Journal of Automated Reasoning, 18, 297–334 (1997)
© 1997 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Computing Circumscription Revisited: A Reduction Algorithm

PATRICK DOHERTY* patdo@ida.liu.se
Department of Computer and Information Science, Linköping University, S-581 83 Linköping, Sweden

WITOLD ŁUKASZEWICZ AND ANDRZEJ SZAŁAS** witlu/szalas@mimuw.edu.pl
Institute of Informatics, Warsaw University, 02-097 Warsaw, Banacha 2, Poland

Abstract. In recent years, a great deal of attention has been devoted to logics of *common-sense* reasoning. Among the candidates proposed, circumscription has been perceived as an elegant mathematical technique for modeling non-monotonic reasoning, but difficult to apply in practice. The major reason for this is the second-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, second-order formulas into equivalent first-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 certain circumscription axioms to first-order formulas. The algorithm takes as input an arbitrary second-order formula and either returns as output an equivalent first-order formula, or terminates with failure. The class of second-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, compare it with existing approaches, and provide formal subsumption results.

Keywords: Circumscription, Non-monotonic Reasoning, Quantifier Elimination, Common-sense Reasoning

1. Introduction and Preliminaries

In recent years, a great deal of attention has been devoted to logics for *common-sense reasoning*. Among the candidates proposed, circumscription ([18], [15]), has been perceived as an elegant mathematical technique for modeling non-monotonic reasoning, but difficult to apply in practice. Practical application of circumscription is made difficult due to two problems. The first concerns the difficulty in finding the proper circumscriptive policy for particular domains of interest. The second concerns the second-order nature of circumscription axioms and the difficulty in finding proper substitutions of predicate expressions for predicate variables so the axioms can be used for making inferences. There have been a number of proposals

* supported in part by Swedish Research Council for Engineering Sciences (TFR) grant 93-270.
** supported in part by KBN grant 3 P406 019 06.

for dealing with the second problem ranging from compiling circumscriptive theories into logic programs [9], to developing specialized inference methods for such theories ([10],[19]).

A third alternative is to focus on the more general problem of finding methods for reducing second-order formulas to logically equivalent first-order formulas, where possible. Although some progress has been made using this approach, the class of second-order circumscription formulas shown to be reducible is not as large as one might desire, the reduction methods proposed are somewhat isolated relative to each other and, most importantly, the existing reduction theorems generally lack algorithmic procedures for doing the reductions.

In this article, we provide a general method which can be used in an algorithmic manner to reduce certain classes of second-order circumscription axioms to logically equivalent first-order formulas. The algorithm takes as input an arbitrary second-order formula and either returns as output an equivalent first-order formula, or terminates with failure. Of course, failure does not imply that there is no first-order equivalent for the input, only that the algorithm can not find one. The class of second-order formulas, and analogously the class of circumscriptive theories which can be reduced, provably subsumes those covered by existing results. The algorithm can be applied successfully to circumscriptive theories which may include mixed quantifiers (some involving Skolemization), variable constants, n-ary tuples of minimized and varied predicates, separable, separated and in some cases, non-separated formulas, and formulas with n-ary predicate variables, among others. In addition to demonstrating the algorithm by applying it to some of these theories, we analyze its strength and provide formal subsumption results based on comparison with existing approaches.

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 discuss related work. In Section 8, 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 V_i 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. [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 $Circ(T; \overline{P}; \overline{S})$, is the sentence

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

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

Definition. 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. [Separated and Separable Formulas] 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 *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 substantially generalizes Ackermann's technique. The equivalence (17) is a second-order formulation of the Skolem reduction (see [3]). 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 [15], 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 [13] we are presented with the following result concerning the reduction of second-order circumscription into first-order logic.

THEOREM 1 *If $T(\overline{P})$ is a separable formula of the form in Equation (2), then $Circ(T; \overline{P}; ())$ 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 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 [13]).

THEOREM 2 *$Circ(T(\overline{P}, \overline{S}); \overline{P}; \overline{S})$ is equivalent to $T(\overline{P}, \overline{S}) \wedge Circ(\exists \overline{\Phi}.\ T(\overline{P}, \overline{\Phi}); \overline{P}; ())$.*

Although Theorem 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 [13]).

THEOREM 3 *If $T(\overline{\Phi})$ is equivalent to Equation (2), 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 [14]. The details are these.

THEOREM 4 *If all occurrences of $\overline{P} = (P_1, \ldots, P_n)$ are positive, then $Circ(T; \overline{P}; ())$ 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 [15], Lifschitz also formulated the following theorem.

THEOREM 5 *If $T(P)$ is a first-order sentence separated w.r.t. P then $Circ(T(P); P; ())$ is equivalent to a first-order sentence.*

2.2. Rabinov's Result

Rabinov [20] provides the following result which subsumes the earlier results of Lifschitz (excluding Theorem 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 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 $Circ(T; P; ())$ is equivalent to

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

where $R_i(P)$ stands for $\neg M_i(P \cap Q_i) \vee Circ(M_i(P); P; ())$ 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 [12], Kolaitis & Papadimitriou present the following result.

THEOREM 7 *If T is a first-order existential sentence, then $Circ(T; \overline{P}; ())$ is equivalent to a first-order sentence.*

2.4. Second-Order Monadic Logic

The following result is due to Löwenheim (see [17]).

THEOREM 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 [8]. It is formulated as follows:

Definition. 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). 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.

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 [22], 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 [22].

LEMMA 1 (ACKERMANN LEMMA) *Let Φ be a predicate variable and $A(\bar{x},\bar{z})$, $B(\Phi)$ 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 right-hand 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), is also used in the algorithm.

PROPOSITION 2 *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 occurring 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 [22]. ■

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 Ackermann's Lemma; (3) application of the Ackermann Lemma; and (4) simplification. These phases are described below. It is always assumed that (i) whenever the goal specific for a current phase is reached, then the remaining steps of the phase are skipped, (ii) every time the equivalence (5) of Proposition 2 is applicable, it should be applied.

(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 occurs 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. (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. (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 Ackermann's 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 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.

(3) **Application of Ackermann's Lemma.** The goal of this phase is to eliminate the second-order quantification over Φ, by applying Ackermann's Lemma, and then try to unskolemize the function variables possibly introduced. This latter step employs the equivalence (17) from Proposition 1.

(4) **Simplification.** Generally, application of Ackermann's Lemma in step (3) often involves the use of equivalence (15) in Proposition 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,
- B is positive w.r.t. Φ, and
- C is negative w.r.t. Φ.

Then, the following proposition holds.

PROPOSITION 3 *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.

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

PROPOSITION 4 *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.

Proof: The algorithm can fail for two reasons. (1) The input can not be put in separated form. (2) If skolemization is necessary, the algorithm may not be able to unskolemize before termination. Since the input is of the form (8), it is separated. In the two cases above, skolemization is not necessary. In both cases, B and C are universal, so no skolemization is necessary relative to $pref$ and $pref'$, respectively. The only way for new existential quantifiers to be introduced during the steps in the algorithm is if any of the conjuncts in B or C contain more than one occurrence of Φ or $\neg\Phi$, respectively. By assumption, this is not the case. ■

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 [12], [13], [14], [20]. 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 9 *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 and $N_0(P)$ is negative w.r.t. P, then the algorithm eliminates the second-order quantifiers from $Circ(T;P;())$.[4]

Proof: The negated second-order part of $Circ(T;P;())$ 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 2. 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. ■

It is worth noting that Lifschitz should probably be credited with rediscovering the positive form of Ackermann's Lemma (Lemma 1, Equation 3). One can observe this is the case by combining Equation 2 in the definition of separability with Theorem 3.

The following theorem shows that the algorithm eliminates second-order quantification in the case of existential theories considered in [12].

