Computing Circumscription Revisited: A Reduction Algorithm

Patrick Doherty and Witold. Łukaszewicz and Andrzej Szałas

Abstract

In recent years, a great deal of attention has been devoted to logics of "commonsense" reasoning. Among the candidates proposed, circumscription has been perceived as an elegant mathematical technique for modeling nonmonotonic reasoning, but difficult to apply in practice. The major reason for this is the 2nd-order nature of circumscription axioms and the difficulty in finding proper substitutions of predicate expressions for predicate variables. One solution to this problem is to compile, where possible, 2nd-order formulas into equivalent 1st-order formulas. Although some progress has been made using this approach, the results are not as strong as one might desire and they are isolated in nature. In this article, we provide a general method which can be used in an algorithmic manner to reduce circumscription axioms to 1st-order formulas. The algorithm takes as input an arbitrary 2nd-order formula and either returns as output an equivalent 1st-order formula, or terminates with failure. The class of 2nd-order formulas, and analogously the class of circumscriptive theories which can be reduced, provably subsumes those covered by existing results. We demonstrate the generality of the algorithm using circumscriptive theories with mixed quantifiers (some involving Skolemization), variable constants, non-separated formulas, and formulas with n-ary predicate variables. In addition, we analyze the strength of the algorithm and compare it with existing approaches providing formal subsumption results.

Computing Circumscription Revisited: A Reduction Algorithm

Patrick Doherty,* Witold Lukaszewicz†and Andrzej Szałas‡

Abstract

In recent years, a great deal of attention has been devoted to logics of "commonsense" reasoning. Among the candidates proposed, circumscription has been perceived as an elegant mathematical technique for modeling nonmonotonic reasoning, but difficult to apply in practice. The major reason for this is the 2nd-order nature of circumscription axioms and the difficulty in finding proper substitutions of predicate expressions for predicate variables. One solution to this problem is to compile, where possible, 2nd-order formulas into equivalent 1st-order formulas. Although some progress has been made using this approach, the results are not as strong as one might desire and they are isolated in nature. In this article, we provide a general method which can be used in an algorithmic manner to reduce circumscription axioms to 1st-order formulas. The algorithm takes as input an arbitrary 2nd-order formula and either returns as output an equivalent 1st-order formula, or terminates with failure. The class of 2nd-order formulas, and analogously the class of circumscriptive theories which can be reduced, provably subsumes those covered by existing results. We demonstrate the generality of the algorithm using circumscriptive theories with mixed quantifiers (some involving Skolemization), variable constants, non-separated formulas, and formulas with n-ary predicate variables. In addition, we analyze the strength of the algorithm and compare it with existing approaches providing formal subsumption results.

1 Introduction and Preliminaries

In recent years, a great deal of attention has been devoted to logical formalisms for "commonsense" reasoning. Among the candidates proposed, three in particular have been studied exhaustively in terms of formal properties and applicability to a variety of domains such as default reasoning, inheritance reasoning, and reasoning about action and change. The three candidates are

- Default Logic [14],

- Autoepistemic Logic [8]),

- Circumscription [11].

The popularity of each of these formalisms has varied through the years as both positive and negative results concerning their respective expressibility and applicability to various domains have been investigated. One problem that has plagued the general acceptance of each of these formalisms as tools for knowledge representation is the difficulty in finding efficient methods for computing inferences. This difficulty is understandable due to the fact that the inference relations in question are generally nonmonotonic in nature. For instance, the popularity of circumscription has declined recently, due to the perceived difficulty in finding proper substitutions for the second-order circumscription axioms normally used when circumscribing predicates. Consequently, a still

open and important research issue is to find more efficient methods for computing inferences for each of the formalisms.

In this article, we will concern ourselves with the search for efficient methods for computing inferences of circumscriptive theories. Circumscription distinguishes itself from default and autoepistemic logic in that the nonmonotonicity is implicit in the use of circumscription rather than an explicit property of the inference relation. This is achieved in general by appealing to the use of a second-order formula which is added to a first-order theory. The purpose of the second order formula is to constrain the models of the theory to those that minimize the extension of a predicate or predicates in some respect.

Circumscription has previously been perceived as an elegant mathematical technique for modeling nonmonotonic reasoning, but difficult to apply in practice. One commonly accepted heuristic for using circumscription is to predict what it is one would want to conclude from the circumscribed theory and then find the proper substitution of predicate expressions in the second-order circumscription axiom that together with the theory would entail the desired conclusion. This heuristic, although reasonable for toy examples is clearly inadequate as a means of using circumscription in practice.

In the literature, there are a number of approaches proposed for computing circumscription that attempt to deal with this inadequacy . These approaches can be classified into three broad groups:

1. Compiling circumscriptive theories into logic programs. A representative example of this approach is Gelfond [5].

2. Developing specialized algorithms to compute preferential entailment. Some representative examples of this approach are a proposal for a theorem prover for circumscription by Ginsberg [6], and the MILO method proposed by Przymusinski [15].

3. Compiling the second-order circumscription axiom into an equivalent first-order formula and then applying classical theorem-proving techniques to the resulting first-order theory.

Although each of the approaches has its advantages and disadvantages, the third is particularly advantageous because the large body of results related to classical theorem-proving techniques are directly applicable. Although some progress has been made using the third approach, the results are somewhat isolated in nature. In addition, a general method which can be used uniformly in an algorithmic manner to make the reductions, is lacking.

In this article, we will describe such a method in terms of an algorithm, which takes a second-order formula as input and returns an equivalent first-order formula as output, or stops returning failure as output. The class of second-order formulas which can be reduced subsumes those covered by existing results. Of course, the fact that the algorithm may stop without returning a first-order formula does not mean that there is not a first-order formula equivalent to the input. It simply implies that their isn't any *or*, the algorithm can not find one if there is. The algorithm is sound in the sense that each step preserves logical equivalence. We will apply the algorithm to the domain of circumscriptive theories and show that by fine-tuning the algorithm, an efficient reduction method for circumscriptive theories is within reach.

1.1 Outline of the Paper

In Sections 1.2, 1.3 and 1.4 the notation used throughout the paper is introduced in addition to a number of preliminary definitions and useful tautologies. In Section 2, existing results on reducing second-order circumscription are reviewed. In Section 3, the major components of the elimination algorithm are described. An in-depth presentation of the algorithm may be found in Appendix A. In Section 4, the strengths and weaknesses of the algorithm are discussed and subsumption results over existing reduction techniques are proven. In Section 5, complexity issues are considered. In Section 6, the algorithm is applied in an informal manner to a number of examples each characterized by a specific feature that illuminates certain aspects of the algorithm and its generality. In Section 7, we conclude with a discussion and future work.

1.2   Notation

An n-ary *predicate expression* is any expression of the form $\lambda \overline{x}.\ A(\overline{x})$, where $\overline{x}$ is a tuple of n individual variables and $A(\overline{x})$ is any formula of first- or second-order classical logic. If U is an n-ary predicate expression of the form $\lambda \overline{x}.\ A(\overline{x})$ and $\overline{\alpha}$ is a tuple of n terms, then $U(\overline{\alpha})$ stands for $A(\overline{\alpha})$. As usual, a predicate constant P is identified with the predicate expression $\lambda \overline{x}.\ P(\overline{x})$. Similarly, a predicate variable Φ is identified with the predicate expression $\lambda \overline{x}.\ \Phi(\overline{x})$.

Truth values true and false are denoted by $\top$ and $\bot$, respectively.

If U and V are predicate expressions of the same arity, then $U \leq V$ stands for $\forall \overline{x}.\ U(\overline{x}) \supset V(\overline{x})$. If $\overline{U} = (U_1, \ldots, U_n)$ and $\overline{V} = (V_1, \ldots, V_n)$ are similar tuples of predicate expressions, i.e. U_i and Vi are of the same arity, $1 \leq i \leq n$, then $\overline{U} \leq \overline{V}$ is an abbreviation for $\bigwedge_{i=1}^{n}[U_i \leq V_i]$. We write $\overline{U} = \overline{V}$ for $(\overline{U} \leq \overline{V}) \wedge (\overline{V} \leq \overline{U})$, and $\overline{U} < \overline{V}$ for $(\overline{U} \leq \overline{V}) \wedge \neg(\overline{V} \leq \overline{U})$.

If A is a formula, $\bar{\sigma} = (\sigma_1, \ldots, \sigma_n)$ and $\bar{\delta} = (\delta_1, \ldots, \delta_n)$ are tuples of any expressions, then $A(\bar{\sigma} \leftarrow \bar{\delta})$ stands for the formula obtained from A by simultaneously replacing *each* occurrence of σ_i by δ_i ($1 \leq i \leq n$). For any tuple $\bar{x} = (x_1, \ldots x_n)$ of individual variables and any tuple $\bar{t} = (t_1, \ldots t_n)$ of terms, we write $\bar{x} = \bar{t}$ to denote the formula $x_1 = t_1 \wedge \cdots \wedge x_n = t_n$. We write $\bar{x} \neq \bar{t}$ as an abbreviation for $\neg(\bar{x} = \bar{t})$.

1.3   Definitions

Definition 1.1 (Second-Order Circumscription) Let $\overline{P}$ be a tuple of distinct predicate constants, $\overline{S}$ be a tuple of distinct function and/or predicate constants disjoint from $\overline{P}$, and let $T(\overline{P}, \overline{S})$ be a sentence. The *second-order circumscription* of $\overline{P}$ in $T(\overline{P}, \overline{S})$ with variable $\overline{S}$, written $Circ_{SO}(T; \overline{P}; \overline{S})$, is the sentence

$$T(\overline{P}, \overline{S}) \wedge \forall \overline{\Phi}\ \overline{\Psi}. \neg\ [T(\overline{\Phi}, \overline{\Psi}) \wedge \overline{\Phi} < \overline{P}] \tag{1}$$

where $\overline{\Phi}$ and $\overline{\Psi}$ are tuples of variables similar to $\overline{P}$ and $\overline{S}$, respectively. ∎

Observe that (1) can be rewritten as

$$T(\overline{P}, \overline{S}) \wedge \forall \overline{\Phi}\ \overline{\Psi}.[T(\overline{\Phi}, \overline{\Psi}) \wedge [\overline{\Phi} \leq \overline{P}]] \supset [\overline{P} \leq \overline{\Phi}]].$$

From the point of view of the elimination of second-order quantification it is often sufficient to consider only the second-order part of the above formula, i.e.

$$\forall \overline{\Phi}\ \overline{\Psi}.[T(\overline{\Phi}, \overline{\Psi}) \wedge [\overline{\Phi} \leq \overline{P}]] \supset [\overline{P} \leq \overline{\Phi}]].$$

Definition 1.2 A predicate variable Φ occurs *positively* (resp. *negatively*) in a formula A if the conjunctive normal form of A contains a subformula of the form $\Phi(\bar{t})$ (resp. $\neg\Phi(\bar{t})$). A formula A is said to be *positive* (resp. *negative*) w.r.t. Φ iff all occurrences of Φ in A are positive (resp. negative). ∎

Definition 1.3 A first-order sentence T is said to be *existential* (*universal*) iff it is of the form $\exists \overline{x}.\ T_1$ (or $\forall \overline{x}.\ T_1$, respectively), where T_1 is quantifier free. ∎

Definition 1.4 A formula T is said to be *monadic* iff it contains one-place predicate constants only, and no function constants except 0-place function constants. ∎

Definition 1.5 Let ϕ be either a predicate constant or a predicate variable and $\overline{\phi}$ be a tuple of predicate constants or a tuple of predicate variables. Then

- a formula $T(\phi)$ is said to be *separated* w.r.t. ϕ iff it is of the form $T_1(\phi) \wedge T_2(\phi)$ where $T_1(\phi)$ is positive w.r.t. ϕ and T_2 is negative w.r.t. ϕ, and

- a sentence T is said to be *separable w.r.t.* $\overline{\phi}$ iff T is equivalent to a formula of the form

$$\bigvee_{i=1}^{m} [B_i(\overline{\phi}) \wedge (\overline{U}_i \leq \overline{\phi})] \tag{2}$$

where $B_i(\overline{\phi})$ is a formula containing no positive occurrences of predicate constants (variables) from $\overline{\phi}$ and each $\overline{U}_i$ is an n-tuple of predicate expressions not containing predicate constants (variables) of $\overline{\phi}$. ∎

1.4 Useful Tautologies

Let us now list some useful tautologies that are used throughout the paper.

Proposition 1.1 The following pairs of formulas are equivalent. (Here Q stands for any quantifier and A, B, C are formulas such that C does not contain free occurrences of the variable x. In clauses (15), (16) and (18), $\bar{t}, \bar{t}_1, \ldots, \bar{t}_n$ are n-tuples of terms and it is assumed that neither C nor any term from $\bar{t}, \bar{t}_1, \ldots, \bar{t}_n$ contains variables from $\bar{x}$. In clause (17), f is a function variable which does not occur in A.)

