparation step for automated proof by resolution and factoring. Most theories in logic programming are based on clausal forms. Recently, question-answering problems (QA problems) have gain wide attention. 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. Most research works on solving QA problems are also based on Skolemization, including those in systems involving integration between formal ontological background knowledge and instance-level rule-oriented components, e.g., interaction between Description Logics and Horn rules (Donini et al., 1998; Horrocks et al., 2005; Levy and Rousset, 1998; Motik et al., 2005) in the Semantic Web's ontology-based rule layer.

Skolemization, however, does not preserve the logical meaning of a formula; the formula resulting from Skolemization is not necessarily equivalent to the original one. Only the satisfiability property of a formula is preserved—the resulting formula is equi-satisfiable with the original formula (Chang and Lee, 1973), i.e., it is satisfiable iff the original formula is.

Equivalent Transformation (ET) of formulas is essential and very useful for solving many kinds of logical problems (Akama and Nantajeewarawat, 2006), 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, which yields many kinds of correct algorithms for solving logical problems. Since classical Skolemization does not result in meaning-preserving transformation, it cannot be used in an ET-based problem-solving process.

Our primary objective here is to develop a theory for extending a space of logical formulas by introduction of function variables and a specialization operation on them in such a way that "meaning-preserving" Skolemization can be achieved in an obtained extended space. Fig. 1 gives a pictorial view of our goal. Assume that α is a given first-order formula with occurrences of existential quantifications. As illustrated in the figure, suppose that α is converted into a CNF β by a sequence of transformation steps based on the usual normalization procedure on the space, say $\mathcal{L}_1$, of first-order logic. When classical Skolemization is used in this conversion, α and β are not necessarily logically equivalent and, thus, β does not always serve as an intermediate equivalent formula for fur-

DOI: 10.5220/0003692003220327
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (KEOD-2011), pages 322-327
ISBN: 978-989-8425-80-5

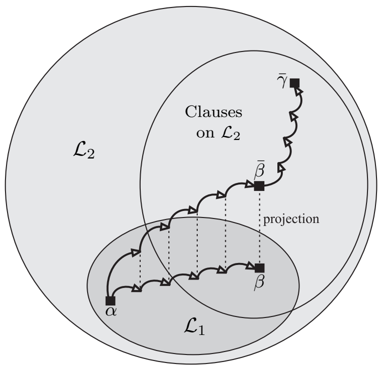

Figure 1: ET-based problem solving with meaning-preserving Skolemization.

ther transformation preserving the logical meaning of α. By contrast, in the expected extended logical structure, referred to as $\mathcal{L}_2$, by using meaning-preserving Skolemization, α is converted into an extended CNF, say β, that is logically equivalent to it. Consequently, α can be further equivalently transformed in the extended space, for example, by using the meaning-preserving transformation path from β to $\overline{\gamma}$ in the figure. It is expected that our meaning-preserving Skolemization framework will provide an important theoretical basis for a large class of automated reasoning tasks.

Section 2 formalizes a class of QA problems and outlines an ET-based method for solving them. Section 3 explains the necessity of meaning-preserving Skolemization and an extension of a logical space. After introducing function constants and function variables, Section 4 formulates an extended logical space and defines the meanings of extended formulas. Section 5 presents an extended conjunctive normal form, called existentially quantified conjunctive normal form (ECNF), along with an algorithm for meaning-preserving conversion of a formula into an ECNF on the extended logical space. Section 6 concludes the paper.

2 QUESTION-ANSWERING PROBLEMS AND ET-BASED SOLUTIONS

To begin with, a question-answering problem is defined. It is followed by a general ET-based solution scheme.

2.1 Question-Answering (QA) Problems

A *question-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 $ans(K,q)$, is defined by

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

i.e., the set of all ground instances of q that follows logically from K. When K consists of only definite clauses, problems in this class are problems that have been discussed in logic programming (Lloyd, 1987). When K is a conjunction of axioms and assertions in Description Logics (Baader et al., 2007), QA problems are usually called *query-answering problems*.

2.2 Solving QA Problems by ET

Using the set of all models of K, denoted by $Model(K)$, the answer to a QA problem $\langle K, q\rangle$, can be equivalently represented as

$$ans(K,q) = (\bigcap Model(K)) \cap rep(q),$$

where $\bigcap Model(K)$ is the intersection of all models of K and $rep(q)$ is the set of all ground instances of q.