THEOREM 10 *If T is a first-order existential sentence, then the algorithm eliminates second-order quantification from $Circ(T; \overline{P}; ())$.*

Proof: The negated second-order part of $Circ(T; P; ())$ 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

$$\begin{array}{ll}
\exists \Phi[T(\Phi) \wedge [\Phi \leq P] \wedge \neg[P \leq \Phi]] & \equiv \\
\exists \Phi \exists \bar{x}[T'(\Phi) \wedge [\Phi \leq P] \wedge \neg[P \leq \Phi]] & \equiv \\
\exists \bar{x} \exists \Phi[T'(\Phi) \wedge \forall \bar{y}(\neg\Phi(\bar{y}) \vee P(\bar{y})) \wedge \exists \bar{z}(P(\bar{z}) \wedge \neg\Phi(\bar{z}))] & \equiv \\
\exists \bar{x} \bar{z} \exists \Phi[T'(\Phi) \wedge \forall \bar{y}(\neg\Phi(\bar{y}) \vee P(\bar{y})) \wedge P(\bar{z}) \wedge \neg\Phi(\bar{z})]. &
\end{array}$$

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, each disjunct can be transformed into the form required in Ackermann's Lemma. ■

THEOREM 11 *If T is a first-order monadic sentence, then the algorithm eliminates second-order quantification from $Circ(T; \overline{P}; \overline{S})$.*

Proof: The reduction that works here is given in Ackermann [[2], pp. 41-46], which uses Proposition 1 and a weaker form of Lemma 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-line arrows. The arrow pointing towards Th 5 is open to signify that although the DLS algorithm in its general form does not fully subsume Theorem 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.

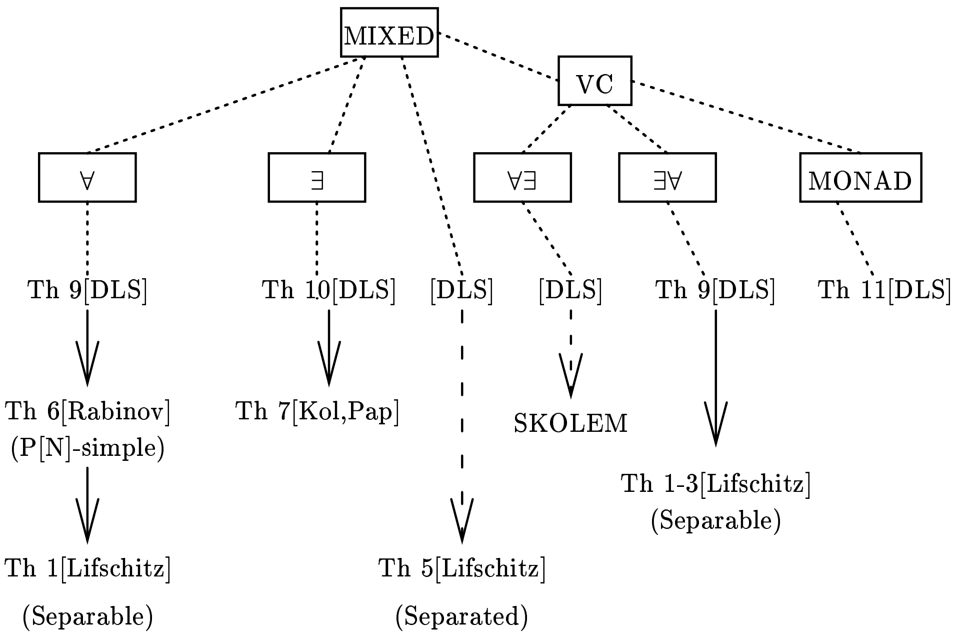


Figure 1. Subsumption Results.

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:

- 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 [22]). However, when one assumes that the accessibility relation is transitive, the elimination is possible, since McKinsey together with transitivity is first-order definable (see [4]).

The same situation may occur when one computes circumscription. Consider the theorem of Lifschitz (Theorem 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 5 the sentence $T(P)$ is assumed to be separated, i.e. it is of the form $T_1(P) \land T_2(P)$, where $T_1(P)$ is positive w.r.t. P and $T_2(P)$ is negative w.r.t. P. Thus $Circ(T_1 \land T_2; P; ())$ is equivalent to

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

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

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

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).

4.1.3. Comparison with SCAN

The SCAN algorithm 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. It is difficult to provide a formal comparison between DLS and SCAN since no syntactic characterization of formulas accepted by SCAN is known. What we can do is provide several examples where DLS fares better than SCAN and mention a method to construct an example where SCAN would fare better than DLS due to its use of optimization techniques associated with the resolution theorem proving method.

Observe that some examples where SCAN fared better than the algorithm given in [22] were known. On the other hand, in the present paper the algorithm described in [22] is strengthened by adding the equivalence (16) of Proposition 1. An additional advantage of our algorithm is that it always terminates, while SCAN may loop.

In the first example, DLS, when given the formula

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

terminates, while for SCAN it does not.

In the second example, DLS, when given the formula

$$\exists x\exists y(P(x) \vee P(y) \vee R(x,y)) \wedge \forall x\exists y(\neg P(x) \vee \neg P(y) \vee S(x,y)),$$

terminates with success and returns a logically equivalent first-order formula, while SCAN fails due to unskolemization problems.

The obvious question arises whether there is an example where SCAN terminates with a logically equivalent first-order formula while DLS terminates with failure. One way to construct such an example would be to provide an input formula where one or more subformulas are subsumed by another part of the formula. In this case, SCAN would delete these clauses using its subsumption deletion heuristic. If these clauses were set up to provide unskolemization problems, then because DLS has no subsumption deletion heuristic, it could possibly fail to unskolemize the example. Of course, it should be possible to extend the DLS algorithm with additional optimization heuristics, but we save this for future research.

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 2 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 occurring situations the formula obtained as the result of applying the Ackermann Lemma can be substantially simplified by using Proposition 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 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 conclusion, 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]

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

and then its form is

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

after application of Ackermann's 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 6.6 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 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 get lost in details. For example, step (2) in the previous Section 3.1 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: [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

$$\begin{aligned}&Circ(\Gamma(Ab, On); Ab; On) \equiv \\ &\quad \Gamma(Ab, On) \wedge \forall\Phi\forall\Psi[[\Gamma(\Phi, \Psi) \wedge \Phi \leq Ab] \supset [Ab \leq \Phi]],\end{aligned} \tag{12}$$

where

$$\begin{aligned}\Gamma(\Phi, \Psi) \equiv\ & [b1 \neq b2 \wedge B(b1) \wedge B(b2) \wedge \neg\Psi(b1)] \\ & \wedge\forall x(B(x) \wedge \neg\Phi(x) \supset \Psi(x)) && (13)\\ \Phi \leq Ab \equiv\ & \forall x(\Phi(x) \supset Ab(x)) && (14)\\ Ab \leq \Phi \equiv\ & \forall x(Ab(x) \supset \Phi(x)). && (15)\end{aligned}$$

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

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

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

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

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

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

Application of Ackermann's Lemma. Applying Ackermann's Lemma to (21), we obtain

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

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

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

Preparation for Ackermann's Lemma. Applying Proposition 1 (15) to $\neg\Psi(b1)$ in (23), we obtain

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

Application of Ackermann's Lemma. We apply Ackermann's Lemma to (24), obtaining

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

Simplification. Using Proposition 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

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

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

$$\begin{aligned}&Circ(\Gamma(Ab, On); Ab; On) \equiv \Gamma(Ab, On) \wedge \forall z[(B(b1) \wedge \neg Ab(b1)) \vee\\ &\quad (z = b1 \wedge B(z)) \vee \neg Ab(z) \vee b1 = b2 \vee \neg B(b1) \vee \neg B(b2)]. \end{aligned} \tag{28}$$

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

$$Ab(b1) \supset \forall z(\neg Ab(z) \vee (z = b1 \wedge B(z))), \tag{29}$$

and thus implies

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

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 Birthday Example

Example: [Birthday Example] This example contains both existentially quantified and universal formulas. In addition, it contains both unary and binary predicates.