(1)	$\neg\neg A$	and	A
(2)	$\neg(A \wedge B)$	and	$\neg A \vee \neg B$
(3)	$\neg(A \vee B)$	and	$\neg A \wedge \neg B$
(4)	$\neg \forall x\, A(x)$	and	$\exists x \neg A(x)$
(5)	$\neg \exists x\, A(x)$	and	$\forall x \neg A(x)$
(6)	$\exists x (A(x) \vee B(x))$	and	$\exists x A(x) \vee \exists x B(x)$
(7)	$\forall x (A(x) \wedge B(x))$	and	$\forall x A(x) \wedge \forall x B(x)$
(8)	$Qx(A(x)) \wedge C$	and	$Qx(A(x) \wedge C)$
(9)	$C \wedge Qx(A(x))$	and	$Qx(C \wedge A(x))$
(10)	$Qx(A(x)) \vee C$	and	$Qx(A(x) \vee C)$
(11)	$C \vee Qx(A(x))$	and	$Qx(C \vee A(x))$
(12)	$QxQyA$	and	$QyQxA$
(13)	$A \wedge (B \vee C)$	and	$(A \wedge B) \vee (A \wedge C)$
(14)	$(A \vee B) \wedge C$	and	$(A \wedge C) \vee (B \wedge C)$
(15)	$A(\bar{t})$	and	$\forall \bar{x}(A(\bar{t} \leftarrow \bar{x}) \vee \bar{x} \neq \bar{t})$
(16)	$A(\bar{t}_1) \vee \cdots \vee A(\bar{t}_n)$	and	$\exists \bar{x}(\bar{x} = \bar{t}_1 \vee \cdots \vee \bar{x} = \bar{t}_n) \wedge A(\bar{t}_1 \leftarrow \bar{x}))$
(17)	$\forall \bar{x} \exists y A(\bar{x} \ldots)$	and	$\exists f \forall \bar{x} A(\bar{x}, y \leftarrow f(\bar{x}), \ldots)$
(18)	$A(\bar{t}_1) \wedge \cdots \wedge A(\bar{t}_n)$	and	$\forall \bar{x}(\bar{x} \neq \bar{t}_1 \wedge \cdots \wedge \bar{x} \neq \bar{t}_n) \vee A(\bar{t}_1 \leftarrow \bar{x}))$. ∎

The equivalence (15) was found particularly useful by Ackermann (see [1],[2]). We extend the method by adding the equivalence (16). It makes the technique work in the case of clauses containing more than one positive (or negative) occurrence of the eliminated predicate. This essentially generalizes the Ackermann technique. The equivalence (17) is a second-order formulation of the Skolem reduction (see [20]). It allows us to perform Skolemization (i.e. elimination of existential quantifiers) and unskolemization (i.e. elimination of Skolem functions) in such a way that equivalence is preserved. We call this equivalence *second-order Skolemization*.

2 Known Results about Reducing Second-Order Circumscription

A collection of current results concerning the reduction of second-order circumscription axioms to the first-order case may be found in the handbook article on circumscription by Lifschitz [11], in addition to references to reduction results of other authors not covered in the handbook. In this section, we provide the relevant results from the handbook, in addition to results by other authors pertaining to reduction results for circumscription and second-order logic. The original notations are slightly adjusted to agree with the notation used in this paper.

2.1 Lifschitz' Results

In Lifschitz [9] we are presented with the following result concerning the reduction of second-order circumscription into first-order logic.

Theorem 2.1 If $T(\overline{P})$ is a formula of the form in Equation (2) of Definition 1.5, then $Circ_{SO}(T; \overline{P}; ())$ is equivalent to

$$\bigvee_{i=1}^{m} [C_i \wedge (\overline{U}_i = \overline{P})]$$

where C_i is

$$B_i(\overline{U}_i) \wedge \bigwedge_{j \neq i} \neg [B_j(\overline{U}_j) \wedge (\overline{U}_j < \overline{U}_i)].$$

■

Theorem 2.1 is inapplicable in cases where constants, different than those being circumscribed, are allowed to vary. This limitation can often be avoided by applying the following result (Lifschitz [9]).

Theorem 2.2 $Circ_{SO}(T(\overline{P}, \overline{S}); \overline{P}; \overline{S})$ is equivalent to $T(\overline{P}, \overline{S}) \wedge Circ_{SO}(\exists \overline{\Phi}. T(\overline{P}, \overline{\Phi}); \overline{P}; ())$. ■

Although Theorem 2.2 allows us to transform any second-order circumscription into a circumscription without variable constants, the transformation introduces new second-order variables. These can often be eliminated as follows (Lifschitz [9]).

Theorem 2.3 If $T(\overline{\Phi})$ is equivalent to (2) of Definition 1.5, then $\exists \overline{\Phi}. T(\overline{\Phi})$ is equivalent to $\bigvee_{i=1}^{m} B_i(\overline{U}_i)$. ■

A different result concerning the reduction of second-order circumscription into first-order logic is presented in Lifschitz [10]. The details are these.

Theorem 2.4 If all occurrences of $\overline{P} = (P_1, \ldots, P_n)$ are positive, then $Circ_{SO}(T; \overline{P}; ())$ is equivalent to

$$\bigwedge_{i=1}^{n} [T(P_i) \wedge \forall \overline{x}. \neg [P_i(\overline{x}) \wedge T(\lambda \overline{y}. (P(\overline{y}) \wedge \overline{y} \neq \overline{x}))]].$$

■

In [11] Lifschitz also formulated the following theorem.

Theorem 2.5 If $T(P)$ is a first-order sentence separated w.r.t. P then $Circ_{SO}(T(P); P; ())$ is equivalent to a first-order sentence. ■

2.2 Rabinov's Result

Rabinov [16] provides the following result which subsumes the earlier results of Lifschitz (excluding Theorem 2.5).

If U and V are predicate expressions, then $U \cap V$ stands for $\lambda \overline{x}. (U(\overline{x}) \wedge V(\overline{x}))$.

Let $D_i(P)$ denote $N_i(P) \wedge M_i(P)$ such that the predicate constant P is positive in M_i and negative in N_i. $D_i(P)$ is said to be *p-simple* if $M_i(P)$ has the form $U_i \leq P$, where U_i is a predicate expression not containing P. $D_i(P)$ is said to be *n-simple* if $N_i(P)$ has the form $P \leq U_i$, where U_i is a predicate expression not containing P.

Theorem 2.6 If $T(P)$ is of the form

$$N_0(P) \wedge \bigvee_i D_i(P)$$

where each $D_i(P)$ is either p-simple or n-simple and P is negative in $N_0(P)$, then $Circ_{SO}(T; P; ())$ is equivalent to

$$T(P) \wedge \bigwedge_i R_i(P)$$

where $R_i(P)$ stands for $\neg M_i(P \cap Q_i) \vee Circ_{SO}(M_i(P); P; ())$ if $D_i(P)$ is n-simple, and for $\neg N_i(U_i) \vee \neg(U_i < P)$ otherwise. ∎

2.3 Kolaitis & Papadimitriou's result

In Kolaitis & Papadimitriou [7] the following result was presented.

Theorem 2.7 If T is a first-order existential sentence, then $Circ_{SO}(T; \overline{P}; ())$ is equivalent to a first-order sentence. ∎

2.4 Second-Order Monadic Logic

The following result is due to Löwenheim (see [13]).

Theorem 2.8 If T is a monadic second-order sentence, then T is equivalent to a first-order sentence T'. ∎

2.5 The SCAN Algorithm

The SCAN algorithm was introduced by D. Gabbay and H. J. Ohlbach in [4]. It is formulated as follows:

Definition 2.1 Given a second-order formula $\exists \Phi_1 \ldots \Phi_n A$, where A is a classical first-order formula perform the following steps:

1. Transform A into clause form using the equivalences given in Proposition (1.1). Ignore the prefix with existential first- and second-order quantifiers.

2. Generate all C-resolvents and C-factors with the predicate variables $\Phi_1, \ldots, \Phi_n$ according to the following rules:

   (a) $\Phi(s_1, \ldots, s_n) \vee C, \ \neg\Phi(t_1, \ldots, t_n) \vee D \ \vdash \ C \vee D \vee s_1 \neq t_1 \vee \ldots \vee s_n \neq t_n$

   (b) $\Phi(s_1, \ldots, s_n) \vee \Phi(t_1, \ldots, t_n) \vee C \ \vdash \ \Phi(s_1, \ldots, s_n) \vee C \vee s_1 \neq t_1 \vee \ldots \vee s_n \neq t_n$.

   No self-resolution is allowed. All equivalence preserving simplifications may be applied freely.

3. If the previous step terminates try to unskolemize the resulting formula. If this is successful, the obtained formula is a first-order formula equivalent to the input second-order one. ∎

Observe that SCAN eliminates the second-order quantification for a large class of formulas and can be applied in computing circumscription. On the other hand, the SCAN algorithm may not terminate and the sets of C-resolvents and C-factors may be large.

In what follows we shall not compare SCAN with our algorithm since no syntactic characterization of formulas accepted by SCAN is known. Observe that some examples where SCAN was better than the algorithm given in [18] were known. On the other hand, in the present paper we strengthen the algorithm of [18] by adding the equivalence (16) of Proposition 1.1. After this modification we conjecture that both approaches are successful for the same class of formulas.

However, the additional advantage of our algorithm is that it always terminates, while SCAN may loop. For example, the formula

$$\forall \Phi [(\forall x \Phi(x) \supset \exists y \Phi(y) \wedge Q(x)) \supset \forall x \neg \Phi(x)]$$

when given as input to our algorithm does terminate, while for SCAN it does not.

3   The Elimination Algorithm

In this section we briefly discuss the elimination algorithm. Its complete formulation can be found in the Appendix. The algorithm was originally formulated, in a weaker form, in [18] in the context of modal logics. It is based on Ackermann's techniques developed in connection with the elimination problem (see [1]).

The elimination algorithm is based on the following lemma, proved by Ackermann in 1934 (see [1]). The proof can also be found in [18].

Lemma 3.1 (Ackermann Lemma) Let Φ be a predicate variable and $A(\bar{x})$, $B(\Phi)$ be formulas without second-order quantification. Let $B(\Phi)$ be positive w.r.t. Φ and let A contain no occurrences of Φ at all. Then the following equivalences hold:

$$\exists \Phi \forall \bar{x} [\Phi(\bar{x}) \vee A(\bar{x}, \bar{z})] \wedge B(\Phi \leftarrow \neg \Phi) \equiv B(\Phi \leftarrow A(\bar{x}, \bar{z})) \tag{3}$$

$$\exists \Phi \forall \bar{x} [\neg \Phi(\bar{x}) \vee A(\bar{x}, \bar{z})] \wedge B(\Phi) \equiv B(\Phi \leftarrow A(\bar{x}, \bar{z})) \tag{4}$$

where in the righthand formulas the arguments $\bar{x}$ of A are each time substituted by the respective actual arguments of Φ (renaming the bound variables whenever necessary). ∎

The following proposition together with the equivalences given in Proposition (1.1) is also used in the algorithm.

Proposition 3.1 Let A be a formula of the form $pref(A_1 \wedge \cdots \wedge A_q)$, where $pref$ is a prefix of first-order quantifiers and $A_1, \ldots, A_q$ are disjunctions of literals. In addition, let Φ be a predicate variable occuring in A and $Conj(A)$ those conjuncts in A where Φ occurs. Assume that for any conjunct in $Conj(A)$, Φ occurs either positively, or both positively and negatively (or analogously, negatively, or both negatively and positively). Then

$$\exists \Phi A \equiv pref(A_{i_1} \wedge \cdots \wedge A_{i_r}) \tag{5}$$

where $i_1, \ldots, i_r \in \{1, \ldots, q\}$ and $A_{i_1}, \ldots, A_{i_r}$ are all the conjuncts that do not contain occurrences of Φ (the empty conjunction is regarded to be equivalent to $\top$).

Proof   See [18]. ∎

3.1   Outline of the Elimination Algorithm

We are now ready to outline the elimination algorithm. The algorithm takes a formula of the form $\exists \Phi.A$, where A is a first-order formula, as an input and returns its first-order equivalent or reports failure[1]. Of course, the algorithm can also be used for formulas of the form $\forall \Phi A$, since the latter formula is equivalent to $\neg \exists \Phi \neg A$. Thus, by repeating the algorithm one can deal with formulas containing many arbitrary second-order quantifiers.

The elimination algorithm consists of four phases: (1) preprocessing; (2) preparation for the Ackermann lemma; (3) application of the Ackermann lemma; and (4) simplification. These phases are described below. It is always assumed that (1) whenever the goal specific for a current phase is reached, then the remaining steps of the phase are skipped, (2) every time the equivalence (5) of Proposition 3.1 is applicable, it should be applied.

---

[1] The failure of the algorithm does not mean that the second-order formula at hand cannot be reduced to its first-order equivalent. The problem we are dealing with is not even partially decidable, for first-order definability of the formulas we consider is not an arithmetical notion (see, for instance, [21]).

(1) **Preprocessing.** The purpose of this phase is to transform the formula $\exists\Phi.A$ into a form that separates positive and negative occurrences of the quantified predicate variable Φ. The form we want to obtain is[2]

$$\exists\bar{x}\exists\Phi[(A_1(\Phi)\wedge B_1(\Phi))\vee\cdots\vee(A_n(\Phi)\wedge B_n(\Phi))], \tag{6}$$

where, for each $1\leq i\leq n$, $A_i(\Phi)$ is positive w.r.t. Φ and $B_i(\Phi)$ is negative w.r.t. Φ.[3] The steps of this phase are the following. (i) Eliminate the connectives $\supset$ and $\equiv$ using the usual definitions. Remove redundant quantifiers. Rename individual variables until all quantified variables are different and no variable is both bound and free. Using the usual equivalences, move the negation connective to the right until all its occurrences immediately precede atomic formulas. (ii) Move universal quantifiers to the right and existential quantifiers to the left, applying as long as possible the equivalences $(8)-(11)$ from Proposition 1.1. (iii) In the matrix of the formula obtained so far, distribute all top-level conjunctions over the disjunctions that occur among their conjuncts, applying the equivalences $(13)-(14)$ from Proposition 1.1. (iv) If the resulting formula is not in the form (6), then report the failure of the algorithm. Otherwise replace (6) by its equivalent given by

$$\exists\bar{x}(\exists\Phi(A_1(\Phi)\wedge B_1(\Phi))\vee\cdots\vee\exists\Phi.(A_n(\Phi)\wedge B_n(\Phi))). \tag{7}$$

Try to find Equation (7)'s first-order equivalent by applying the next phases in the algorithm to each disjunct in (7) separately. If the first-order equivalents of each disjunct are successfully obtained then return their disjunction, preceded by the prefix $\exists\bar{x}$, as the output of the algorithm.