Calculating $\bigcap Model(K)$ directly may require high computation cost. To reduce the cost, K is transformed into a simplified formula K' such that all models of K is preserved and $\bigcap Model(K') \cap rep(q)$ can be determined at a low cost. Obviously, if $Model(K) = Model(K')$, then $ans(K,q) = ans(K',q)$.

3 NEED FOR MEANING-PRESERVING SKOLEMIZATION

3.1 Use of Conjunctive Normal Forms

A conjunctive normal form (CNF) is a set of clauses, interpreted as a conjunction. Most important methods for theorem proving deal with logical formulas in CNFs, using basic operations such as unification, resolution, unfolding, and factoring. Based on CNFs, a transformation scheme for solving a QA problem $\langle K, q\rangle$ typically consists of two steps:

1. K is converted into a CNF K'.
2. $\langle K', q\rangle$ is transformed equivalently into $\langle K'', q\rangle$, where K' is also a CNF.

From $\langle K'', q\rangle$, the answer to the problem $\langle K, q\rangle$ is determined by

$$ans(K,q) = (\bigcap Model(K'')) \cap rep(q).$$

3.2 Traditional Skolemization

For traditional transformation of first order formulas into CNFs, all transformations are basically equivalent transformation. They include, for example, the implication law ($p \rightarrow q \equiv \neg p \vee q$), the De morgan's laws ($\neg(p \wedge q) \equiv \neg p \vee \neg q$), etc. It is well-known, however, that traditional Skolemization is not meaning-preserving. 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.

3.3 Introduction of Meaning-Preserving Skolemization

The basic idea of meaning-preserving Skolemization is to use existentially quantified function variables instead of function constants. 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).

3.4 Need for an Extended Space

Formulas (2) and (3) above both contain a function variable and a quantification on that function variable, which are not included in usual first-order formulas. The use of them leads to an extension of the basic concepts for the first-order logic. Let $\mathcal{L}_1$ be the space of all conventional first-order formulas. We need the following extensions:

- 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.
- An extended definition of the truth value of a formula with quantifications on function variables, which determines the semantics of formulas on the extended space.

This requirement raises a question: "How to define the extended space $\mathcal{L}_2$ and the semantics of formulas thereon?" Section 4 provides an answer to this question.

4 AN EXTENDED SPACE FOR SKOLEMIZATION

4.1 Function Constants and Function Variables

A usual function symbol in first-order logic denotes an unevaluated function; it is used for constructing a syntactically new term from existing terms (possibly recursively) without evaluating those existing terms. A different class of functions is used in the extended space $\mathcal{L}_2$. A function in this class is an actual mathematical function; it takes ground terms as input, and associates with them an output ground term. The input ground terms are evaluated for determining the output. We called a function in this class a *function constant*.

In order to clearly separate function constants and function variables from usual functions and usual terms, a new built-in predicate *func* is introduced. Given any n-ary function constant or n-ary function variable $\overline{f}$,

$$\mathit{func}(\overline{f}, t_1, \ldots, t_n, t_{n+1}),$$

where the t_i are usual terms, is considered as an atom of a new type, called a *func-atom*. When $\overline{f}$ is a function constant and the t_i are all ground, the truth value of this atom is evaluated as follows: it is true iff $\overline{f}(t_1, \ldots, t_n) = t_{n+1}$.

Accordingly, function constants and function variables are syntactically differentiated from usual terms. Function constants and function variables appear only as the first arguments of *func*-atoms, while usual terms appear as other arguments of them. Arguments of usual atoms can only be usual terms. By such clear-cut separation, we need not consider unification of usual terms and function variables/function constants. This makes a computation process easier to understand since computation methods similar to those used in the usual first-order logic can be adopted.

The space $\mathcal{L}_1$ is then extended into the space $\mathcal{L}_2$ by inclusion of *func*-atoms and quantifications on function variables. There are two disjoint classes of atoms in $\mathcal{L}_2$:

- *func*-atoms introduced above;
- usual atoms, constructed in the usual way from ordinary predicates and usual terms.

From these atoms, formulas in $\mathcal{L}_2$ are constructed using logical connectives (i.e., $\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$) and quantifications in the usual way, except that in addition to quantifications on usual variables, function variables are also quantified. Like a quantification on a usual variable, a quantification on a function variable v_h is either a universal quantification $\forall v_h$ or an existential quantification $\exists v_h$.