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

$$\begin{aligned}&[\exists x \exists y(B(y) \wedge F(x,y) \wedge \neg G(x,y))]\\ &\quad \wedge[\forall x \forall y(B(y) \wedge F(x,y) \wedge \neg Ab(x,y) \supset G(x,y))], \end{aligned} \tag{31}$$

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

$$\begin{aligned}&Circ(\Gamma(Ab, G); Ab; G) \equiv\\ &\quad \Gamma(Ab, G) \wedge \forall \Phi \forall \Psi[\Gamma(\Phi, \Psi) \wedge [\Phi \leq Ab] \supset [Ab \leq \Phi]], \end{aligned} \tag{32}$$

where

$$\begin{aligned} \Gamma(\Phi,\Psi) \equiv & \quad [\exists x \exists y(B(y) \wedge F(x,y) \wedge \neg\Psi(x,y))] \\ & \wedge[\forall x \forall y(B(y) \wedge F(x,y) \wedge \neg\Phi(x,y) \supset \Psi(x,y))] \qquad (33) \\ \Phi \leq Ab \equiv & \quad \forall x \forall y[\Phi(x,y) \supset Ab(x,y)] \qquad (34) \\ Ab \leq \Phi \equiv & \quad \forall x \forall y[Ab(x,y) \supset \Phi(x,y)]. \qquad (35) \end{aligned}$$

In the following, we will reduce

$$\forall\Phi\forall\Psi[\Gamma(\Phi,\Psi) \wedge [\Phi \leq Ab] \supset [Ab \leq \Phi]] \qquad (36)$$

in (32). Negating (36), we obtain

$$\exists\Phi\exists\Psi[\Gamma(\Phi,\Psi) \wedge [\Phi \leq Ab] \wedge \neg[Ab \leq \Phi]]. \qquad (37)$$

We remove Ψ first.

Preprocessing. Replacing $\Gamma(\Phi,\Psi)$, $\Phi \leq Ab$ and $Ab \leq \Phi$ by their equivalents given by (33)–(35), eliminating $\supset$, renaming individual variables and moving existential quantifiers over individual variables to the left, we obtain

$$\begin{aligned} &\exists x \exists y \exists q \exists r \exists\Phi\exists\Psi[B(y) \wedge F(x,y) \wedge \neg\Psi(x,y) \wedge \forall u \forall z(\neg B(z) \vee \neg F(u,z) \vee \\ &\quad \Phi(u,z) \vee \Psi(u,z)) \wedge \forall s \forall t(\neg\Phi(s,t) \vee Ab(s,t)) \wedge Ab(q,r) \wedge \neg\Phi(q,r)]. \qquad (38) \end{aligned}$$

Preparation for Ackermann's Lemma. (38) is in the form suitable for application of Ackermann's Lemma. To see this, we rewrite it as

$$\begin{aligned} &\exists x \exists y \exists q \exists r \exists\Phi\exists\Psi\forall u \forall z[(\Psi(u,z) \vee \neg B(z) \vee \neg F(u,z) \vee \Phi(u,z)) \wedge \neg\Psi(x,y) \\ &\quad \wedge B(y) \wedge F(x,y) \wedge \forall s \forall t(\neg\Phi(s,t) \vee Ab(s,t)) \wedge Ab(q,r) \wedge \neg\Phi(q,r)]. \qquad (39) \end{aligned}$$

Application of Ackermann's Lemma. Applying Ackermann's Lemma to (39), we obtain

$$\begin{aligned} &\exists x \exists y \exists q \exists r \exists\Phi[(\neg B(y) \vee \neg F(x,y) \vee \Phi(x,y)) \wedge B(y) \wedge F(x,y) \\ &\quad \wedge \forall s \forall t(\neg\Phi(s,t) \vee Ab(s,t)) \wedge Ab(q,r) \wedge \neg\Phi(q,r)]. \qquad (40) \end{aligned}$$

We now remove Φ in (40).

Preprocessing. (40) is in the form which is the goal of this phase.

Preparation for Ackermann's Lemma. Using Proposition 1 (15), we replace (40) by

$$\begin{aligned} &\exists x \exists y \exists q \exists r \exists\Phi\forall v \forall w[(\Phi(v,w) \vee v \neq x \vee w \neq y \vee \neg B(y) \vee \neg F(x,y)) \\ &\wedge \forall s \forall t(\neg\Phi(s,t) \vee Ab(s,t)) \wedge \neg\Phi(q,r) \wedge B(y) \wedge F(x,y) \wedge Ab(q,r)]. \qquad (41) \end{aligned}$$

Application of Ackermann's Lemma. Applying Ackermann's Lemma to (41), we obtain

$$\begin{aligned} &\exists x \exists y \exists q \exists r \forall s \forall t[(s \neq x \vee t \neq y \vee \neg B(y) \vee \neg F(x,y) \vee Ab(s,t)) \\ &\quad \wedge(q \neq x \vee r \neq y \vee \neg B(y) \vee \neg F(x,y)) \wedge B(y) \wedge F(x,y) \wedge Ab(q,r)]. \qquad (42) \end{aligned}$$

Simplification. We replace (42) by

$$\exists x \exists y \exists q \exists r[(\neg B(y) \vee \neg F(x,y) \vee Ab(x,y)) \\ \wedge(q \neq x \vee r \neq y \vee \neg B(y) \vee \neg F(x,y)) \wedge B(y) \wedge F(x,y) \wedge Ab(q,r)]. \quad (43)$$

Negating (43), we obtain

$$\forall x \forall y \forall q \forall r[(B(y) \wedge F(x,y) \wedge \neg Ab(x,y)) \\ \vee(q = x \wedge r = y \wedge B(y) \wedge F(x,y)) \vee \neg B(y) \vee \neg F(x,y) \vee \neg Ab(q,r)]. \quad (44)$$

(44) is logically equivalent to

$$\forall x \forall y \forall q \forall r[\neg(B(y) \wedge F(x,y)) \vee ((B(y) \wedge F(x,y)) \wedge \\ (\neg Ab(x,y) \vee (q = x \wedge r = y))) \vee \neg Ab(q,r)], \quad (45)$$

which is equivalent to

$$\forall x \forall y \forall q \forall r[\neg(B(y) \wedge F(x,y)) \vee \neg Ab(x,y) \vee (q = x \wedge r = y) \vee \neg Ab(q,r)]. \quad (46)$$

The first-order formula (46) is logically equivalent to the second-order formula (36). Consequently,

$$Circ(\Gamma(Ab,G); Ab; G) \equiv \Gamma(Ab,G) \wedge \\ \forall x \forall y \forall q \forall r[\neg(B(y) \wedge F(x,y)) \vee \neg Ab(x,y) \vee (q = x \wedge r = y) \vee \neg Ab(q,r)]. \quad (47)$$

A more informative sentence, equivalent to (46), is

$$\forall x \forall y \forall q \forall r[Ab(x,y) \wedge Ab(q,r) \wedge B(y) \wedge F(x,y) \supset (q = x \wedge r = y)]. \quad (48)$$

(48), 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. □

6.3. The Hospital Example

Example: [Hospital Example] Let Γ be the theory

$$[\forall x \exists y(Ab(x,y) \supset H(x,y))] \wedge [\forall x \exists y(\neg Ab(x,y) \supset H(x,y))]. \quad (49)$$

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

$$Circ(\Gamma; Ab; H) \equiv \Gamma \wedge \forall \Phi \forall \Psi[\Gamma(\Phi,\Psi) \wedge [\Phi \leq Ab] \supset [Ab \leq \Phi]], \quad (50)$$

where

$$\Gamma(\Phi,\Psi) \equiv \forall x \exists y[\Phi(x,y) \supset \Psi(x,y)] \wedge \forall x \exists y[\neg\Phi(x,y) \supset \Psi(x,y)] \quad (51)$$

$$\Phi \leq Ab \equiv \forall x \forall y[\Phi(x,y) \supset Ab(x,y)] \quad (52)$$

$$Ab \leq \Phi \equiv \forall x \forall y[Ab(x,y) \supset \Phi(x,y)]. \quad (53)$$

In the following, we will reduce

$$\forall\Phi\forall\Psi[\Gamma(\Phi,\Psi)\wedge[\Phi\leq Ab]\supset[Ab\leq\Phi]] \tag{54}$$

in (50). Negating (54), we obtain

$$\exists\Phi\exists\Psi[\Gamma(\Phi,\Psi)\wedge[\Phi\leq Ab]\wedge\neg[Ab\leq\Phi]]. \tag{55}$$