(2) **Preparation for the Ackermann lemma.** The goal of this phase is to transform a formula of the form $\exists\Phi(A(\Phi)\wedge B(\Phi))$, where $A(\Phi)$ (resp. $B(\Phi)$) is positive (resp. negative) w.r.t. Φ, into one of the forms (3) or (4) given in Lemma 3.1. Both forms can always be obtained and both transformations should be performed because none, one or both forms may require Skolemization. Unskolemization, which occurs in the next phase, could fail in one form, but not the other. In addition, one form may be substantially smaller than the other. The steps of this phase are based on equivalences $(13)-(17)$ from Proposition 1.1.

(3) **Application of the Ackermann Lemma.** The goal of this phase is to eliminate the second-order quantification over Φ, by applying the Ackermann lemma, and then to unskolemize the function variables possibly introduced. This latter step employs the equivalence (17) from Proposition 1.1.

(4) **Simplification.** Generally, application of Ackermann's Lemma in step (3) often involves the use of equivalence (15) in Proposition 1.1 in the left to right direction. If so, the same equivalence, or its generalization (18), may often be used after application of the Lemma in the right to left direction, substantially shortening the resulting formula.

3.2   Discussion of the Algorithm

Assume we have a second-order formula A of the form

$$\exists\Phi.[(pref\,B)\wedge(pref'C)], \tag{8}$$

where,

- $pref$ and $pref'$ are sequences of first-order quantifiers,

- B and C are quantifier-free formulas in conjunctive normal forms,

---

[2] It should be emphasized that not every formula is reducible into this form.

[3] To increase the strength of the algorithm, it is essential to move as many existentially quantified variables as possible into the prefix of (6).

8

- B is positive w.r.t. Φ, and

- C is negative w.r.t. Φ.

Then, the following proposition holds.

Proposition 3.2 Let A be an input formula of the form (8). Then, as a result, the algorithm returns a first-order formula provided that unskolemization (if necessary) succeeds. ∎

Observe that Skolem functions are introduced in the second step of the algorithm whenever existential quantifiers are to be eliminated. These can appear in the input formula or may be introduced via application of the equivalence (16) of Proposition 1.1.

In the following proposition, we formulate conditions under which no Skolem functions are introduced and the algorithm terminates successfully.

Proposition 3.3 If one of the following conditions holds

- B is universal and each conjunct of B contains at most one occurrence of Φ, or

- C is universal and each conjunct of C contains at most one occurrence of $\neg\Phi$,

then the algorithm always returns a first-order formula as output. ∎

If the input formula cannot be transformed into the form (8) then the algorithm fails.

4   On the Strength of the Algorithm

Let us first prove that the algorithm subsumes, and is even stronger than the results given in [7, 9, 10, 16]. Recall that the formulation of those results is quoted in Section 2.

Let us start with Rabinov's result (and thus the subsumed results of Lifschitz). In fact, the following theorem is stronger than the result of Rabinov.

Theorem 4.1 If $T(P)$ is of the form

$$N_0(P) \wedge \bigvee_i D_i(P)$$

where each $D_i(P)$ is either p-simple or contains no positive occurrences of P[4] and $N_0(P)$ is negative w.r.t. P, then the algorithm eliminates the second-order quantifiers from $Circ_{SO}(T; P; ())$.

<u>Proof</u>
The negated second-order part of $Circ_{SO}(T; P; ())$ takes the following form,

$$\exists\Phi.N_0(\Phi) \wedge \bigvee_i D_i(\Phi)\wedge[\Phi \leq P]\wedge\neg[P \leq \Phi].$$

The following steps show the respective reduction

$$
\begin{array}{ll}
\exists\Phi.N_0(\Phi) \wedge \bigvee_i D_i(\Phi)\wedge[\Phi \leq P]\wedge\neg[P \leq \Phi] & \equiv \\
\exists\Phi.\bigvee_i(N_0(\Phi) \wedge D_i(\Phi)\wedge[\Phi \leq P]\wedge\neg[P \leq \Phi]) & \equiv \\
\bigvee_i \exists\Phi.N_0(\Phi) \wedge D_i(\Phi)\wedge[\Phi \leq P]\wedge\neg[P \leq \Phi] & \equiv \\
\bigvee_i \exists\Phi.N_0(\Phi) \wedge D_i(\Phi)\wedge\forall\bar{x}.(\neg\Phi(\bar{x})\vee P(\bar{x}))\wedge\exists\bar{z}.(P(\bar{z})\wedge\neg\Phi(\bar{z})) & \equiv \\
\bigvee_i \exists\bar{z}.\exists\Phi.N_0(\Phi) \wedge D_i(\Phi)\wedge\forall\bar{x}.(\neg\Phi(\bar{x})\vee P(\bar{x}))\wedge\neg\Phi(\bar{z})\wedge P(\bar{z}) & \equiv \\
\bigvee_i \exists\bar{z}.\exists\Phi.D_i(\Phi)\wedge N_0(\Phi)\wedge\forall\bar{x}.(\neg\Phi(\bar{x})\vee P(\bar{x}))\wedge\neg\Phi(\bar{z})\wedge P(\bar{z}). &
\end{array}
$$

Observe that all occurrences of Φ in $N_0(\Phi)\wedge\forall\bar{x}.(\neg\Phi(\bar{x})\vee P(\bar{x}))\wedge\neg\Phi(\bar{z})\wedge P(\bar{z})$ are negative. Moreover, D_i either contains no positive occurrences of Φ or is p-simple. In the first case, there are no positive occurrences of Φ at all and it suffices to apply equivalence (5) of Proposition 3.1. In the second case, where D_i is p-simple, it takes the form $U_i \leq \Phi$, i.e. $\forall\bar{x}.\Phi(\bar{x})\vee\neg U_i(\bar{x})$ and the Ackermann lemma can be applied directly. ∎

---

[4] Rabinov requires n-simplicity here

The following theorem shows that the algorithm eliminates second-order quantification in the case of existential theories considered in [7].

Theorem 4.2 If T is a first-order existential sentence, then the algorithm eliminates second-order quantification from $Circ_{SO}(T; \overline{P}; ())$.

Proof
The negated second-order part of $Circ_{SO}(T; P; ())$ takes the following form,

$$\exists \Phi. T(\Phi) \wedge [\Phi \leq P] \wedge \neg [P \leq \Phi].$$

By assumption $T(\Phi)$ is of the form $\exists \bar{x}.T'(\bar{x})$, where $T'(\bar{x})$ is quantifier free.

The following steps show the respective reduction

$$
\begin{aligned}
&\exists \Phi. T(\Phi) \wedge [\Phi \leq P] \wedge \neg[P \leq \Phi] &\equiv \\
&\exists \Phi. \exists \bar{x}. T\prime(\Phi) \wedge [\Phi \leq P] \wedge \neg[P \leq \Phi] &\equiv \\
&\exists \bar{x} \exists \Phi. T\prime(\Phi) \wedge \forall \bar{y}(\neg \Phi(\bar{y}) \vee P(\bar{y})) \wedge \exists \bar{z}.(P(\bar{z}) \wedge \neg \Phi(\bar{z})) &\equiv \\
&\exists \bar{x} \bar{z} \exists \Phi. T\prime(\Phi) \wedge \forall \bar{y}(\neg \Phi(\bar{y}) \vee P(\bar{y})) \wedge P(\bar{z}) \wedge \neg \Phi(\bar{z}).
\end{aligned}
$$

Next we transform the above formula into the disjunctive normal form (treating $\forall \bar{y}(\neg \Phi(y) \vee P(\bar{y}))$ as an atomic formula) and then distribute the existential quantifiers over disjunctions. Now each disjunct is a conjunction of atomic formulas and $\forall \bar{y}(\neg \Phi(y) \vee P(\bar{y}))$. Thus, by a simple application of equivalence (15) of proposition 1.1, each disjunct can be transformed into the form required in the Ackermann lemma. ∎

Theorem 4.3 If T is a first-order monadic sentence, then the algorithm eliminates second-order quantification from $Circ_{SO}(T; \overline{P}; \overline{S})$.

Proof
The reduction that works here is given in Ackermann [[2], pp. 41-46], which uses Proposition 1.1 and a weaker form of Lemma 3.1. It can easily be reformulated in terms of the phases of our elimination algorithm.
∎

4.1 Comparison of Approaches

In comparing the different approaches and results concerning the reduction of circumscriptive theories, we will refer to Figure 1 below, which provides a pictorial view of the subsumption relation between the various theorems and types of theories reduced. DLS refers to our algorithm, MIXED refers to theories with mixed quantifiers, VC refers to theories which allow variable constants, and MONAD refers to theories with only monadic sentences. In addition, $\forall$ and $\exists$ refer to purely universal and existential theories, respectively, while $\forall \exists$ refers to those theories where Skolemization is necessary, and $\exists \forall$ refers to mixed theories not requiring Skolemization. The solid arrows denote subsumption. In addition there are two broken solid arrows. The arrow pointing towards Th 2.5 is broken to signify that although the DLS algorithm in its general form does not fully subsume Theorem 2.5, when specialized appropriately, it does. The arrow pointing towards SKOLEM is broken to signify that the DLS algorithm works for those theories involving Skolemization when the unskolemization step is successful and the algorithm returns a first-order formula as output. Since, it may not be possible to unskolemize certain theories successfully, there is no complete subsumption of this class.

4.1.1 Positive Results

In addition to the results described in the previous section, observe that the method we propose is also stronger in regard to the following features:
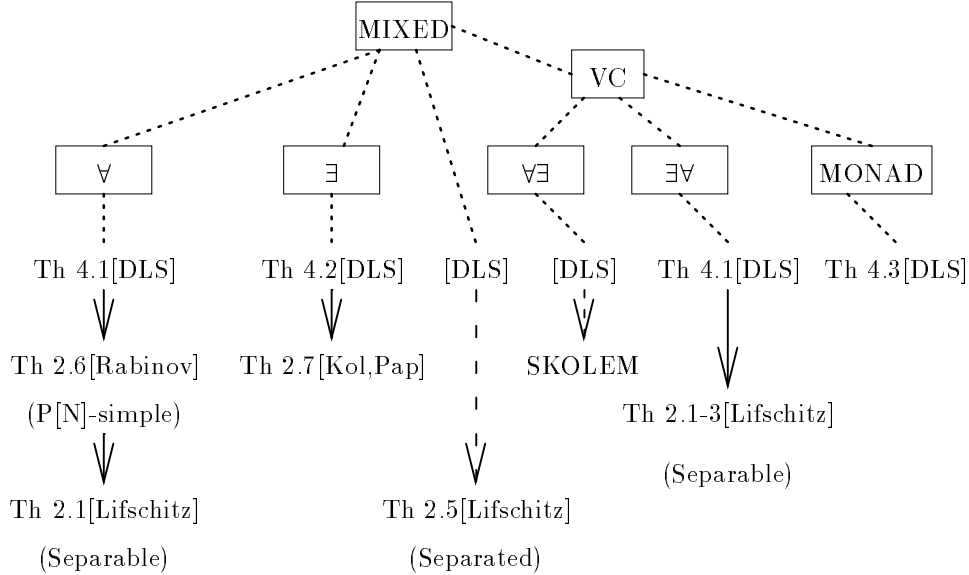
Figure 1: Subsumption Results.

- DLS provides us with a more general approach to existential quantification due to the possibility of allowing Skolemization. Thus it works for combinations of existential and universal quantifiers. On the other hand, Kolaitis and Papadimitriou consider pure existential formulas, while Lifschitz and Rabinov consider pure universal theories.

- DLS does not distinguish between theories with variable constants and those without. On the other hand both Rabinov, Kolaitis and Papadimitriou, (and Lifschitz to some extent), restrict their theories to those without variable constants. In some cases, Lifschitz's results can reduce theories with variable constants if the theories are separable and no Skolemization is involved. (See the next section for problems DLS has with separated theories).

- DLS permits as input circumscriptive theories with arbitrary numbers of minimized and varied predicates. This is not the case for Rabinov's result nor for Lifschitz's result pertaining to separated formulas.

- DLS describes how to constructively transform formulas into the required form.

4.1.2   Negative Results

Note that in the end of Section 3.2 we characterized the class of formulas for which the algorithm fails. Let us now discuss an additional source of weaknesses of the algorithm and a possible way of overcoming those weaknesses.

Observe that the elimination algorithm we deal with is independent of any particular theory. On the other hand, it is well known that second-order quantifiers can sometimes be eliminated when additional information is given.

One good illustrative example originates from the area of modal logics. Namely, McKinsey's axiom is not equivalent to any first-order formula. Accordingly, our algorithm fails (see [18]). However, when one assumes that the accessibility relation is transitive, the elimination is possible, since McKinsey together with transitivity is first-order definable (see [21]).

The same situation may occur when one computes circumscription. Consider the theorem of Lifschitz (Theorem 2.5 above). It permits us to deal with any sequences of first-order quantifiers

provided that the formula is separated. The proof given by Lifschitz is based on a clever move which applies knowledge about the first-order theory one works with. Observe that in Theorem 2.5 the sentence $T(P)$ is assumed to be separated, i.e. it is of the form $T_1(P) \wedge T_2(P)$, where $T_1(P)$ is positive w.r.t. P and $T_2(P)$ is negative w.r.t. P. Thus $Circ_{SO}(T_1 \wedge T_2; P; ())$ is equivalent to

$$T_1(P) \wedge T_2(P) \wedge \neg \exists \Phi . \neg \, [T_1(\Phi) \wedge T_2(\Phi) \wedge \Phi < P].$$

Since $T_2(P)$ is negative w.r.t. P, $T_2(P)$ together with $\Phi < P$ imply $T_2(\Phi)$. Thus when $T_2(P)$ is taken into consideration, one substantially simplifies the second order circumscription into the following second-order formula

$$T_1(P) \wedge T_2(P) \wedge \neg \exists \Phi . \neg \, [T_1(\Phi) \wedge \Phi < P].$$

The last formula is reducible to a first-order sentence (and is, in fact, in the scope of our algorithm).