In the following, let *Var* denote the set of all usual variables and *FVar* the set of all function variables. For any expression E, any $v \in Var$ (respectively, any $v_h \in FVar$), and any usual term t (respectively, any function constant f), let $E\{v/t\}$ (respectively, $E\{v_h/f\}$) denote the expression obtained from E by replacing each occurrence of v with t (respectively, each occurrence of v_h with f).

4.2 Interpretations and Models

Let $\mathcal{G}$ be the set of all ground atoms. An *interpretation* is a subset of $\mathcal{G}$. Given an interpretation I, the truth value of a closed formula under I is defined as follows:

1. For any ground atom g, g is true under I iff $g \in I$.
2. For any closed formula α, $\neg\alpha$ is true under I iff α is false under I.
3. For any closed formulas α and β, $\alpha \wedge \beta$ (respectively, $\alpha \vee \beta$, $\alpha \rightarrow \beta$, and $\alpha \leftrightarrow \beta$) is true under I iff α and β are true (respectively, at least one of α and β is true, at least one of $\neg\alpha$ and β is true, and α and β have the same truth value) under I.
4. For any $v \in Var$, a closed formula $\forall v : E$ is true under I iff for any ground term t, $E\{v/t\}$ is true under I.
5. For any $v \in Var$, a closed formula $\exists v : E$ is true under I iff there exists at least one ground term t such that $E\{v/t\}$ is true under I.
6. For any $v_h \in FVar$, a closed formula $\forall v_h : E$ is true under I iff for any function constant f, $E\{v_h/f\}$ is true under I.
7. For any $v_h \in FVar$, a closed formula $\exists v_h : E$ is true under I iff there exists at least one function constant f such that $E\{v_h/f\}$ is true under I.

An interpretation I is a *model* of a closed formula α iff α is true under I.

4.3 A Safe Extension into a New Space

The introduction of function variables necessitates the extension of the original space $\mathcal{L}_1$ into the extended one $\mathcal{L}_2$. A solution path for meaning-preserving Skolemization is:

1. First, transform a given formula α on $\mathcal{L}_1$ into the same formula on $\mathcal{L}_2$.
2. Next, transform α in the space of $\mathcal{L}_2$ into an extended conjunctive normal form.

Obviously, $\mathcal{L}_1$ is a subset of $\mathcal{L}_2$. Moreover, $\mathcal{L}_2$ is a safe extension of $\mathcal{L}_1$, i.e.,

> any formula on $\mathcal{L}_1$ have the same meaning as the same formula on $\mathcal{L}_2$,

the reason being that all formulas on $\mathcal{L}_1$ do not include function variables and function constants, and the definitions of the truth values of closed formulas with quantified function variables under an interpretation do not affect the truth values of formulas on $\mathcal{L}_1$. The first step in the solution path above is thus an equivalent transformation step.

The second step raises another question: "What are extended conjunctive normal forms?" Section 5 provides an answer to this question along with an algorithm for the second step.

5 AN ALGORITHM FOR MEANING-PRESERVING SKOLEMIZATION

Based on the notion of a formula tree, an extended conjunctive normal form, called an *existentially quantified conjunctive normal form* (*ECNF*), is defined. An algorithm for transforming a formula on $\mathcal{L}_1$ into an equivalent ECNF on $\mathcal{L}_2$ is then presented.

5.1 Formula Trees

Given a formula α on $\mathcal{L}_2$, the *formula tree* of α, denoted by $FT(\alpha)$, is a binary tree constructed inductively as follows:

1. If α is an atomic formula, then $FT(\alpha)$ is a one-vertex binary tree whose root is α.
2. If $\alpha = \neg\beta$, then $FT(\alpha)$ is a binary tree such that
   - $root(FT(\alpha)) = \neg$, and
   - $root(FT(\alpha))$ has only one child, with $FT(\beta)$ being the subtree of $FT(\alpha)$ rooted at this child.
3. If $\alpha = \beta \wedge \gamma$ (respectively, $\beta \vee \gamma$, $\beta \rightarrow \gamma$, $\beta \leftrightarrow \gamma$), then $FT(\alpha)$ is a binary tree such that

- $root(FT(\alpha)) = \wedge$ (respectively, $\vee$, $\rightarrow$, $\leftrightarrow$), and
- $root(FT(\alpha))$ has two children, with $FT(\beta)$ being the subtree of $FT(\alpha)$ rooted at the left child and $FT(\gamma)$ being the subtree of $FT(\alpha)$ rooted at the right child.