Since we will try to remove Φ first, we replace (55) by its equivalent given by

$$\exists\Psi\exists\Phi[\Gamma(\Phi,\Psi)\wedge[\Phi\leq Ab]\wedge\neg[Ab\leq\Phi]]. \tag{56}$$

Preprocessing. Replacing $\Gamma(\Phi,\Psi)$, $\Phi\leq Ab$ and $Ab\leq\Phi$ by their equivalents given by (51)–(53), eliminating $\supset$ and renaming individual variables, we obtain

$$\begin{aligned}&\exists\Psi\exists\Phi[\forall x\exists y(\neg\Phi(x,y)\vee\Psi(x,y))\wedge\forall q\exists r(\Phi(q,r)\vee\Psi(q,r))\\&\quad\wedge\forall u\forall v(\neg\Phi(u,v)\vee Ab(u,v))\wedge\exists s\exists t(Ab(s,t)\wedge\neg\Phi(s,t))].\end{aligned} \tag{57}$$

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

$$\begin{aligned}&\exists s\exists t\exists\Psi\exists\Phi[\forall q\exists r(\Phi(q,r)\vee\Psi(q,r))\wedge\forall x\exists y(\neg\Phi(x,y)\vee\Psi(x,y))\\&\qquad\wedge\forall u\forall v(\neg\Phi(u,v)\vee Ab(u,v))\wedge Ab(s,t)\wedge\neg\Phi(s,t)].\end{aligned} \tag{58}$$

Preparation for Ackermann's Lemma. Using Proposition 1 (15), we replace $\Phi(q,r)$ in (58) by $\forall z\forall w(\Phi(z,w)\vee z\neq q\vee w\neq r)$. This results in

$$\begin{aligned}&\exists s\exists t\exists\Psi\exists\Phi[\forall q\exists r(\forall z\forall w(\Phi(z,w)\vee z\neq q\vee w\neq r)\vee\Psi(q,r))\\&\wedge\forall x\exists y(\neg\Phi(x,y)\vee\Psi(x,y))\wedge\forall u\forall v(\neg\Phi(u,v)\vee Ab(u,v))\wedge Ab(s,t)\wedge\neg\Phi(s,t)]\end{aligned} \tag{59}$$

which is equivalent to

$$\begin{aligned}&\exists s\exists t\exists\Psi\exists\Phi[\forall q\exists r\forall z\forall w(\Phi(z,w)\vee z\neq q\vee w\neq r\vee\Psi(q,r))\\&\wedge\forall x\exists y(\neg\Phi(x,y)\vee\Psi(x,y))\wedge\forall u\forall v(\neg\Phi(u,v)\vee Ab(u,v))\wedge Ab(s,t)\wedge\neg\Phi(s,t)].\end{aligned} \tag{60}$$

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)

$$\begin{aligned}&\exists s\exists t\exists\Psi\exists f\exists\Phi\forall z\forall w[(\Phi(z,w)\vee\forall q(z\neq q\vee w\neq f(q)\vee\Psi(q,f(q))))\\&\wedge\forall x\exists y(\neg\Phi(x,y)\vee\Psi(x,y))\wedge\forall u\forall v(\neg\Phi(u,v)\vee Ab(u,v))\wedge Ab(s,t)\wedge\neg\Phi(s,t)].\end{aligned} \tag{61}$$

Application of Ackermann's Lemma. Applying Ackermann's Lemma to (61), we obtain

$$\begin{aligned}&\exists s\exists t\exists f\exists\Psi[\forall x\exists y(\forall q(x\neq q\vee y\neq f(q)\vee\Psi(q,f(q)))\vee\Psi(x,y))\\&\qquad\wedge\forall u\forall v(\forall q(u\neq q\vee v\neq f(q)\vee\Psi(q,f(q)))\vee Ab(u,v))\\&\qquad\qquad\wedge Ab(s,t)\wedge\forall q(s\neq q\vee t\neq f(q)\vee\Psi(q,f(q)))]\end{aligned} \tag{62}$$

which is equivalent to

$$\begin{aligned}&\exists s\exists t\exists f\exists\Psi\forall x\exists y\forall q\forall u\forall v[(x\neq q\vee y\neq f(q)\vee\Psi(q,f(q))\vee\Psi(x,y))\\&\qquad\wedge(u\neq q\vee v\neq f(q)\vee\Psi(q,f(q))\vee Ab(u,v))\\&\qquad\wedge Ab(s,t)\wedge(s\neq q\vee t\neq f(q)\vee\Psi(q,f(q)))].\end{aligned} \tag{63}$$

Since all occurrences of Ψ in each conjunct in (63) are positive, all the conjuncts including Ψ, together with $\exists\Psi$, can be removed by Proposition 2. This yields

$$\exists s \exists t \exists f \exists \Psi \forall x \exists y \forall q \forall u \forall v . Ab(s,t) \tag{64}$$

which reduces to

$$\exists s \exists t Ab(s,t). \tag{65}$$

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

$$\forall s \forall t \neg Ab(s,t). \tag{66}$$

The first-order formula (66) is logically equivalent to the second-order formula (54). Consequently,

$$Circ(\Gamma; Ab; H) \equiv \Gamma \wedge \forall s \forall t . \neg Ab(s,t). \tag{67}$$

which implies

$$\forall x \exists y H(x,y). \tag{68}$$

□

6.4. The Vancouver Example

This is a variant of an example from Reiter [21]. Rather than using the function *city* as Reiter does, we will use a relation $C(x,y)$ with suitable axioms.

Example: [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,

$$\forall x (\neg Ab(x) \supset city(x) = city(wife(x))) \tag{69}$$

is replaced with

$$\forall x \forall y \forall z (\neg Ab(x) \wedge C(x,y) \wedge C(wife(x), z) \supset y = z). \tag{70}$$

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

$$\forall x \forall y \forall z (C(x,y) \wedge C(x,z) \supset y = z). \tag{71}$$

We do not require that all people live in cities, i.e. we reject the axiom[6]

$$\forall x \exists y C(x,y). \tag{72}$$

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

$$[\forall x \forall y \forall z(\neg Ab(x) \wedge C(x,y) \wedge C(wife(x),z) \supset y=z)] \wedge \\ [\forall x \forall y \forall z(C(x,y) \wedge C(x,z) \supset y=z)]. \tag{73}$$

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

$$Circ(\Gamma(Ab,C); Ab; C) \equiv \Gamma(Ab,C) \wedge \forall \Phi \forall \Psi [\Gamma(\Phi,\Psi) \wedge [\Phi \leq Ab] \supset [Ab \leq \Phi], \tag{74}$$

where

$$\Gamma(\Phi,\Psi) \equiv [\forall x \forall y \forall z(\neg \Phi(x) \wedge \Psi(x,y) \wedge \Psi(wife(x),z) \supset y=z)] \wedge \\ [\forall x \forall y \forall z(\Psi(x,y) \wedge \Psi(x,z) \supset y=z)]. \tag{75}$$

$$\Phi \leq Ab \equiv \forall x(\Phi(x) \supset Ab(x)) \tag{76}$$

$$Ab \leq \Phi \equiv \forall x(Ab(x) \supset \Phi(x)). \tag{77}$$

In the following, we will reduce

$$\forall \Phi \forall \Psi [\Gamma(\Phi,\Psi) \wedge [\Phi \leq Ab] \supset [Ab \leq \Phi]] \tag{78}$$

in (74). Negating (78), we obtain

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

We start by removing Ψ.