The above examples show that the general algorithm we presented can (and should be) tuned to the particular situation it is applied to. Since circumscription is always defined over some first-order theory, moves similar to the method used by Lifschitz above, should be incorporated into the algorithm. If this is done for the case of separated theories, then the specialized version of our algorithm subsumes all previous results concerning the reduction of circumscriptive theories. (see Section 4.2 for more details).

4.2   A General Methodology for Use of the Algorithm

The elimination algorithm we present is very general in that it can be applied to any second-order formula. However, in specific applications the algorithm can often be substantially improved. Two such improvements have already been provided. In the first case, we suggest using Proposition 3.1 each time it is applicable. The reason for this is that the proposition allows one to immediately eliminate a second-order variable in the formula in question. As shall be seen in section 6, the proposition is not just a theoretical result – it is sometimes applicable when reducing circumscription axioms.

Another improvement we provide is the phase of simplification. It was observed that in many practically occuring situations the formula obtained as the result of applying the Ackermann lemma can be substantially simplified by using Proposition 1.1 (15). Perhaps this is the case because of the specific form of the circumscription axiom. However, it is obviously worth doing while applying the elimination algorithm to circumscriptive theories.

As already stated, our algorithm subsumes almost all known results concerning the reduction of circumscriptive theories. The only exception appears to be the result of Lifschitz, presented in Theorem 2.5. However, when specifying our algorithm for the purpose of reducing circumscription, this specialized result can easily be built into the preprocessing phase. Moreover, even the result of Rabinov, which is subsumed by the elimination algorithm, could also be built in, simply to make the algorithm more effective.

In concluding, the situation may be summarized as follows. Given the general form of our algorithm and a domain to which it will be applied, analyze the syntactic character of theories in the domain and integrate any useful reduction heuristic in the preprocessing or simplification phase of the algorithm.

5   Complexity of Reduction

Observe that the elimination algorithm we consider, terminates and is easily mechanizable. Let us now estimate its complexity.

First observe that during phase (3) of the algorithm, the form of the formula to be transformed is[5]

$$\exists \Phi \forall \bar{x} [\Phi(\bar{x}) \vee A(\bar{x}, \bar{z})] \wedge B(\Phi \leftarrow \neg \Phi) \tag{9}$$

---

[5] The second form considered in lemma 3.1 is symmetric to the first one.

and then its form is

$$B(\Phi \leftarrow A(\bar{x}, \bar{z})) \tag{10}$$

after application of the Ackermann lemma.

Thus, if the length of (9) is n, then the length of (10) is less than n^2. Observe, however, that this worst case occurs when Φ has $O(n)$ occurrences in (9). In practical examples, however, the length of (10) is usually $O(n)$ (and often less than the length of (9)).

The worst case analysis of steps (1) and (2) shows that the size of the transformed formula can increase exponentially (due to possible transformations between disjunctive and conjunctive normal forms). This, however, is again a rare phenomenon – see examples below, in particular Section 6.7 concerning a Kolaitis and Papadimitriou example.

6   Applying the Algorithm to some Examples

The best way to understand how the algorithm works is to apply it to examples. In this section, we apply the algorithm to a number of different examples, each highlighting a particular strength of the algorithm. A number of examples are taken directly from the literature concerning the use of circumscription in knowledge representation. We take a number of liberties in applying the algorithm so as not to drown in details. For example, step (2) in the previous section states that both forms of Ackermann's Lemma should be considered. In the examples, we choose one form and apply the algorithm. This saves considerable space. Also, the simplification phase is omitted unless it can be applied.

6.1   Block Example

Example 6.1 (Block example) Let $\Gamma(Ab, On)$ be the theory

$$[b1 \neq b2 \wedge B(b1) \wedge B(b2) \wedge \neg On(b1)] \wedge [\forall x . B(x) \wedge \neg Ab(x) \supset On(x)], \tag{11}$$

where B and On are abbreviations for $Block$ and $Ontable$, respectively. The circumscription of $\Gamma(Ab, On)$ with Ab minimized and On varied is

$$Circ_{SO}(\Gamma(Ab, On); Ab; On) \equiv \Gamma(Ab, On) \wedge \forall \Phi \forall \Psi [[\Gamma(\Phi, \Psi) \wedge \Phi \leq Ab] \supset [Ab \leq \Phi]], \tag{12}$$

where

$$\Gamma(\Phi, \Psi) \equiv \quad [b1 \neq b2 \wedge B(b1) \wedge B(b2) \wedge \neg \Psi(b1)] \wedge [\forall x . B(x) \wedge \neg \Phi(x) \supset \Psi(x)] \tag{13}$$

$$\Phi \leq Ab \equiv \quad \forall x . \Phi(x) \supset Ab(x) \tag{14}$$

$$Ab \leq \Phi \equiv \quad \forall x . Ab(x) \supset \Phi(x). \tag{15}$$

In the following, we will reduce

$$\forall \Phi \forall \Psi [\Gamma(\Phi, \Psi) \wedge [\Phi \leq Ab] \supset [Ab \leq \Phi]] \tag{16}$$

in (12). Negating (16), we get

$$\exists \Phi \exists \Psi [\Gamma(\Phi, \Psi) \wedge [\Phi \leq Ab] \wedge \neg [Ab \leq \Phi]]. \tag{17}$$

Since we will try to remove Φ first, we replace (17) by the equivalent

$$\exists \Psi \exists \Phi [\Gamma(\Phi, \Psi) \wedge [\Phi \leq Ab] \wedge \neg [Ab \leq \Phi]]. \tag{18}$$

Preprocessing. Replacing $\Gamma(\Phi, \Psi)$, $\Phi \leq Ab$ and $Ab \leq \Phi$ by their equivalent forms (13)–(15), eliminating $\supset$ and renaming individual variables, we obtain

$$\exists \Psi \exists \Phi [b1 \neq b2 \wedge B(b1) \wedge B(b2) \wedge \neg \Psi(b1)$$
$$\wedge \forall x (\neg B(x) \vee \Phi(x) \vee \Psi(x)) \wedge \forall y (\neg \Phi(y) \vee Ab(y)) \wedge \exists z (Ab(z) \wedge \neg \Phi(z))]. \tag{19}$$

We next move $\exists z$ to the left, obtaining

$$\exists z \exists \Psi \exists \Phi [b1 \neq b2 \wedge B(b1) \wedge B(b2) \wedge \neg \Psi(b1)$$
$$\wedge \forall x (\neg B(x) \vee \Phi(x) \vee \Psi(x)) \wedge \forall y (\neg \Phi(y) \vee Ab(y)) \wedge (Ab(z) \wedge \neg \Phi(z))]. \tag{20}$$

Preparation for the Ackermann lemma. (20) is in the form suitable for application of the Ackermann lemma. To make this more explicit, we rewrite (20) as

$$\exists z \exists \Psi \exists \Phi \forall x [(\Phi(x) \vee \neg B(x) \vee \Psi(x)) \wedge \forall y (\neg \Phi(y) \vee Ab(y)) \wedge \neg \Phi(z) \wedge Ab(z)$$
$$\wedge \neg \Psi(b1) \wedge b1 \neq b2 \wedge B(b1) \wedge B(b2)]. \tag{21}$$

Application of the Ackermann lemma. Applying the Ackermann lemma to (21), we obtain

$$\exists z \exists \Psi [\forall y (\neg B(y) \vee \Psi(y) \vee Ab(y)) \wedge (\neg B(z) \vee \Psi(z)) \wedge Ab(z)$$
$$\wedge \neg \Psi(b1) \wedge b1 \neq b2 \wedge B(b1) \wedge B(b2)]. \tag{22}$$

We next try to remove Ψ in (22).

Preprocessing. The formula (22) is already in the form which is the goal of this phase. To see this, we rewrite it as

$$\exists z \exists \Psi [\neg \Psi(b1) \wedge \forall y (\Psi(y) \vee \neg B(y) \vee Ab(y))$$
$$\wedge (\Psi(z) \vee \neg B(z)) \wedge Ab(z) \wedge b1 \neq b2 \wedge B(b1) \wedge B(b2)]. \tag{23}$$

Preparation for the Ackermann lemma. Applying Proposition 1.1 (15) to $\neg \Psi(b1)$ in (23), we obtain

$$\exists z \exists \Psi \forall s [(\neg \Psi(s) \vee s \neq b1) \wedge \forall y (\Psi(y) \vee \neg B(y) \vee Ab(y))$$
$$\wedge (\Psi(z) \vee \neg B(z)) \wedge Ab(z) \wedge b1 \neq b2 \wedge B(b1) \wedge B(b2)]. \tag{24}$$

Application of the Ackermann lemma. We apply the Ackermann lemma to (24), obtaining

$$\exists z \forall y [(y \neq b1 \vee \neg B(y) \vee Ab(y))$$
$$\wedge (z \neq b1 \vee \neg B(z)) \wedge Ab(z) \wedge b1 \neq b2 \wedge B(b1) \wedge B(b2)]. \tag{25}$$

Simplification. Using Proposition 1.1 (15), we replace (25) by

$$\exists z [(\neg B(b1) \vee Ab(b1))$$
$$\wedge (z \neq b1 \vee \neg B(z)) \wedge Ab(z) \wedge b1 \neq b2 \wedge B(b1) \wedge B(b2)]. \tag{26}$$

Negating (26) results in

$$\forall z [(B(b1) \wedge \neg Ab(b1)) \vee (z = b1 \wedge B(z) \vee \neg Ab(z) \vee b1 = b2 \vee \neg B(b1) \vee \neg B(b2)]. \tag{27}$$

The first-order formula (27) is logically equivalent to the second-order formula (16). Consequently,

$$Circ_{SO}(\Gamma(Ab, On); Ab; On) \equiv \Gamma(Ab, On) \wedge \tag{28}$$
$$\forall z [(B(b1) \wedge \neg Ab(b1)) \vee (z = b1 \wedge B(z) \vee \neg Ab(z) \vee b1 = b2 \vee \neg B(b1) \vee \neg B(b2)]. \tag{29}$$

At this stage, the algorithm terminates but we can continue simplifying relative to the original theory. (27) together with (11) implies

$$Ab(b1) \supset \forall z (\neg Ab(z) \vee (z = b1 \wedge B(z))) \tag{30}$$

and thus implies

$$\forall z (Ab(z) \supset (z = b1 \wedge B(z))), \tag{31}$$

which states that for any object z, either it is normal ($\neg Ab(z)$) or it is a block and $b1$. In other words, the only abnormal object is the block $b1$.

6.2 The Scope Example

This example is due to Etherington, Kraus and Perlis [3]. It contains both an existentially and a universally quantified formula. The theory contains a counterexample axiom which asserts that there is an exception to a default rule stating that birds normally fly. The example is considered conceptually problematic because even though the number of exceptions to the rule is minimized, any bird considered in the theory might be the exceptional bird. Consequently, nothing more can be entailed about any particular birds flying capabilities other than what could be entailed using the original axioms.

Example 6.2 (Scope Example) Let $\Gamma(Ab, F)$ be the theory

$$[\exists x . B(x) \wedge \neg F(x)] \wedge [\forall x . B(x) \wedge \neg Ab(x) \supset F(x)], \tag{32}$$

where B and F stand for $Bird$ and $Flies$, respectively. The circumscription of $\Gamma(Ab, F)$ with Ab minimized and F varied is

$$Circ_{SO}(\Gamma(Ab, F); Ab; F) \equiv \Gamma(Ab, F) \wedge \forall \Phi \forall \Psi[\Gamma(\Phi, \Psi) \wedge [\Phi \leq Ab] \supset [Ab \leq \Phi]], \tag{33}$$

where

$$\Gamma(\Phi, \Psi) \equiv \quad [\exists x . B(x) \wedge \neg \Psi(x)] \wedge [\forall x . B(x) \wedge \neg \Phi(x) \supset \Psi(x)] \tag{34}$$

$$\Phi \leq Ab \equiv \qquad \forall x . \Phi(x) \supset Ab(x) \tag{35}$$

$$Ab \leq \Phi \equiv \qquad \forall x . Ab(x) \supset \Phi(x). \tag{36}$$

In the following, we will reduce

$$\forall \Phi \forall \Psi[\Gamma(\Phi, \Psi) \wedge [\Phi \leq Ab] \supset [Ab \leq \Phi]] \tag{37}$$

in (33). Negating (37), we obtain

$$\exists \Phi \exists \Psi[\Gamma(\Phi, \Psi) \wedge [\Phi \leq Ab] \wedge \neg [Ab \leq \Phi]]. \tag{38}$$

We start by removing Ψ.