4. If $\alpha = \forall v : \beta$ (respectively, $\exists v : \beta$), where $v \in Var$, then $FT(\alpha)$ is a binary tree such that
   - $root(FT(\alpha)) = \forall v$ (respectively, $\exists v$), and
   - $root(FT(\alpha))$ has only one child, with $FT(\beta)$ being the subtree of $FT(\alpha)$ rooted at this child.
5. If $\alpha = \forall v_h : \beta$ (respectively, $\exists v_h : \beta$), where $v_h \in FVar$, then $FT(\alpha)$ is a binary tree such that
   - $root(FT(\alpha)) = \forall v_h$ (respectively, $\exists v_h$), and
   - $root(FT(\alpha))$ has only one child, with $FT(\beta)$ being the subtree of $FT(\alpha)$ rooted at this child.

The following notation is used in subsequent text:

- For any $v \in Var$, a $\forall v$-vertex and an $\exists v$-vertex in a formula tree are also called a $\forall$*Var-vertex* and an $\exists$*Var-vertex*, respectively.
- For any $v_h \in FVar$, a $\forall v_h$-vertex and an $\exists v_h$-vertex in a formula tree are also called a $\forall$*FVar-vertex* and an $\exists$*FVar-vertex*, respectively.

5.2 Existentially Quantified Conjunctive Normal Forms (ECNF)

A formula α on $\mathcal{L}_2$ is said to be in an *existentially quantified conjunctive normal form* (*ECNF*) iff 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 of $\vee$-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 is similar to a usual conjunctive normal form 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 conjunctive normal form 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.

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
$$a_1, \ldots, a_m \leftarrow b_1, \ldots b_n, \mathbf{f}_1, \ldots, \mathbf{f}_p.$$

5.3 Conversion Algorithm

Assume that

- the initial space INI is the set of all formulas on $\mathcal{L}_2$ that are also formulas on $\mathcal{L}_1$, and
- the target space FIN is the set of all formulas in ECNFs on $\mathcal{L}_2$.

Let a formula α in INI be given as input and $\mathbf{T} = FT(\alpha)$. To transform α into a formula in FIN, $\mathbf{T}$ is changed successively by the steps described below. Fig. 2 depicts an outline of the procedure.

1. *Preparation:*
   (a) Convert $\rightarrow$ and $\leftrightarrow$ equivalently into $\neg$, $\wedge$, and $\vee$, using the following logical equivalences:
   - $\beta \rightarrow \gamma \equiv \neg\beta \vee \gamma$
   - $\beta \leftrightarrow \gamma \equiv (\neg\beta \vee \gamma) \wedge (\neg\gamma \vee \beta)$

   (b) Rename quantified variables so that for any two occurrences of quantifications Qv and $Q'w$, where $v, w \in Var$, $v \neq w$.
2. *Move $\neg$ inwards:* Move $\neg$ inwards equivalently until each occurrence of $\neg$ immediately precedes an atom, using the following logical equivalences:
$$\begin{array}{rcl} \neg(\neg\beta) & \equiv & \beta \\ \neg(\beta \wedge \gamma) & \equiv & \neg\beta \vee \neg\gamma \\ \neg(\beta \vee \gamma) & \equiv & \neg\beta \wedge \neg\gamma \\ \neg\forall x : \alpha & \equiv & \exists x : \neg\alpha \\ \neg\exists x : \alpha & \equiv & \forall x : \neg\alpha \end{array}$$
3. *Move down $\vee$-vertices:* Repeatedly move down $\vee$-vertices in the current state of $\mathbf{T}$ through $\exists$*Var*-vertices, $\forall$*Var*-vertices, and $\wedge$-vertices as far as possible using the following logical equivalences:
$$\begin{array}{rcl} (\exists x : \beta) \vee \gamma & \equiv & \exists x : (\beta \vee \gamma) \\ (\forall x : \beta) \vee \gamma & \equiv & \forall x : (\beta \vee \gamma) \\ (\beta \wedge \gamma) \vee \delta & \equiv & (\beta \vee \delta) \wedge (\gamma \vee \delta) \end{array}$$
4. *Move up $\wedge$-vertices:* Repeatedly move up $\wedge$-vertices in the current state of $\mathbf{T}$ through $\forall$*Var*-vertices as far as possible using the following logical equivalence:
$$\forall x : (\beta \wedge \gamma) \quad \equiv \quad (\forall x : \beta) \wedge (\forall x : \gamma)$$