Preprocessing. Replacing $\Gamma(\Phi,\Psi)$, $\Phi \leq Ab$ and $Ab \leq \Phi$ by their equivalents given by (75)–(77), eliminating $\supset$ and renaming individual variables, we obtain

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

Note that Ψ can be removed directly using Proposition 2, since all instances of Ψ occurring in (80) are negative. To prepare the latter formula into the form suitable for the application of Proposition 2, we move all quantifiers ranging over individual variables to the left. This results in

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

Applying Proposition 2 to (81), we obtain

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

which reduces to

$$\exists \Phi \forall s \exists t[(\neg \Phi(s) \vee Ab(s)) \wedge Ab(t) \wedge \neg \Phi(t). \tag{83}$$

We next try to remove Φ. Again, this can be done directly using Proposition 2, since all instances of Φ in (83) are negative. This results in

$$\forall s \exists t Ab(t) \tag{84}$$

which is equivalent to

$$\exists t Ab(t). \tag{85}$$

Taking the negation of (85) results in

$$\forall t \neg Ab(t). \tag{86}$$

The first-order formula (86) is logically equivalent to the second-order formula (78). Consequently,

$$Circ(\Gamma; Ab; C) \equiv \Gamma \wedge \forall t \neg Ab(t). \tag{87}$$

□

6.5. 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: [Preprocessing Example]
Let Γ be the theory

$$P(a) \supset P(b). \tag{88}$$

The circumscription of Γ, with P minimized is

$$Circ(\Gamma; P; ()) \equiv \Gamma \wedge \forall\Phi[\Gamma(\Phi)[\wedge\Phi \leq P] \supset [P \leq \Phi]], \tag{89}$$

where

$$\begin{aligned} \Gamma(\Phi) \equiv &\quad \Phi(a) \supset \Phi(b), & (90)\\ \Phi \leq Ab \equiv &\ \forall x(\Phi(x) \supset P(x)), & (91)\\ Ab \leq \Phi \equiv &\ \forall x(P(x) \supset \Phi(x)). & (92) \end{aligned}$$

In the following, we will reduce

$$\forall\Phi[\Gamma(\Phi) \wedge [\Phi \leq Ab] \supset [Ab \leq \Phi]] \tag{93}$$

in (89). Negating (93), we obtain

$$\exists\Phi[\Gamma(\Phi) \wedge [\Phi \leq Ab] \wedge \neg[Ab \leq \Phi]]. \tag{94}$$

Preprocessing. Replacing $\Gamma(\Phi, \Psi)$, $\Phi \leq Ab$ and $Ab \leq \Phi$ by their equivalents given by (90)–(92), eliminating $\supset$ and renaming individual variables, we obtain

$$\exists\Phi[(\neg\Phi(a) \vee \Phi(b)) \wedge \forall x(\neg\Phi(x) \vee P(x)) \wedge \exists y(P(y) \wedge \neg\Phi(y))]. \tag{95}$$

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

$$\exists y \exists\Phi[(\neg\Phi(a) \vee \Phi(b)) \wedge \forall x(\neg\Phi(x) \vee P(x)) \wedge P(y) \wedge \neg\Phi(y)]. \tag{96}$$

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

$$\begin{aligned}&\exists y\exists\Phi[(\neg\Phi(a) \wedge \forall x(\neg\Phi(x) \vee P(x)) \wedge P(y) \wedge \neg\Phi(y))\\ &\qquad \vee(\Phi(b) \wedge \forall x(\neg\Phi(x) \vee P(x)) \wedge P(y) \wedge \neg\Phi(y))].\end{aligned} \tag{97}$$

Applying Proposition 1(6) to (97), we obtain

$$\begin{aligned}&\exists y\exists\Phi[\neg\Phi(a) \wedge \forall x(\neg\Phi(x) \vee P(x)) \wedge P(y) \wedge \neg\Phi(y)]\\ &\vee\exists y\exists\Phi[\Phi(b) \wedge \forall x(\neg\Phi(x) \vee P(x)) \wedge P(y) \wedge \neg\Phi(y)].\end{aligned} \tag{98}$$

The preprocessing phase is successfully completed. We now process the disjuncts from (98) as separate problems. Let us begin with the first disjunct. Note that Φ can be removed from it directly, using Proposition 2. To this end, we replace the disjunct with its equivalent given by

$$\exists y\exists\Phi\forall x[\neg\Phi(a) \wedge (\neg\Phi(x) \vee P(x)) \wedge P(y) \wedge \neg\Phi(y)]. \tag{99}$$

Applying Proposition 2 to (99), we obtain

$$\exists y\forall x P(y) \tag{100}$$

which reduces to

$$\exists y P(y). \tag{101}$$

We have succeeded in reducing the first disjunct in (98) to a first-order formula. We now try to do the same for the second one, i.e.

$$\exists y\exists\Phi[\Phi(b) \wedge \forall x(\neg\Phi(x) \vee P(x)) \wedge P(y) \wedge \neg\Phi(y)]. \tag{102}$$

Preparation for Ackermann's Lemma. Using Proposition 1 (15), we replace (102) by

$$\exists y\exists\Phi\forall z[(\Phi(z) \vee z \neq b) \wedge \forall x(\neg\Phi(x) \vee P(x)) \wedge P(y) \wedge \neg\Phi(y)]. \tag{103}$$

Application of Ackermann's Lemma. Applying Ackermann's Lemma to (103), we obtain

$$\exists y\forall x[(x \neq b \vee P(x)) \wedge y \neq b \wedge P(y)]. \tag{104}$$

We have now succeeded in reducing the second disjunct of (98). The original second-order formula (97) has now been reduced to the equivalent first-order formula

$$[\exists y P(y)] \vee [\exists y\forall x[(x \neq b \vee P(x)) \wedge y \neq b \wedge P(y)]]. \tag{105}$$

Applying the simplification step to (105) using Proposition 1 (15), we obtain

$$[\exists y P(y)] \vee [\exists y[P(b) \wedge y \neq b \wedge P(y)]]. \tag{106}$$

Finally, we negate (106), obtaining

$$[\forall y \neg P(y)] \wedge \forall y[\neg P(b) \vee y = b \vee \neg P(y)]. \tag{107}$$

The first-order formula (107) is logically equivalent to the second-order formula (93). Consequently,

$$Circ(\Gamma; P; ()) \equiv \Gamma \wedge [\forall y \neg P(y)] \wedge \forall y[\neg P(b) \vee y = b \vee \neg P(y)]. \tag{108}$$

(107) can be simplified using some standard equivalences:

$$[\forall y \neg P(y)] \wedge \forall y[\neg P(b) \vee y = b \vee \neg P(y)] \equiv \forall y \neg P(y). \tag{109}$$

□

6.6. An Existential Example

Kolaitis and Papadimitriou [12] 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 [12] 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: [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

$$\exists x_1 \exists x_2[R(x_1, x_2) \wedge P(x_1) \wedge P(x_2)] \tag{110}$$

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

$$\begin{gathered}\exists x_1(R(x_1,x_1) \wedge P(x_1) \wedge (\forall y(P(y) \equiv y = x_1)) \\ \vee[\exists x_1 \exists x_2(R(x_1,x_2) \wedge P(x_1) \wedge P(x_2) \wedge (x_1 \neq x_2) \wedge \\ (\forall y(P(y) \equiv (y = x_1 \vee y = x_2))) \wedge \neg R(x_1,x_1) \wedge \neg R(x_2,x_2))].\end{gathered} \tag{111}$$

We apply our reduction algorithm to the same theory and compare the results.
Let $\Gamma(P)$ be the theory

$$\exists x_1 \exists x_2[R(x_1, x_2) \wedge P(x_1) \wedge P(x_2)]. \tag{112}$$

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

$$Circ(\Gamma(P); P; ()) \equiv \Gamma(P) \wedge \forall\Phi[\Gamma(\Phi) \wedge [\Phi \leq P] \supset [P \leq \Phi]], \tag{113}$$

where

$$\begin{aligned} \Gamma(\Phi) &\equiv \exists x_1 \exists x_2 [R(x_1, x_2) \wedge \Phi(x_1) \wedge \Phi(x_2)] && (114)\\ \Phi \leq P &\equiv \forall x(\Phi(x) \supset P(x)) && (115)\\ P \leq \Phi &\equiv \forall x(P(x) \supset \Phi(x)). && (116) \end{aligned}$$