Preprocessing. Replacing $\Gamma(\Phi, \Psi)$, $\Phi \leq Ab$ and $Ab \leq \Phi$ by their equivalents given by (34)–(36), eliminating $\supset$ and renaming individual variables, we obtain

$$\exists \Phi \exists \Psi[\exists w(B(w) \wedge \neg \Psi(w)) \wedge \forall x(\neg B(x) \vee \Phi(x) \vee \Psi(x))$$
$$\wedge \forall y(\neg \Phi(y) \vee Ab(y)) \wedge \exists z(Ab(z) \wedge \neg \Phi(z))]. \tag{39}$$

Moving $\exists w$ and $\exists z$ to the left, we obtain

$$\exists w \exists z \exists \Phi \exists \Psi[(B(w) \wedge \neg \Psi(w)) \wedge \forall x(\neg B(x) \vee \Phi(x) \vee \Psi(x))$$
$$\wedge \forall y(\neg \Phi(y) \vee Ab(y)) \wedge (Ab(z) \wedge \neg \Phi(z))]. \tag{40}$$

Preparation for the Ackermann lemma. (40) is in a form suitable for application of the Ackermann lemma. To see this, we rewrite it as

$$\exists w \exists z \exists \Phi \exists \Psi \forall x[(\Psi(x) \vee \neg B(x) \vee \Phi(x)) \wedge (\neg \Psi(w) \wedge B(w))$$
$$\wedge \forall y(\neg \Phi(y) \vee Ab(y)) \wedge Ab(z) \wedge \neg \Phi(z)]. \tag{41}$$

Application of the Ackermann lemma. Applying the Ackermann lemma to (41), we get

$$\exists w \exists z \exists \Phi[((\neg B(w) \vee \Phi(w)) \wedge B(w)) \wedge \forall y(\neg \Phi(y) \vee Ab(y)) \wedge Ab(z) \wedge \neg \Phi(z)] \tag{42}$$

which is equivalent to

$$\exists w \exists z \exists \Phi[\Phi(w) \wedge B(w) \wedge \forall y(\neg \Phi(y) \vee Ab(y)) \wedge Ab(z) \wedge \neg \Phi(z)]. \tag{43}$$

We now remove Φ in (43).

Preprocessing.(43) is in the form which is the goal of this phase.

Preparation for the Ackermann lemma. We apply Proposition 1.1 (15) to $\Phi(w)$ in (43) obtaining

$$\exists w \exists z \exists \Phi [\forall s (\Phi(s) \vee s \neq w) \wedge B(w) \wedge \forall y (\neg \Phi(y) \vee Ab(y)) \wedge \neg \Phi(z) \wedge Ab(z)]. \tag{44}$$

Application of the Ackermann lemma. Applying the Ackermann lemma to (44), we get

$$\exists w \exists z \forall y [(y \neq w \vee Ab(y)) \wedge B(w) \wedge z \neq w \wedge Ab(z)] \tag{45}$$

Simplification. Using Proposition 1.1 (15), we replace (45) by

$$\exists w \exists z [Ab(w) \wedge B(w) \wedge z \neq w \wedge Ab(z)]. \tag{46}$$

We then take the negation of (46), obtaining

$$\forall w \forall z [\neg Ab(w) \vee \neg B(w) \vee z = w \vee \neg Ab(z)] \tag{47}$$

which is equivalent to

$$\forall w \forall z [Ab(w) \wedge B(w) \supset (Ab(z) \supset z = w)]. \tag{48}$$

The first-order formula (48) is logically equivalent to the second-order formula (37). Consequently,

$$Circ_{SO}(\Gamma(Ab, F); Ab; F) \equiv \Gamma(Ab, F) \wedge \forall w \forall z [Ab(w) \wedge Ab(z) \supset (B(w) \supset z = w)]. \tag{49}$$

(48) asserts that there is at most one abnormal object that is a bird. Together with the theory (32), we can infer that there is one and only one abnormal bird object, the unspecified bird that doesn't fly. Suppose the additional assertion that Fred is a bird, $B(Fred)$, is added to the theory. As stated previously, nothing can be asserted about the flying capabilities of Fred.

It is interesting to note the direct relation the first-order equivalent of the circumscription axiom has to the theory. It is easily observed for example that adding the unique names assumption to the theory would result in

$$\forall w \forall z [w \neq z \wedge Ab(w) \wedge B(w) \supset \neg Ab(z)]. \tag{50}$$

So, for any two different birds, one or the other is abnormal. Of course, we do not know which one.

Another of the advantages of the algorithmic approach to reducing circumscription axioms is the ease with which both a theory and its reduction proof can be modified. For example, Etherington *et. al.* [3] continue discussing the example by adding the assertion

$$B(Fred) \wedge B(Opus) \wedge Ab(Opus) \tag{51}$$

along with a unique names assumption. The reduction proof for the modified theory is similar to the current reduction proof with the distinction that the first conjunct in (44), $\Phi(s) \vee s \neq w$, is replaced with

$$\Phi(s) \vee (s \neq w \wedge s \neq Opus) \tag{52}$$

and the steps which follow are modified accordingly. The final reduction formula would then be

$$\forall w. Ab(w) \wedge B(w) \supset w = Opus. \tag{53}$$

It would then follow that $F(Fred)$.

6.3 The Birthday Example

Example 6.3 (Birthday Example) This example contains both existentially quantified and universal formulas. In addition, it contains both unary and binary predicates. It is analogous to the scoping example in the previous section, but uses binary predicates.

Let $\Gamma(Ab, G)$ be the theory

$$[\exists x \exists y (B(y) \wedge F(x,y) \wedge \neg G(x,y))] \wedge [\forall x \forall y (B(y) \wedge F(x,y) \wedge \neg Ab(x,y) \supset G(x,y))], \tag{54}$$

where B, F and G are abbreviations for $Birthday$, $Friend$ and $Gives - Gift$, respectively. Here $Ab(x, y)$ has the following intuitive interpretation: "x behaves abnormally w.r.t. y in the situation when y has a birthday and x is a friend of y". The circumscription of $\Gamma(Ab, G)$ with Ab minimized and G varied is

$$Circ_{SO}(\Gamma(Ab, G); Ab; G) \equiv \Gamma(Ab, G) \wedge \forall \Phi \forall \Psi [\Gamma(\Phi, \Psi) \wedge [\Phi \le Ab] \supset [Ab \le \Phi]], \tag{55}$$

where

$$\Gamma(\Phi, \Psi) \equiv \quad [\exists x \exists y (B(y) \wedge F(x,y) \wedge \neg \Psi(x,y))] \wedge [\forall x \forall y (B(y) \wedge F(x,y) \wedge \neg \Phi(x,y) \supset \Psi(x,y))] \tag{56}$$

$$\Phi \le Ab \equiv \quad\quad\quad\quad\quad\quad\quad\quad \forall x \forall y [\Phi(x,y) \supset Ab(x,y)] \tag{57}$$

$$Ab \le \Phi \equiv \quad\quad\quad\quad\quad\quad\quad\quad \forall x \forall y [Ab(x,y) \supset \Phi(x,y)]. \tag{58}$$

In the following, we will reduce

$$\forall \Phi \forall \Psi [\Gamma(\Phi, \Psi) \wedge [\Phi \le Ab] \supset [Ab \le \Phi]] \tag{59}$$

in (55). Negating (59), we obtain

$$\exists \Phi \exists \Psi [\Gamma(\Phi, \Psi) \wedge [\Phi \le Ab] \wedge \neg [Ab \le \Phi]]. \tag{60}$$

We remove Ψ first.

Preprocessing. Replacing $\Gamma(\Phi, \Psi)$, $\Phi \le Ab$ and $Ab \le \Phi$ by their equivalents given by (56)–(58), eliminating $\supset$, renaming individual variables and moving existential quantifiers over individual variables to the left, we obtain

$$\exists x \exists y \exists q \exists r \exists \Phi \exists \Psi [B(y) \wedge F(x,y) \wedge \neg \Psi(x,y)$$
$$\wedge \forall u \forall z (\neg B(z) \vee \neg F(u,z) \vee \Phi(u,z) \vee \Psi(u,z))$$
$$\wedge \forall s \forall t (\neg \Phi(s,t) \vee Ab(s,t)) \wedge Ab(q,r) \wedge \neg \Phi(q,r)]. \tag{61}$$

Preparation for the Ackermann lemma. (61) is in the form suitable for application of the Ackermann lemma. To see this, we rewrite it as

$$\exists x \exists y \exists q \exists r \exists \Phi \exists \Psi \forall u \forall z [(\Psi(u,z) \vee \neg B(z) \vee \neg F(u,z) \vee \Phi(u,z)) \wedge \neg \Psi(x,y)$$
$$\wedge B(y) \wedge F(x,y) \wedge \forall s \forall t (\neg \Phi(s,t) \vee Ab(s,t)) \wedge Ab(q,r) \wedge \neg \Phi(q,r)]. \tag{62}$$

Application of the Ackermann lemma. Applying the Ackermann lemma to (62), we obtain

$$\exists x \exists y \exists q \exists r \exists \Phi [(\neg B(y) \vee \neg F(x,y) \vee \Phi(x,y)) \wedge B(y) \wedge F(x,y)$$
$$\wedge \forall s \forall t (\neg \Phi(s,t) \vee Ab(s,t)) \wedge Ab(q,r) \wedge \neg \Phi(q,r)]. \tag{63}$$

We now remove Φ in (63).

Preprocessing. (63) is in the form which is the goal of this phase.

Preparation for the Ackermann lemma. Using Proposition 1.1 (15), we replace (63) by

$$\exists x \exists y \exists q \exists r \exists \Phi \forall v \forall w [(\Phi(v,w) \vee v \ne x \vee w \ne y \vee \neg B(y) \vee \neg F(x,y))$$
$$\wedge \forall s \forall t (\neg \Phi(s,t) \vee Ab(s,t)) \wedge \neg \Phi(q,r) \wedge B(y) \wedge F(x,y) \wedge Ab(q,r)]. \tag{64}$$

Application of the Ackermann lemma. Applying the Ackermann lemma to (64), we obtain

$$\exists x \exists y \exists q \exists r \forall s \forall t [(s \neq x \lor t \neq y \lor \neg B(y) \lor \neg F(x,y) \lor Ab(s,t))$$
$$\land (q \neq x \lor r \neq y \lor \neg B(y) \lor \neg F(x,y)) \land B(y) \land F(x,y) \land Ab(q,r)]. \tag{65}$$

Simplification. We replace (65) by

$$\exists x \exists y \exists q \exists r [(\neg B(y) \lor \neg F(x,y) \lor Ab(x,y))$$
$$\land (q \neq x \lor r \neq y \lor \neg B(y) \lor \neg F(x,y)) \land B(y) \land F(x,y) \land Ab(q,r)]. \tag{66}$$

Negating (66), we obtain

$$\forall x \forall y \forall q \forall r [(B(y) \land F(x,y) \land \neg Ab(x,y))$$
$$\lor (q = x \land r = y \land B(y) \land F(x,y)) \lor \neg B(y) \lor \neg F(x,y) \lor \neg Ab(q,r)]. \tag{67}$$

(67) is logically equivalent to

$$\forall x \forall y \forall q \forall r [\neg (B(y) \land F(x,y)) \lor ((B(y) \land F(x,y)) \land (\neg Ab(x,y) \lor (q = x \land r = y)))$$
$$\lor \neg Ab(q,r)], \tag{68}$$

which is equivalent to

$$\forall x \forall y \forall q \forall r [\neg (B(y) \land F(x,y)) \lor \neg Ab(x,y) \lor (q = x \land r = y) \lor \neg Ab(q,r)]. \tag{69}$$

The first-order formula (69) is logically equivalent to the second-order formula (59). Consequently,

$$Circ_{SO}(\Gamma(Ab,G); Ab; G) \equiv \Gamma(Ab,G) \land$$
$$\forall x \forall y \forall q \forall r [\neg (B(y) \land F(x,y)) \lor \neg Ab(x,y) \lor (q = x \land r = y) \lor \neg Ab(q,r)]. \tag{70}$$

A more informative sentence, equivalent to (69), is

$$\forall x \forall y \forall q \forall r [Ab(x,y) \land Ab(q,r) \land B(y) \land F(x,y) \supset (q = x \land r = y)]. \tag{71}$$

(71), together with the theory $\Gamma(Ab,G)$, states that there is exactly one pair of individuals, x and y, such that y has a birthday, x is a friend of y and x does not give a gift to y.