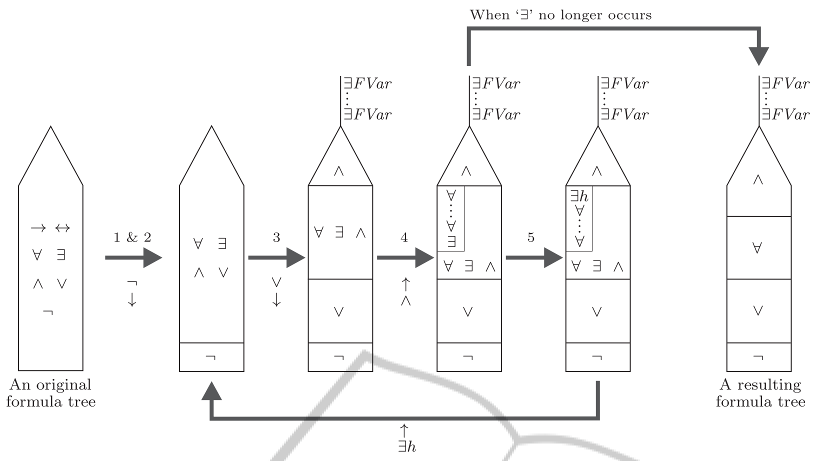

Figure 2: An overview of the conversion procedure.

5. If **T** includes an $\exists$*Var*-vertex, then:
   (a) *Skolemization:* In **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
   $$\exists h, \forall x_1, \ldots, \forall x_n, \forall y : (\beta \vee \neg func(h, x_1, \ldots, x_n, y)),$$
   where $h \in FVar$ such that h has not been used so far.
   (b) *Move up an* $\exists$*FVar-vertex:* Repeatedly move up the new $\exists FVar$-vertex (introduced at Step 5a) through $\wedge$-vertices as far as possible using the following logical equivalence:
   $$(\exists FVar : \beta) \wedge \gamma \quad \equiv \quad \exists FVar : (\beta \wedge \gamma)$$
   (c) Go to Step 3.
6. Stop with the formula represented by the current state of **T** as the output formula.

It is shown in (Akama and Nantajeewarawat, 2011) that this algorithm always terminates and yields an output ECNF in FIN that has the same logical meaning as the input formula.

6 CONCLUSIONS

ET-based computation often requires a search in a certain formula space for a simplified formula that is logically equivalent to an originally given one. Extension of logical formulas in general enlarges the search space both for finding a suitable equivalent logical formula and for finding meaning-preserving formula transformation sequences, thereby increasing the possibility of finding efficient computation paths. The theory for extending a space of formulas by introduction of function variables presented herein allows one to use Skolemization as an equivalent transformation step. It opens up new possibilities for employing ET-based computation to solve logical problems with unrestricted use of existential quantifications.

REFERENCES

Akama, K. and Nantajeewarawat, E. (2006). Formalization of the Equivalent Transformation Computation Model. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 10(3):245–259.

Akama, K. and Nantajeewarawat, E. (2011). Meaning-Preserving Skolemization. Technical report, Hokkaido University, Sapporo, Japan.

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2007). *The Description Logic Handbook*. Cambridge University Press, second edition.

Chang, C.-L. and Lee, R. C.-T. (1973). *Symbolic Logic and Mechanical Theorem Proving*. Academic Press.

Donini, F. M., Lenzerini, M., Nardi, D., and Schaerf, A. (1998). $\mathcal{AL}$-log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems*, 16:227–252.

Horrocks, I., Patel-schneider, P. F., Bechhofer, S., and Tsarkov, D. (2005). OWL Rules: A Proposal and Prototype Implementation. *Journal of Web Semantics*, 3(1):23–40.

Levy, A. Y. and Rousset, M.-C. (1998). Combining Horn Rules and Description Logics in CARIN. *Artificial Intelligence*, 104(1–2):165–209.

Lloyd, J. W. (1987). *Foundations of Logic Programming*. Springer-Verlag, second, extended edition.

Motik, B., Sattler, U., and Studer, R. (2005). Query Answering for OWL-DL with Rules. *Journal of Web Semantics*, 3(1):41–60.