In the following, we will reduce

$$\forall\Phi[\Gamma(\Phi) \wedge [\Phi \leq P] \supset [P \leq \Phi]] \tag{117}$$

in (113). Negating (117), we obtain

$$\exists\Phi[\Gamma(\Phi) \wedge [\Phi \leq P] \wedge \neg[P \leq \Phi]]. \tag{118}$$

Preprocessing. Replacing $\Gamma(\Phi)$, $\Phi \leq P$ and $P \leq \Phi$ by their equivalents given by (114)–(116), eliminating $\supset$ and renaming individual variables, we obtain

$$\begin{aligned} &\exists\Phi[\exists x_1 \exists x_2 (R(x_1, x_2) \wedge \Phi(x_1) \wedge \Phi(x_2)) \wedge \\ &\quad \forall y(\neg\Phi(y) \vee P(y)) \wedge \exists z(P(z) \wedge \neg\Phi(z))]. \end{aligned} \tag{119}$$

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

$$\begin{aligned} &\exists x_1 \exists x_2 \exists z \exists\Phi[R(x_1, x_2) \wedge \Phi(x_1) \wedge \Phi(x_2) \wedge \\ &\qquad \forall y(\neg\Phi(y) \vee P(y)) \wedge P(z) \wedge \neg\Phi(z)]. \end{aligned} \tag{120}$$

Preparation for Ackermann's Lemma. Applying Proposition 1 (15) and some standard equivalences, we replace (120) by

$$\begin{aligned} &\exists x_1 \exists x_2 \exists z \exists\Phi\forall q[(\Phi(q) \vee (q \neq x_1 \wedge q \neq x_2)) \wedge (R(x_1, x_2) \wedge \\ &\qquad\qquad \forall y(\neg\Phi(y) \vee P(y)) \wedge P(z) \wedge \neg\Phi(z)]. \end{aligned} \tag{121}$$

Application of Ackermann's Lemma. Ackermann's Lemma can now be applied to (121) resulting in

$$\begin{aligned} &\exists x_1 \exists x_2 \exists z[R(x_1, x_2) \wedge \forall y((y \neq x_1 \wedge y \neq x_2) \vee P(y)) \wedge \\ &\qquad\qquad P(z) \wedge z \neq x_1 \wedge z \neq x_2]. \end{aligned} \tag{122}$$

Simplification. Applying Proposition 1(18) to (122) results in

$$\exists x_1 \exists x_2 \exists z[R(x_1, x_2) \wedge P(x_1) \wedge P(x_2) \wedge P(z) \wedge z \neq x_1 \wedge z \neq x_2]. \tag{123}$$

Negating (123), we obtain

$$\forall x_1 \forall x_2 \forall z[\neg R(x_1, x_2) \vee \neg P(x_1) \vee \neg P(x_2) \vee \neg P(z) \vee z = x_1 \vee z = x_2]. \tag{124}$$

The first-order formula (124) is logically equivalent to the second-order formula (117). Consequently,

$$\begin{aligned} &Circ(\Gamma; P) \equiv \Gamma(P) \wedge \\ &\quad \forall x_1 \forall x_2 \forall z[\neg R(x_1, x_2) \vee \neg P(x_1) \vee \neg P(x_2) \vee \neg P(z) \vee z = x_1 \vee z = x_2]. \end{aligned} \quad (125)$$

Comparing (125) with (111), it is easily observed that there is a substantial difference in the size of the formulas. For example, the output of DLS contains a total of 62 symbols versus 92 for the Kolaitis and Papadimitriou approach. Without counting brackets and parentheses, the symbol count is 44 versus 57, respectively. Of course, this is only one example. Whether the comparison holds for a larger space of problems is questionable. □

6.7. 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. 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.
3. Execution of the algorithm is easily followed. For shorter examples the DLS algorithm can be applied with pencil and paper, although an implementation of DLS is obviously a better means of doing the reductions.

7. Related Work

Besides the work already mentioned in this article, there are a number of other references related to the reduction problem that are worth mentioning and also analyzing in future work. In addition to his work in the area of *correspondence theory* ([3],[4]), Johan van Benthem has also investigated the logic of circumscription and its first-order reduction [5]. Kartha and Lifschitz [11], have recently introduced an action theory using nested circumscription [16], for reasoning about action and change. The nested circumscription theories are reduced using the SCAN algorithm. Doherty, Łukaszewicz, and Szałas [6], have recently investigated the characterization of normal logic programs using a generalization of the algorithm described in this article. Rather than reducing second-order formulas to logically equivalent first-order formulas, one reduces to a fixpoint formula and looks for bounds on the fixpoint. If successful then there is a reduction from the fixpoint formula to a first-order formula. Doherty, Łukaszewicz, and Szałas [7], are also investigating first-order reductions of a general form of domain circumscription which also uses a generalization of the DLS algorithm.

8. 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 and simplification stages. In addition, we feel that the use of circumscription as a knowledge representation tool deserves reevaluation in light of not only the results described here, but recent results of Lifschitz and Kartha [11] from a related project in which reductions are based on the SCAN algorithm. The common view of circumscription as an elegant formalism for conceptual analysis, but one that is difficult to apply in practice due to its second-order nature, requires modification if these and other algorithms can be applied practically, as we believe they can.

Acknowledgments

We would like to thank Thomas Drakengren, Joakim Gustafsson, Lars Karlsson, G. Neelakantan Kartha, and Vladimir Lifschitz for useful discussions.

Appendix

The Algorithm in Detail

A.1. 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[7]. 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 four basic phases: (1) preprocessing; (2) preparation for Ackermann's Lemma; (3) application of Ackermann's Lemma; and (4) simplification. 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 the phase are skipped.

A.1.1. Preprocessing

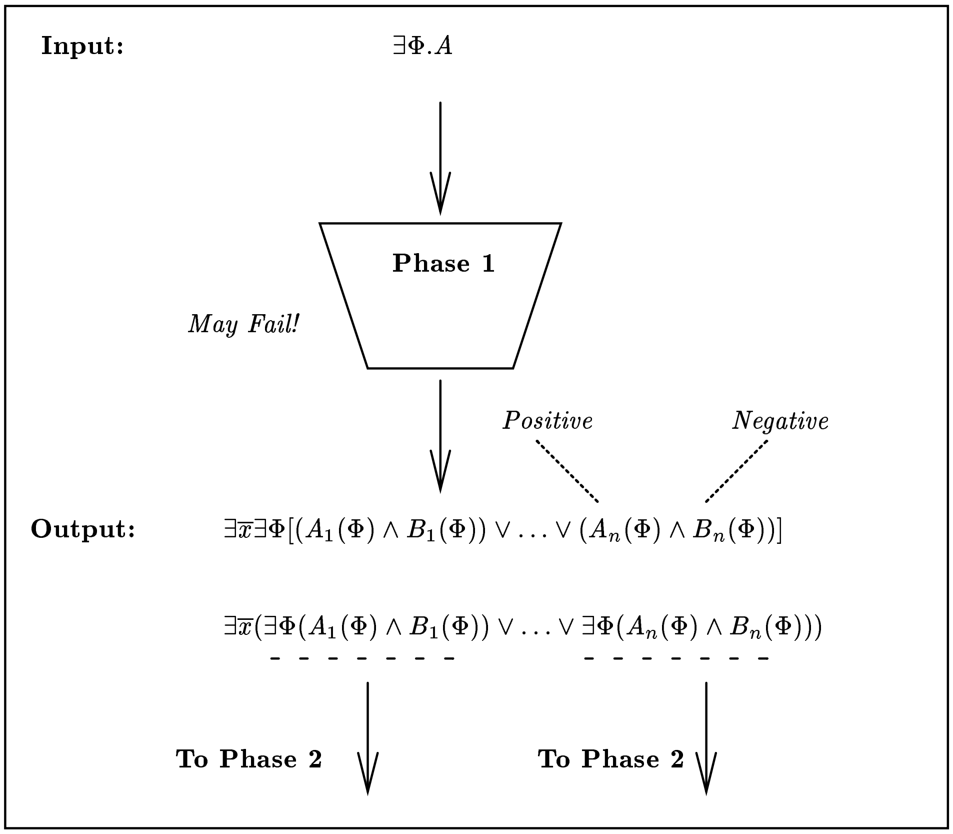

Input:
$\exists\Phi.A$
Phase 1
May Fail!
Positive
Negative
Output:
$\exists\overline{x}\exists\Phi[(A_1(\Phi)\wedge B_1(\Phi))\vee\ldots\vee(A_n(\Phi)\wedge B_n(\Phi))]$
$\exists\overline{x}(\exists\Phi(A_1(\Phi)\wedge B_1(\Phi))\vee\ldots\vee\exists\Phi(A_n(\Phi)\wedge B_n(\Phi)))$
To Phase 2
To Phase 2

Figure A.1. Phase 1: Preprocessing the Input.