6.4 The Hospital Example

Example 6.4 (Hospital Example) Let Γ be the theory

$$[\forall x \exists y (Ab(x,y) \supset H(x,y))] \land [\forall x \exists y (\neg Ab(x,y) \supset H(x,y))]. \tag{72}$$

Here $H(x,y)$ and $Ab(x,y)$ are to be intuitively interpreted as "x is in a hospital in a situation y" and "x behaves abnormally in a situation y", respectively. The circumscription of Γ, with Ab minimized and H varied is

$$Circ_{SO}(\Gamma; Ab; H) \equiv \Gamma \land \forall \Phi \forall \Psi [\Gamma(\Phi, \Psi) \land [\Phi \leq Ab] \supset [Ab \leq \Phi]], \tag{73}$$

where

$$\Gamma(\Phi, \Psi) \equiv \quad \forall x \exists y [\Phi(x,y) \supset \Psi(x,y)] \land \quad \forall x \exists y [\neg \Phi(x,y) \supset \Psi(x,y)] \tag{74}$$
$$\Phi \leq Ab \equiv \quad \forall x \forall y [\Phi(x,y) \supset Ab(x,y)] \tag{75}$$
$$Ab \leq \Phi \equiv \quad \forall x \forall y [Ab(x,y) \supset \Phi(x,y)]. \tag{76}$$

In the following, we will reduce

$$\forall \Phi \forall \Psi [\Gamma(\Phi, \Psi) \land [\Phi \leq Ab] \supset [Ab \leq \Phi]] \tag{77}$$

in (73). Negating (77), we obtain

$$\exists \Phi \exists \Psi [\Gamma(\Phi, \Psi) \land [\Phi \leq Ab] \land \neg [Ab \leq \Phi]]. \tag{78}$$

Since we will try to remove Φ first, we replace (78) by its equivalent given by

$$\exists\Psi\exists\Phi[\Gamma(\Phi,\Psi)\wedge[\Phi\le Ab]\wedge\neg[Ab\le\Phi]]. \tag{79}$$

Preprocessing. Replacing $\Gamma(\Phi,\Psi)$, $\Phi\le Ab$ and $Ab\le\Phi$ by their equivalents given by (74)–(76), eliminating $\supset$ and renaming individual variables, we obtain

$$\exists\Psi\exists\Phi[\forall x\exists y(\neg\Phi(x,y)\vee\Psi(x,y))\wedge\forall q\exists r(\Phi(q,r)\vee\Psi(q,r))$$
$$\wedge\forall u\forall v(\neg\Phi(u,v)\vee Ab(u,v))\wedge\exists s\exists t(Ab(s,t)\wedge\neg\Phi(s,t))]. \tag{80}$$

Moving $\exists s\exists t$ to the left and rearranging the resulting formula, we obtain

$$\exists s\exists t\exists\Psi\exists\Phi[\forall q\exists r(\Phi(q,r)\vee\Psi(q,r))\wedge\forall x\exists y(\neg\Phi(x,y)\vee\Psi(x,y))$$
$$\wedge\forall u\forall v(\neg\Phi(u,v)\vee Ab(u,v))\wedge Ab(s,t)\wedge\neg\Phi(s,t)]. \tag{81}$$

Preparation for the Ackermann lemma. Using Proposition 1.1 (15), we replace $\Phi(q,r)$ in (81) by $\forall z\forall w(\Phi(z,w)\vee z\ne q\vee w\ne r)$. This results in

$$\exists s\exists t\exists\Psi\exists\Phi[\forall q\exists r(\forall z\forall w(\Phi(z,w)\vee z\ne q\vee w\ne r)\vee\Psi(q,r))\wedge\forall x\exists y(\neg\Phi(x,y)\vee\Psi(x,y))$$
$$\wedge\forall u\forall v(\neg\Phi(u,v)\vee Ab(u,v))\wedge Ab(s,t)\wedge\neg\Phi(s,t)] \tag{82}$$

which is equivalent to

$$\exists s\exists t\exists\Psi\exists\Phi[\forall q\exists r\forall z\forall w(\Phi(z,w)\vee z\ne q\vee w\ne r\vee\Psi(q,r))\wedge\forall x\exists y(\neg\Phi(x,y)\vee\Psi(x,y))$$
$$\wedge\forall u\forall v(\neg\Phi(u,v)\vee Ab(u,v))\wedge Ab(s,t)\wedge\neg\Phi(s,t)]. \tag{83}$$

We next eliminate $\exists r$ by Skolemization and move $\forall q$ to the right and $\forall z\forall w$ to the left. The resulting formula is (below f is the introduced function variable)

$$\exists s\exists t\exists\Psi\exists f\exists\Phi\forall z\forall w[(\Phi(z,w)\vee\forall q(z\ne q\vee w\ne f(q)\vee\Psi(q,f(q))))\wedge\forall x\exists y(\neg\Phi(x,y)\vee\Psi(x,y))$$
$$\wedge\forall u\forall v(\neg\Phi(u,v)\vee Ab(u,v))\wedge Ab(s,t)\wedge\neg\Phi(s,t)]. \tag{84}$$

Application of the Ackermann lemma. Applying the Ackermann lemma to (84), we obtain

$$\exists s\exists t\exists f\exists\Psi[\forall x\exists y(\forall q(y\ne q\vee y\ne f(q)\vee\Psi(q,f(q)))\vee\Psi(x,y))$$
$$\wedge\forall u\forall v(\forall q(u\ne q\vee v\ne f(q)\vee\Psi(q,f(q)))\vee Ab(u,v))$$
$$\wedge Ab(s,t)\wedge\forall q(s\ne q\vee t\ne f(q)\vee\Psi(q,f(q)))] \tag{85}$$

which is equivalent to

$$\exists s\exists t\exists f\exists\Psi\forall x\exists y\forall q\forall u\forall v[(y\ne q\vee y\ne f(q)\vee\Psi(q,f(q))\vee\Psi(x,y))$$
$$\wedge(u\ne q\vee v\ne f(q)\vee\Psi(q,f(q))\vee Ab(u,v))$$
$$\wedge Ab(s,t)\wedge(s\ne q\vee t\ne f(q)\vee\Psi(q,f(q)))]. \tag{86}$$

Since all occurrences of Ψ in each conjunct in (86) are positive, all the conjuncts including Ψ, together with $\exists\Psi$, can be removed by Proposition 3.1. This yields

$$\exists s\exists t\exists f\exists\Psi\forall x\exists y\forall q\forall u\forall v.Ab(s,t) \tag{87}$$

which reduces to

$$\exists s\exists t.Ab(s,t). \tag{88}$$

Since we negated the original formula before applying the algorithm, we now negate the result, obtaining

$$\forall s\forall t.\neg Ab(s,t). \tag{89}$$

The first-order formula (89) is logically equivalent to the second-order formula (77). Consequently,

$$Circ_{SO}(\Gamma;Ab;H)\equiv\Gamma\wedge\forall s\forall t.\neg Ab(s,t). \tag{90}$$

which implies

$$\forall x\exists y H(x,y). \tag{91}$$

6.5 The Vancouver Example

This is a variant of an example from Reiter [17]. Rather than using the function *city* as Reiter does, we will use a relation $C(x, y)$ with suitable axioms.

Example 6.5 (Vancouver Example) We begin by defining the binary relation C with the intention that $C(x, y)$ holds iff the city of x is y. In our axiomatization, Reiter's axiom,

$$\forall x. \neg Ab(x) \supset city(x) = city(wife(x)) \tag{92}$$

is replaced with

$$\forall x \forall y \forall z. \neg Ab(x) \land C(x, y) \land C(wife(x), z) \supset y = z. \tag{93}$$

In addition, we add the following axiom guaranteeing that C represents a function:

$$\forall x \forall y \forall z. C(x, y) \land C(x, z) \supset y = z. \tag{94}$$

We do not require that all people live in cities, i.e. we reject the axiom

$$\forall x \exists y. C(x, y). \tag{95}$$

So, the distinction is that our representation of the *city* function is partial, whereas Reiter's is total. Intuitively, our choice seems to make more sense.

Let $\Gamma(Ab, C)$ be the theory

$$[\forall x \forall y \forall z. \neg Ab(x) \land C(x, y) \land C(wife(x), z) \supset y = z] \land [\forall x \forall y \forall z. C(x, y) \land C(x, z)) \supset y = z]. \tag{96}$$

The circumscription of $\Gamma(Ab, City)$ with Ab minimized and C varied is

$$Circ_{SO}(\Gamma(Ab, C); Ab; C) \equiv \Gamma(Ab, C) \land \forall \Phi \forall \Psi [\Gamma(\Phi, \Psi) \land [\Phi \leq Ab] \supset [Ab \leq \Phi], \tag{97}$$

where

$$\Gamma(\Phi, \Psi) \equiv \quad [\forall x \forall y \forall z. \neg \Phi(x) \land \Psi(x, y) \land \Psi(wife(x), z) \supset y = z] \land$$
$$[\forall x \forall y \forall z. \Psi(x, y) \land \Psi(x, z) \supset y = z]. \tag{98}$$

$$\Phi \leq Ab \equiv \qquad\qquad \forall x. \Phi(x) \supset Ab(x) \tag{99}$$

$$Ab \leq \Phi \equiv \qquad\qquad \forall x. Ab(x) \supset \Phi(x). \tag{100}$$

In the following, we will reduce

$$\forall \Phi \forall \Psi [\Gamma(\Phi, \Psi) \land [\Phi \leq Ab] \supset [Ab \leq \Phi]] \tag{101}$$

in (97). Negating (101), we obtain

$$\exists \Phi \exists \Psi [\Gamma(\Phi, \Psi) \land [\Phi \leq Ab] \land \neg[Ab \leq \Phi]]. \tag{102}$$

We start by removing Ψ.

Preprocessing. Replacing $\Gamma(\Phi, \Psi)$, $\Phi \leq Ab$ and $Ab \leq \Phi$ by their equivalents given by (98)–(100), eliminating $\supset$ and renaming individual variables, we obtain

$$\exists \Phi \exists \Psi [\forall x \forall y \forall z (\Phi(x) \lor \neg \Psi(x, y) \lor \neg \Psi(wife(x), z) \lor y = z) \land$$
$$\forall u \forall v \forall w (\neg \Psi(u, v) \lor \neg \Psi(u, w) \lor v = w)) \land \forall s (\neg \Phi(s) \lor Ab(s)) \land \exists t (Ab(t) \land \neg \Phi(t))]. \tag{103}$$

Note that Ψ can be removed directly using Proposition 3.1, since all instances of Ψ occurring in (103) are negative. To prepare the latter formula into the form suitable for the application of Proposition 3.1, we move all quantifiers ranging over individual variables to the left. This results in

$$\exists \Phi \exists \Psi \forall x \forall y \forall z \forall u \forall v \forall w \forall s \exists t [(\Phi(x) \lor \neg \Psi(x, y) \lor \neg \Psi(wife(x), z) \lor y = z) \land$$
$$(\neg \Psi(u, v) \lor \neg \Psi(u, w) \lor v = w) \land (\neg \Phi(s) \lor Ab(s)) \land Ab(t) \land \neg \Phi(t)]. \tag{104}$$

Applying Proposition 3.1 to (104), we obtain

$$\exists\Phi\forall x\forall y\forall z\forall u\forall v\forall w\forall s\exists t\,[(\neg\Phi(s)\vee Ab(s))\wedge Ab(t)\wedge\neg\Phi(t)] \tag{105}$$

which reduces to

$$\exists\Phi\forall s\exists t\,[(\neg\Phi(s)\vee Ab(s))\wedge Ab(t)\wedge\neg\Phi(t). \tag{106}$$

We next try to remove Φ. Again, this can be done directly using Proposition 3.1, since all instances of Φ in (106) are negative. This results in

$$\forall s\exists t.Ab(t) \tag{107}$$

which is equivalent to

$$\exists t.Ab(t). \tag{108}$$

Taking the negation of (108) results in

$$\forall t.\neg Ab(t). \tag{109}$$

The first-order formula (109) is logically equivalent to the second-order formula (101). Consequently,

$$Circ_{SO}(\Gamma;Ab;C)\equiv\Gamma\wedge\forall t\neg Ab(t). \tag{110}$$

6.6 A Preprocessing Example

In the previous examples, the preprocessing phase was very simple. In this example, which appears to be a relatively trivial theory, the preprocessing stage is much more complex.

Example 6.6 (Preprocessing Example) Let Γ be the theory

$$P(a)\supset P(b). \tag{111}$$

The circumscription of Γ, with P minimized is

$$Circ_{SO}(\Gamma;P;())\equiv\Gamma\wedge\forall\Phi[\Gamma(\Phi)[\wedge\Phi\leq P]\supset[P\leq\Phi]], \tag{112}$$

where

$$\Gamma(\Phi)\equiv\quad\Phi(a)\supset\Phi(b), \tag{113}$$
$$\Phi\leq Ab\equiv\quad\forall x.\Phi(x)\supset P(x), \tag{114}$$
$$Ab\leq\Phi\equiv\quad\forall x.P(x)\supset\Phi(x). \tag{115}$$

In the following, we will reduce

$$\forall\Phi[\Gamma(\Phi)\wedge[\Phi\leq Ab]\supset[Ab\leq\Phi]] \tag{116}$$

in (112). Negating (116), we obtain

$$\exists\Phi[\Gamma(\Phi)\wedge[\Phi\leq Ab]\wedge\neg[Ab\leq\Phi]]. \tag{117}$$

Preprocessing. Replacing $\Gamma(\Phi,\Psi)$, $\Phi\leq Ab$ and $Ab\leq\Phi$ by their equivalents given by (113)–(115), eliminating $\supset$ and renaming individual variables, we obtain

$$\exists\Phi[(\neg\Phi(a)\vee\Phi(b))\wedge\forall x(\neg\Phi(x)\vee P(x))\wedge\exists y(P(y)\wedge\neg\Phi(y))]. \tag{118}$$

Moving $\exists y$ to the left, we obtain

$$\exists y\exists\Phi[(\neg\Phi(a)\vee\Phi(b))\wedge\forall x(\neg\Phi(x)\vee P(x))\wedge P(y)\wedge\neg\Phi(y)]. \tag{119}$$

Since positive and negative occurrences of Φ are not properly separated, we distribute the conjunction $\forall x(\neg\Phi(x)\vee P(x))\wedge P(y)\wedge\neg\Phi(y)$ over $\neg\Phi(a)\vee\Phi(b)$. This leads to

$$\exists y\exists\Phi[(\neg\Phi(a)\wedge\forall x(\neg\Phi(x)\vee P(x))\wedge P(y)\wedge\neg\Phi(y))$$
$$\vee(\Phi(b)\wedge\forall x(\neg\Phi(x)\vee P(x))\wedge P(y)\wedge\neg\Phi(y))]. \tag{120}$$

Applying Proposition 1.1(6) to (120), we obtain

$$\exists y\exists\Phi[\neg\Phi(a)\wedge\forall x(\neg\Phi(x)\vee P(x))\wedge P(y)\wedge\neg\Phi(y)]$$
$$\vee\exists y\exists\Phi[\Phi(b)\wedge\forall x(\neg\Phi(x)\vee P(x))\wedge P(y)\wedge\neg\Phi(y)]. \tag{121}$$

The preprocessing phase is successfully completed. We now process the disjuncts from (121) as separate problems. Let us begin with the first disjunct. Note that Φ can be removed from it directly, using Proposition 3.1. To this end, we replace the disjunct with its equivalent given by

$$\exists y\exists\Phi\forall x[\neg\Phi(a)\wedge(\neg\Phi(x)\vee P(x))\wedge P(y)\wedge\neg\Phi(y)]. \tag{122}$$

Applying Proposition 3.1 to (122), we obtain

$$\exists y\forall x.P(y) \tag{123}$$

which reduces to

$$\exists y.P(y). \tag{124}$$

We have succeeded in reducing the first disjunct in (121) to a first-order formula. We now try to do the same for the second one, i.e.

$$\exists y\exists\Phi[\Phi(b)\wedge\forall x(\neg\Phi(x)\vee P(x))\wedge P(y)\wedge\neg\Phi(y)]. \tag{125}$$

Preparation for the Ackermann lemma. Using Proposition 1.1 (15), we replace (125) by

$$\exists y\exists\Phi\forall z[(\Phi(z)\vee z\neq b)\wedge\forall x(\neg\Phi(x)\vee P(x))\wedge P(y)\wedge\neg\Phi(y)]. \tag{126}$$

Application of the Ackermann lemma. Applying the Ackermann lemma to (126), we obtain

$$\exists y\forall x[(x\neq b\vee P(x))\wedge y\neq b\wedge P(y)]. \tag{127}$$

We have now succeeded in reducing the second disjunct of (121). The original second-order formula (121) has now been reduced to the equivalent first-order formula

$$[\exists y.P(y)]\vee[\exists y\forall x[(x\neq b\vee P(x))\wedge y\neq b\wedge P(y)]]. \tag{128}$$

Applying the simplification step to (128) using Proposition 1.1 (15), we obtain

$$[\exists y.P(y)]\vee[\exists y[P(b)\wedge y\neq b\wedge P(y)]]. \tag{129}$$

Finally, we negate (129), obtaining

$$[\forall y\neg P(y)]\wedge\forall y[\neg P(b)\vee y=b\vee\neg P(y)]. \tag{130}$$

The first-order formula (130) is logically equivalent to the second-order formula (116). Consequently,

$$Circ_{SO}(\Gamma;P;())\equiv\Gamma\wedge[\forall y\neg P(y)]\wedge\forall y[\neg P(b)\vee y=b\vee\neg P(y)]. \tag{131}$$

(130) can be simplified using some standard equivalences:

$$[\forall y\neg P(y)]\wedge\forall y[\neg P(b)\vee y=b\vee\neg P(y)]\equiv\forall y.\neg P(y). \tag{132}$$

6.7 An Existential Example

Kolaitis and Papadimitriou [7] show that the circumscription of any existential first-order formula is equivalent to a first-order formula. We have already shown that the algorithm we propose here generalizes Kolaitis and Papadimitriou's results. It is interesting to compare these reduction techniques in terms of complexity. Kolaitis and Papadimitriou [7] state

> We notice that computing a first-order sentence equivalent to the circumscription of P in an existential first-order formula $\psi(P)$ seems to increase the size of $\psi(P)$ exponentially, a phenomenon not observed in the other known cases of first-order circumscription studied in [Lif85]. It would be interesting to determine whether this is inherent to existential first-order formula, or a particular creation of our proof.

Example 6.7 (Existential Example) We now take the example used by Kolaitis and Papadimitriou and compare the resulting first-order formula with that generated by our algorithm. Kolaitis and Papadimitriou apply their reduction technique to the theory

$$\exists x_1 \exists x_2 [R(x_1, x_2) \wedge P(x_1) \wedge P(x_2)] \tag{133}$$

and circumscribe P without varying predicates. The first-order equivalent they obtain is

$$\exists x_1 (R(x_1, x_1) \wedge P(x_1) \wedge (\forall y (P(y) \equiv y = x_1))$$
$$\vee [\exists x_1 \exists x_2 (R(x_1, x_2) \wedge P(x_1) \wedge P(x_2) \wedge (x_1 \neq x_2) \wedge$$
$$(\forall y (P(y) \equiv (y = x_1 \vee y = x_2))) \wedge \neg R(x_1, x_1) \wedge \neg R(x_2, x_2))]. \tag{134}$$

We apply our reduction algorithm to the same theory and compare the results.

Let $\Gamma(P)$ be the theory

$$\exists x_1 \exists x_2 [R(x_1, x_2) \wedge P(x_1) \wedge P(x_2)]. \tag{135}$$

The circumscription of $\Gamma(P)$ with P minimized without variable predicates is

$$Circ_{SO}(\Gamma(P); P; ()) \equiv \Gamma(P) \wedge \forall \Phi [\Gamma(\Phi) \wedge [\Phi \leq P] \supset [P \leq \Phi]], \tag{136}$$

where

$$\Gamma(\Phi) \equiv \quad \exists x_1 \exists x_2 [R(x_1, x_2) \wedge \Phi(x_1) \wedge \Phi(x_2)] \tag{137}$$
$$\Phi \leq P \equiv \qquad \forall x. \Phi(x) \supset P(x) \tag{138}$$
$$P \leq \Phi \equiv \qquad \forall x. P(x) \supset \Phi(x). \tag{139}$$

In the following, we will reduce

$$\forall \Phi [\Gamma(\Phi) \wedge [\Phi \leq P] \supset [P \leq \Phi]] \tag{140}$$

in (136). Negating (140), we obtain

$$\exists \Phi [\Gamma(\Phi) \wedge [\Phi \leq P] \wedge \neg [P \leq \Phi]]. \tag{141}$$

Preprocessing. Replacing $\Gamma(\Phi)$, $\Phi \leq P$ and $P \leq \Phi$ by their equivalents given by (137)–(139), eliminating $\supset$ and renaming individual variables, we obtain

$$\exists \Phi [\exists x_1 \exists x_2 (R(x_1, x_2) \wedge \Phi(x_1) \wedge \Phi(x_2)) \wedge$$
$$\forall y (\neg \Phi(y) \vee P(y)) \wedge \exists z (P(z) \wedge \neg \Phi(z))]. \tag{142}$$

We next move $\exists x_1 \exists x_2 \exists z$ to the left, obtaining

$$\exists x_1 \exists x_2 \exists z \exists \Phi [R(x_1, x_2) \wedge \Phi(x_1) \wedge \Phi(x_2) \wedge$$
$$\forall y (\neg \Phi(y) \vee P(y)) \wedge P(z) \wedge \neg \Phi(z)]. \tag{143}$$

Preparation for the Ackermann lemma. Applying Proposition 1.1 (15) and some standard equivalences, we replace (143) by

$$\exists x_1 \exists x_2 \exists z \exists \Phi \forall q [(\Phi(q) \vee (q \neq x_1 \wedge q \neq x_2)) \wedge (R(x_1, x_2) \wedge$$
$$\forall y(\neg \Phi(y) \vee P(y)) \wedge P(z) \wedge \neg \Phi(z)]. \tag{144}$$

Application of the Ackermann lemma. The Ackermann lemma can now be applied to (144) resulting in

$$\exists x_1 \exists x_2 \exists z [R(x_1, x_2) \wedge \forall y((y \neq x_1 \wedge y \neq x_2) \vee P(y)) \wedge$$
$$P(z) \wedge z \neq x_1 \wedge z \neq x_2]. \tag{145}$$

Simplification. Applying Proposition 1.1(18) to (144) results in

$$\exists x_1 \exists x_2 \exists z [R(x_1, x_2) \wedge P(x_1) \wedge P(x_2) \wedge P(z) \wedge z \neq x_1 \wedge z \neq x_2]. \tag{146}$$

Negating (146), we obtain

$$\forall x_1 \forall x_2 \forall z [\neg R(x_1, x_2) \vee \neg P(x_1) \vee \neg P(x_2) \vee \neg P(z) \vee z = x_1 \vee z = x_2]. \tag{147}$$

The first-order formula (147) is logically equivalent to the second-order formula (140). Consequently,

$$Circ_{SO}(\Gamma; P) \equiv \Gamma(P) \wedge$$
$$\forall x_1 \forall x_2 \forall z [\neg R(x_1, x_2) \vee \neg P(x_1) \vee \neg P(x_2) \vee \neg P(z) \vee z = x_1 \vee z = x_2]. \tag{148}$$

Comparing (148) with (134), it is easily observed that there is a substantial difference in the size of the formulas.

6.8 Interpreting the Results

There are a number of interesting observations that can be made on the basis of the above examples.

1. In all the examples, the first-order equivalent of the circumscription axiom is shorter than the axiom itself.

2. In some examples, the first-order equivalent of the circumscription axiom may seem a little strange when considered in isolation. However, it always makes perfect sense when the theory at hand is taken into account. Reconsider, for instance, the example 6.2. The first-order equivalent of the circumscription axiom is

$$\forall w \forall z [Ab(w) \wedge B(w) \supset (Ab(z) \supset z = w)] \tag{149}$$

   which states that there is at most one abnormal object that is a bird. The conjunction of (149) and the sentence being circumscribed result in a formula stating that there is exactly one abnormal bird object that does not fly.

3. Note that for certain examples, such as the hospital example, the first-order formula returned is of such a nature that without the algorithm, finding a substitution for the circumscription axiom would be highly unlikely.

7 Conclusion

In this paper, we have presented a general algorithm which transforms second-order formulas into logically equivalent first-order formulas for a large class of second-order formulas. The algorithm has been shown to have a number of attractive properties, including a potentially wide area

for practical application. To support this claim, we have provided a detailed description of the algorithms application to the reduction of circumscription axioms. In addition, we have shown that the algorithm, in its general form, provably subsumes nearly all existing results concerning the reduction of circumscription axioms. In the cases not subsumed, we have shown, via the general methodology for use of the algorithm, how a slight specialization of the algorithm provides a remedy, not only for these particular cases, but for other potential exceptions. In contrast to previous results, the algorithm is more constructive in the sense that it provides a step-by-step method for transforming a formula and it also terminates.

In the future, we plan on investigating specializations of the algorithm where the general methodology proposed may be used in combination with information about the structure of particular domains of interest, such as the domain of action and change, to generate specific heuristics which can be integrated with the preprocessing stage. In addition, we feel that the use of circumscription as a knowledge representation tool deserves reevaluation in light of the results described here. The common view of circumscription as an elegant formalism for conceptual analysis, but one that is difficult to apply practically due to its second-order nature, requires modification if the algorithm can be applied practically, as we believe it can.

References

[1] Ackermann, W. (1935) *Untersuchungen über das Eliminationsproblem der mathematischen Logik*, Mathematische Annalen, 110, 390-413.

[2] Ackermann, W. (1954) *Solvable Cases of the Decision Problem*, North-Holland, Amsterdam.

[3] Etherington, D., Kraus, S., Perlis, D. (1991) *Nonmonotonicity and the Scope of Reasoning*, in: Artificial Intelligence **52**(3), 221-262.

[4] Gabbay, D., Ohlbach, H. J. (1992) *Quantifier Elimination in Second-Order Predicate Logic*, Report of the Max-Planck Institut für Informatik, MPI-I-92-231, Saarbrücken, Germany.

[5] Gelfond, M., Lifschitz, V. (1989) *Compiling Circumscriptive Theories into Logic Programs*, in: Proc. 2nd Int'l Workshop on Non-Monotonic Reasoning, Lecture Notes in Artificial Intelligence, 346, Springer-Verlag, Berlin, 74-99.

[6] Ginsberg, M. L. (1989) *A Circumscriptive Theorem Prover*, in: Artificial Intelligence, **39**, 209-230.

[7] Kolaitis, P., Papadimitriou, C. (1988) *Some Computational Aspects of Circumscription*, in: Proc. AAAI-88, St Paul, MN, 465-469.

[8] Konolige, V. (1994) *Autoepistemic Logic*, in: Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 3 (D.M. Gabbay, C.J. Hogger & J.A. Robinson, eds.), Clarendon Press, Oxford.

[9] Lifschitz, V. (1985) *Computing Circumscription*, in: Proc. 9th IJCAI, Los Angeles, CA, 121-127.

[10] Lifschitz, V. (1988) *Pointwise Circumscription*, in: Readings in Non-Monotonic Reasoning, M. Ginsberg (ed.), Morgan Kaufmann Publishers, Palo Alto, CA, 179-193.

[11] Lifschitz, V. (1994) *Circumscription*, in: Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 3 (D.M. Gabbay, C.J. Hogger & J.A. Robinson, eds.), Clarendon Press, Oxford, 297-352.

[12] Lukaszewicz, W. (1990) *Non-monotonic Reasoning. Formalization of Commonsense Reasoning*, Ellis Horwood Ltd.

[13] Löwenheim, L. (1915) *Über Möglichkeiten im Relativekalkül*, Mathematische Annalen, 137-148.

[14] Poole, D. (1994) *Default Logic*, in: Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 3 (D.M. Gabbay, C.J. Hogger & J.A. Robinson, eds.), Clarendon Press, Oxford.

[15] Przymusinski, T. (1991) *An Algorithm to Compute Circumscription*, in: Artificial Intelligence **38**, 49-73.

[16] Rabinov, A. (1989) *A Generalization of Collapsible Cases of Circumscription (Research Note)*, Artificial Intelligence **38**, 111-117.

[17] Reiter, R. (1980) *A Logic for Default Reasoning*, Artificial Intelligence **13**, 81-131.

[18] Szalas, A. (1993) *On the Correspondence Between Modal and Classical Logic: an Automated Approach*, Journal of Logic and Computation, **3**, 605-620.

[19] Szalas, A. (1994) *On an Automated Translation of Modal Proof Rules into Formulas of the Classical Logic*, Journal of Applied Non-classical Logics, **4**, 119-128.

[20] Van Benthem, J. (1983) *Modal Logic and Classical Logic*, Bibliopolis, Napoli.

[21] Van Benthem, J. (1984): *Correspondence Theory*, in: Handbook of Philosophical Logic, Vol. 2 (D. Gabbay and F. Guenthner, eds.), D. Reidel Pub. Co., 167-247.