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

$$\exists\bar{x}\exists\Phi[(A_1(\Phi)\wedge B_1(\Phi))\vee\cdots\vee(A_n(\Phi)\wedge B_n(\Phi))], \tag{A.1}$$

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

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 occurs both bound and free. Using the usual equivalences, move the negation connective to the right until all its occurrences immediately proceed atomic formulas.

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, containing both positive and negative occurrences of Φ, that occur among their conjuncts. For this purpose, apply the following equivalences:

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

   only if $B \vee C$ $(A \vee B)$ have both positive and negative occurrences of Φ

   If the resulting formula is not in the form (A.1), then report the failure of the algorithm. Otherwise replace (A.1) by its equivalent given by

   $$\exists\bar{x}(\exists\Phi(A_1(\Phi) \wedge B_1(\Phi)) \vee \cdots \vee \exists\Phi.(A_n(\Phi) \wedge B_n(\Phi))). \tag{A.2}$$

   For each disjunct $\exists\Phi(A_i(\Phi) \wedge B_i(\Phi))$ of (A.2) try to find its first-order equivalent by apply 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: 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.

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

$\square$

A.1.2. Preparation for Ackermann's Lemma

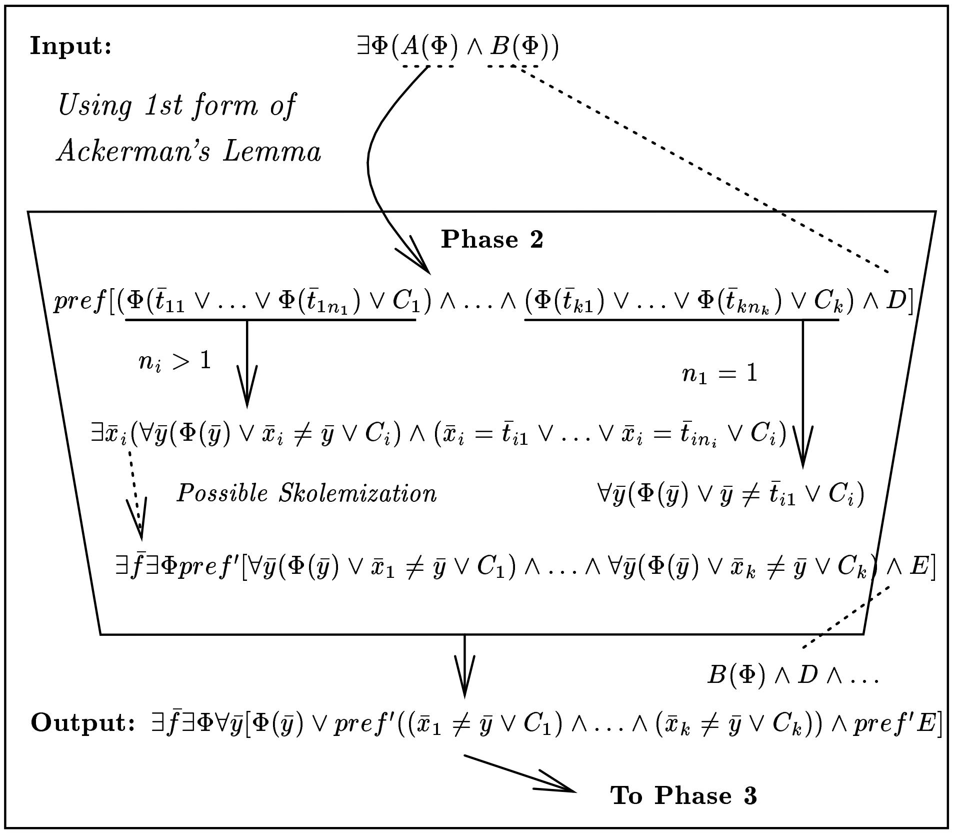

Input: $\exists\Phi(A(\Phi) \wedge B(\Phi))$
Using 1st form of Ackerman's Lemma
Phase 2
$pref[(\Phi(\bar{t}_{11} \vee \ldots \vee \Phi(\bar{t}_{1n_1}) \vee C_1) \wedge \ldots \wedge (\Phi(\bar{t}_{k1}) \vee \ldots \vee \Phi(\bar{t}_{kn_k}) \vee C_k) \wedge D]$
$n_i > 1$
$n_1 = 1$
$\exists\bar{x}_i(\forall\bar{y}(\Phi(\bar{y}) \vee \bar{x}_i \neq \bar{y} \vee C_i) \wedge (\bar{x}_i = \bar{t}_{i1} \vee \ldots \vee \bar{x}_i = \bar{t}_{in_i} \vee C_i)$
Possible Skolemization
$\forall\bar{y}(\Phi(\bar{y}) \vee \bar{y} \neq \bar{t}_{i1} \vee C_i)$
$\exists\bar{f}\exists\Phi pref'[\forall\bar{y}(\Phi(\bar{y}) \vee \bar{x}_1 \neq \bar{y} \vee C_1) \wedge \ldots \wedge \forall\bar{y}(\Phi(\bar{y}) \vee \bar{x}_k \neq \bar{y} \vee C_k) \wedge E]$
$B(\Phi) \wedge D \wedge \ldots$
Output: $\exists\bar{f}\exists\Phi\forall\bar{y}[\Phi(\bar{y}) \vee pref'((\bar{x}_1 \neq \bar{y} \vee C_1) \wedge \ldots \wedge (\bar{x}_k \neq \bar{y} \vee C_k)) \wedge pref'E]$
To Phase 3

Figure A.2. Phase 2: Preparation for Ackermann's 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 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 Ackermann's 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 in Step 1 of form $(\Phi(\bar{t}_{i1}) \vee \cdots \vee \Phi(\bar{t}_{in_i}) \vee C_i)$, 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 $pref$ in Step 1. This step is justified by equivalence (16) of Proposition 1. In addition, move each of the second conjuncts, $(\bar{x}_i = \bar{t}_{i1} \vee \cdots \vee \bar{x}_i = \bar{t}_{in_i} \vee C_i)$, into D in Step 1, renaming it D'.

3. Transform each conjunct in Step 1 of form $(\Phi(\bar{t}_{i1}) \vee C_i)$ 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

$$\forall \bar{x}.\exists y.A(\bar{x}, y, \ldots) \equiv \exists f.\forall \bar{x}.A(\bar{x}, y \leftarrow f(\bar{x}), \ldots), \tag{A.3}$$

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

$$\exists \bar{f} \exists \Phi pref'[\forall \bar{y}(\Phi(\bar{y}) \vee \bar{x}_1 \neq \bar{y} \vee C_1) \wedge \cdots \wedge \forall \bar{y}(\Phi(\bar{y}) \vee \bar{x}_k \neq \bar{y} \vee C_k) \wedge E], \tag{A.4}$$

where $\bar{f}$ is the tuple of the introduced Skolem functions, $pref'$ only contains universal quantifiers, and E is $D' \wedge B(\Phi)$.

5. Transform (A.4) into its equivalent given by

$$\exists \bar{f} \exists \Phi \forall \bar{y}[\Phi(\bar{y}) \vee pref'((\bar{x}_1 \neq \bar{y} \vee C_1) \wedge \cdots \wedge (\bar{x}_k \neq \bar{y} \vee C_k)) \wedge pref'E. \tag{A.5}$$

Example: [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.