A   The Algorithm

The algorithm takes a formula of the form $\exists\Phi.A$, where A is a first-order formula, as an input and returns its first-order equivalent or reports failure[6]. Of course, the algorithm can also be used for formulas of the form $\forall\Phi.A$, since the latter formula is equivalent to $\neg\exists\Phi.\neg A$. Thus, by repeating the algorithm one can deal with formulas containing arbitrarily many second-order quantifiers.

The algorithm consists of three basic phases: (1) preprocessing; (2) preparation for the Ackermann lemma; (3) application of the Ackermann lemma. These phases are described below. It is always assumed that whenever the goal specific for a current phase is reached, then the remaining steps of of the phase are skipped.

A.1   Preprocessing

The purpose of this phase is to transform the formula $\exists\Phi.A$ into a form that separates positive and negative occurrences of the quantified predicate variable Φ. The form we want to obtain is[7]

$$\exists\bar{x}\exists\Phi[(A_1(\Phi)\wedge B_1(\Phi))\vee\cdots\vee(A_n(\Phi)\wedge B_n(\Phi))], \tag{150}$$

where, for each $1\leq i\leq n$, $A_i(\Phi)$ is positive w.r.t. Φ and $B_i(\Phi)$ is negative w.r.t. Φ.[8]

To achieve the goal of this phase, apply the steps below in the following order.

1. Eliminate the connectives $\supset$ and $\equiv$ using the usual definitions. Remove redundant quantifiers. Rename individual variables until all quantified variables are different and no variable is both bound and free. Using the usual equivalences, move the negation connective to the right until all its occurrences immediately proceed atomic formulas.

---

[6] The failure of the algorithm does not mean that the second-order formula at hand cannot be reduced to its first-order equivalent. The problem we are dealing with is not even partially decidable, for first-order definability of the formulas we consider is not an arithmetical notion (se, for instance, [21]).

[7] It should be emphasized that not every formula is reducible into this form.

[8] To increase the strength of the algorithm, it is essential to move as many existentially quantified variables as possible into the prefix of (150).

2. Move universal quantifiers to the right and existential quantifiers to the left applying as long as possible the following equivalences (below $Q \in \{\forall, \exists\}$, $\circ \in \{\vee, \wedge\}$ and B contains no occurrences of variables $\bar{x}$):

- $Q\bar{x}(A(\bar{x}) \circ B) \equiv (Q\bar{x}A(\bar{x})) \circ B$
- $Q\bar{x}(B \circ A(\bar{x})) \equiv B \circ Q\bar{x}A(\bar{x})$.

3. Move to the right the existential quantifiers that are in the scope of universal quantifiers using the equivalences of step 2.

4. Repeat (2) and (3) as long as no new existentially quantified variable can be moved into the prefix.

5. In the matrix of the formula obtained so far, distribute all top-level conjunctions over the disjunctions that occur among their conjuncts. For this purpose, apply the following equivalences:

- $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$
- $(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C)$

If the resulting formula is not in the form (150), then report the failure of the algorithm. Otherwise replace (150) by its equivalent given by

$$\exists \bar{x}(\exists \Phi(A_1(\Phi) \wedge B_1(\Phi)) \vee \cdots \vee \exists \Phi.(A_n(\Phi) \wedge B_n(\Phi))). \tag{151}$$

For each disjunct $\exists \Phi(A_i(\Phi) \wedge B_i(\Phi))$ of (151) try to find its first-order equivalent by performing the next phases to the formula $\exists \Phi(A_i(\Phi) \wedge B_i(\Phi))$. If all the equivalents are obtained, return their disjunction, preceded by the prefix $\exists \bar{x}$, as the output of the algorithm.

The following example illustrates the described phase.

Example A.1 Consider the formula $\exists \Phi[\forall x \exists y(P(y) \vee \exists t(\Phi(t) \vee P(x) \vee R(x, t))) \wedge \exists z \Phi(z) \wedge \exists u \neg \Phi(u)]$. The following lines show the subsequent transformations.

$$
\begin{array}{ll}
\exists \Phi[\forall x \exists y(P(y) \vee \exists t(\Phi(t) \vee P(x) \vee R(x, t))) \wedge \exists z \Phi(z) \wedge \exists u \neg \Phi(u)] & \equiv \quad \text{(by 2)} \\
\exists z u \exists \Phi[\forall x \exists y(P(y) \vee \exists t(\Phi(t) \vee P(x) \vee R(x, t))) \wedge \Phi(z) \wedge \neg \Phi(u)] & \equiv \quad \text{(by 3)} \\
\exists z u \exists \Phi[\forall x (\exists y P(y) \vee \exists t(\Phi(t) \vee P(x) \vee R(x, t))) \wedge \Phi(z) \wedge \neg \Phi(u)] & \equiv \quad \text{(by 2)} \\
\exists z u \exists \Phi[(\exists y P(y) \vee \forall x \exists t(\Phi(t) \vee P(x) \vee R(x, t))) \wedge \Phi(z) \wedge \neg \Phi(u)] & \equiv \quad \text{(by 2)} \\
\exists z u y \exists \Phi[(P(y) \vee \forall x \exists t(\Phi(t) \vee P(x) \vee R(x, t))) \wedge \Phi(z) \wedge \neg \Phi(u)] & \equiv \quad \text{(by 5)} \\
\exists z u y \exists \Phi[(P(y) \wedge \Phi(z) \wedge \neg \Phi(u)) \vee (\forall x \exists t(\Phi(t) \vee P(x) \vee R(x, t)) \wedge \Phi(z) \wedge \neg \Phi(u))] & \blacksquare
\end{array}
$$

A.2 Preparation for the Ackermann Lemma

The goal of this phase is to transform a formula of the form $\exists \Phi(A(\Phi) \wedge B(\Phi))$, where $A(\Phi)$ (resp. $B(\Phi)$) is positive (resp. negative) w.r.t. Φ, into one of the forms (3) or (4) given in Lemma 3.1. Both forms can always be obtained. However, Skolemization is sometimes necessary and unskolemization, which is to be performed in the next phase, may fail. Accordingly, the algorithm performs both transformations. Due to the symmetry of the Ackermann lemma, the steps stated below describe only one of those transformations, namely that leading to the form (3).

1. Transform $A(\Phi)$ into the form

$$pref[(\Phi(\bar{t}_{11}) \vee \cdots \vee \Phi(\bar{t}_{1n_1}) \vee C_1) \wedge \cdots \wedge (\Phi(\bar{t}_{k1}) \vee \cdots \vee \Phi(\bar{t}_{kn_k}) \vee C_k) \wedge D],$$

where $pref$ is a prefix of first-order quantifiers and Φ does not occur in $C_1, \ldots, C_k, D$. This step is always possible by applying the usual technique of obtaining the conjunctive normal form.

2. Transform each conjunct $(\Phi(\bar{t}_{i1})\vee\cdots\vee\Phi(\bar{t}_{in_i})\vee C_i)$, where $n_i > 1$, into its equivalent

$$\exists\bar{x}_i(\forall\bar{y}(\Phi(\bar{y})\vee\bar{x}_i\neq\bar{y}\vee C_i)\wedge(\bar{x}_i=\bar{t}_{i1}\vee\cdots\vee\bar{x}_i=\bar{t}_{in_i}\vee C_i)$$

and move all existential quantifiers into the prefix $pref$.[9]

3. Transform each conjunct $(\Phi(\bar{t}_{i1}\vee C_i)$ into its equivalent $\forall\bar{y}(\Phi(\bar{y})\vee\bar{y}\neq\bar{t}_{i1}\vee C_i)$

4. Remove all existential quantifiers from the prefix $pref$ using the equivalence of Skolem given by

$$\forall\bar{x}.\exists y.A(\bar{x},y,\ldots)\equiv\exists f.\forall\bar{x}.A(\bar{x},y\leftarrow f(\bar{x}),\ldots),\tag{152}$$

where f is a new function variable. After this transformation the input formula takes the form

$$\exists\bar{f}\exists\Phi pref'[\forall\bar{y}(\Phi(\bar{y})\vee\bar{x}_1\neq\bar{y}\vee C_1)\wedge\cdots\wedge\forall\bar{y}(\Phi(\bar{y})\vee\bar{x}_k\neq\bar{y}\vee C_k)\wedge E],\tag{153}$$

where $\bar{f}$ is the tuple of the introduced Skolem functions and $pref'$ only contains universal quantifiers.

5. Transform (153) into its equivalent given by

$$\exists\bar{f}\exists\Phi\forall\bar{y}[\Phi(\bar{y})\vee pref'((\bar{x}_1\neq\bar{y}\vee C_1)\wedge\cdots\wedge(\bar{x}_k\neq\bar{y}\vee C_k))\wedge pref'E.\tag{154}$$

Example A.1 [continued] There are two formulas to be considered in this phase, namely $\exists\Phi(P(y)\wedge\Phi(z)\wedge\neg\Phi(u))$ and $\exists\Phi\forall x\exists t(\Phi(t)\vee P(x))\wedge\Phi(z)\wedge\neg\Phi(u)$. We apply phase 2 to the former of the above formulas.

$$\begin{array}{lll}\exists\Phi(P(y)\wedge\Phi(z)\wedge\neg\Phi(u)) & \equiv & \text{(by 2)}\\ \exists\Phi(P(y)\wedge\forall r(\Phi(r)\vee z\neq r)\wedge\neg\Phi(u))\end{array}$$

Applying phase 2 to the second formula proceeds as follows.

$$\begin{array}{lll}\exists\Phi\forall x\exists t(\Phi(t)\vee P(x)\vee R(x,t))\wedge\Phi(z)\wedge\neg\Phi(u) & \equiv & \text{(by 3)}\\ \exists\Phi\forall x\exists t\forall r(\Phi(r)\vee r\neq t\vee P(x)\vee R(x,t))\wedge\forall r(\Phi(r)\vee z\neq r)\wedge\neg\Phi(u) & \equiv & \text{(by 4)}\\ \exists f\exists\Phi\forall x\forall r(\Phi(r)\vee r\neq f(x)\vee P(x)\vee R(x,f(x)))\wedge\forall r(\Phi(r)\vee z\neq r)\wedge\neg\Phi(u) & \equiv & \text{(by 5)}\\ \exists f\exists\Phi\forall r[\Phi(r)\vee(\forall x(r\neq f(x)\vee P(x)\vee R(x,f(x)))\wedge z\neq r)]\wedge\neg\Phi(u) & \blacksquare\end{array}$$

A.3 Application of the Ackermann Lemma

The goal of this phase is to eliminate the second-order quantification over Φ, applying the Ackermann lemma, and then to unskolemize the introduced function variables. The phase consists of the following two steps.

1. Apply the Ackermann lemma to the formula (154). The resulting formula is of the form

$$\exists\bar{f}[pref E(\neg\Phi\leftarrow pref((\bar{x}_1\neq\bar{y}\vee C_1)\wedge\cdots\wedge(\bar{x}_k\neq\bar{y}\vee C_k)))]$$

2. Try to remove all existential quantifiers over function variables using the equivalence (152). If this is impossible and both transformations mentioned in the beginning of section A.2 failed, report failure of the algorithm; otherwise try the remaining possibility.

---

[9]This step is justified by equivalence (16) of Proposition 1.1.

Example A.1 [continued] We apply phase 3 for the pair of formulas obtained as the result of phase 2.

$$\exists\Phi(P(y)\wedge\forall r(\Phi(r)\vee z\neq r)\wedge\neg\Phi(u)) \quad\equiv\quad (\text{by } 1)$$
$$P(y)\wedge z\neq u.$$

$$\exists f\exists\Phi\forall r[\Phi(r)\vee(\forall x(r\neq f(x)\vee P(x)\vee R(x,f(x)))\wedge z\neq r)]\wedge\neg\Phi(u) \quad\equiv\quad (\text{by } 1)$$
$$\exists f\forall x(u\neq f(x)\vee P(x)\vee R(x,f(x)))\wedge z\neq u \quad\equiv\quad (\text{by } 2)$$
$$\forall x\exists t(u\neq t\vee P(x)\vee R(x,t))\wedge z\neq u.$$

A.4 Simplification

The formula obtained as the result of the previous phase can often be substantially simplified using Proposition 1.1 (15), or its generalization (18). The simplification phase consists of one step. In the formula obtained after successfully performing phase 3,

1. replace each subformula of the form $\forall\bar{x}(A(\bar{t}\leftarrow\bar{x})\vee\bar{x}\neq\bar{t})$ by $A(\bar{t})$, and

2. replace each subformula of the form $\forall\bar{x}(\bar{x}\neq\bar{t}_1\wedge\cdots\wedge\bar{x}\neq\bar{t}_n)\vee A(\bar{t}_1\leftarrow\bar{x}))$ by $A(\bar{t}_1)\wedge\cdots\wedge A(\bar{t}_n)$.

Example A.1 [continued] Since the simplification phase is inapplicable to the formulas obtained in phase 3, the first-order equivalent of the input formula we finally obtain is

$$\exists z\exists u\exists y(P(y)\wedge z\neq u\wedge\forall x\exists t(u\neq t\vee P(x)\vee R(x,t))\wedge z\neq u).$$

∎

A.5 Remarks about the Algorithm

- The algorithm always terminates and is sound, i.e. the output first-order formula, if obtained, is equivalent to the input second-order formula.

- Although the algorithm may seem a bit complex, the calculations it describes may be performed without any computer support.

- For the sake of clarity the algorithm is not presented in its most efficient form. The possible directions for its optimization follow from the examples presented throughout the paper.

- Observe that one usually deals with the elimination problem over a first-order definable class of models. In such cases it is sometimes possible to considerably simplify the input formula before running the algorithm (see Section 4.1.2). Such a possibility can be considered as an additional heuristics in the preprocessing phase.