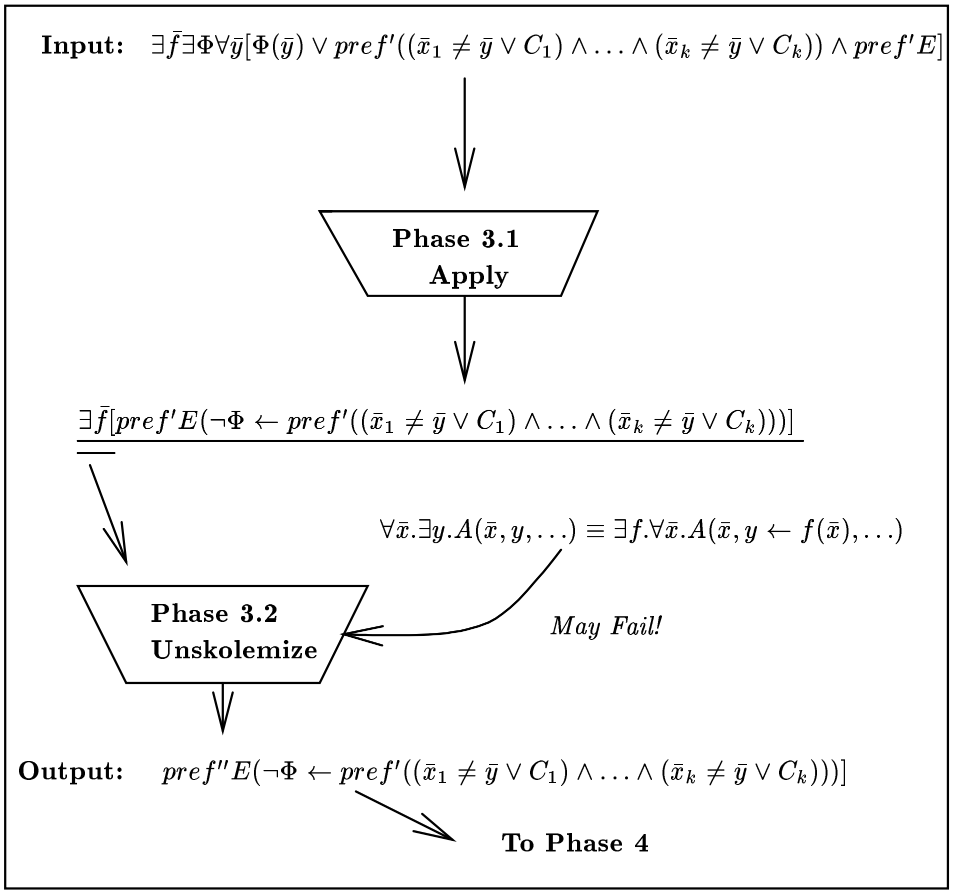
$$\begin{array}{ll} \exists \Phi(P(y) \wedge \Phi(z) \wedge \neg\Phi(u)) & \equiv \text{ (by 2)} \\ \exists \Phi(P(y) \wedge \forall r(\Phi(r) \vee z \neq r) \wedge \neg\Phi(u)) & \end{array}$$

Applying phase 2 to the second formula proceeds as follows.

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

□

A.1.3. Application of Ackermann's Lemma



Input: $\exists\bar{f}\exists\Phi\forall\bar{y}[\Phi(\bar{y}) \vee pref'((\bar{x}_1 \neq \bar{y} \vee C_1) \wedge \ldots \wedge (\bar{x}_k \neq \bar{y} \vee C_k)) \wedge pref'E]$
Phase 3.1
Apply
$\exists\bar{f}[pref'E(\neg\Phi \leftarrow pref'((\bar{x}_1 \neq \bar{y} \vee C_1) \wedge \ldots \wedge (\bar{x}_k \neq \bar{y} \vee C_k)))]$
$\forall\bar{x}.\exists y.A(\bar{x}, y, \ldots) \equiv \exists f.\forall\bar{x}.A(\bar{x}, y \leftarrow f(\bar{x}), \ldots)$
May Fail!
Phase 3.2
Unskolemize
Output: $pref''E(\neg\Phi \leftarrow pref'((\bar{x}_1 \neq \bar{y} \vee C_1) \wedge \ldots \wedge (\bar{x}_k \neq \bar{y} \vee C_k)))]$
To Phase 4

Figure A.3. Phase 3: Application of Ackermann's Lemma

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

1. Apply Ackermann's Lemma to the formula (A.5). 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 (A.3). If this is impossible, the algorithm fails for the first form of Ackermann's Lemma. Using the second form returned from A.1.2, try to remove the existential quantifiers over function variables. If this is successful, go to the next step. If not, the algorithm fails.

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

$$\begin{array}{ll} \exists\Phi(P(y) \wedge \forall r(\Phi(r) \vee z \neq r) \wedge \neg\Phi(u)) & \equiv \text{ (by 1)} \\ P(y) \wedge z \neq u. & \end{array}$$

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

□

A.1.4. Simplification

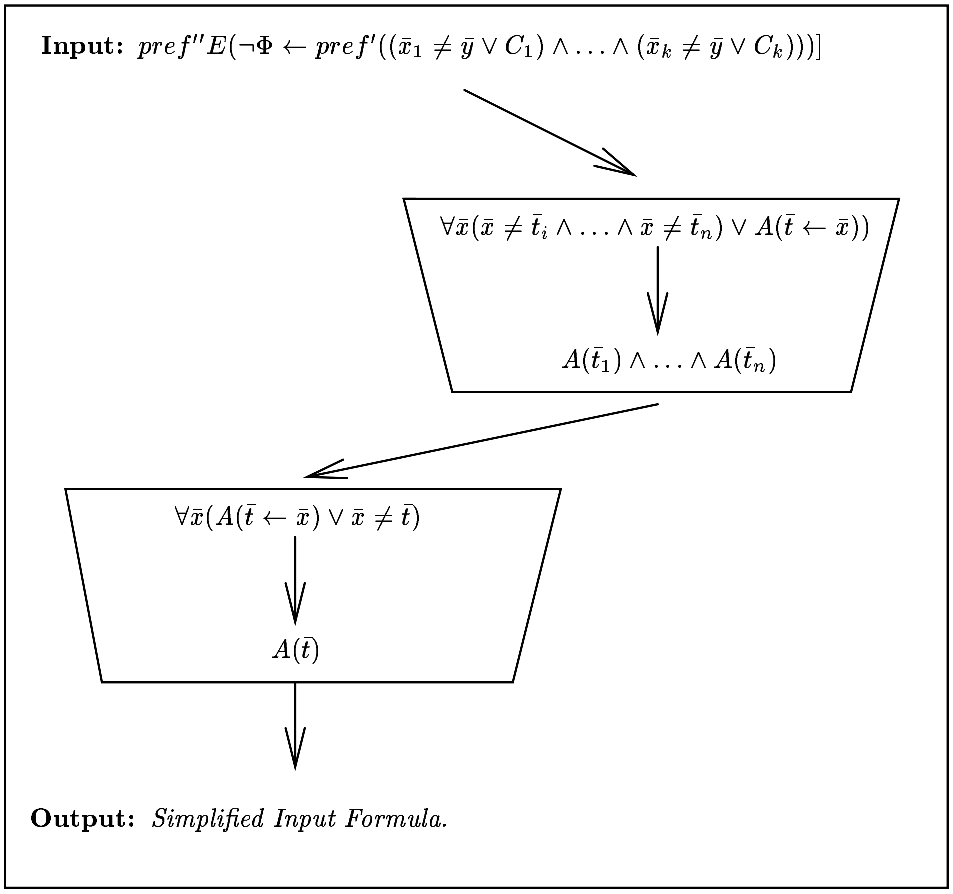

Input: $pref''E(\neg\Phi \leftarrow pref'((\bar{x}_1 \neq \bar{y} \vee C_1) \wedge \ldots \wedge (\bar{x}_k \neq \bar{y} \vee C_k)))]$
$\forall\bar{x}(\bar{x} \neq \bar{t}_i \wedge \ldots \wedge \bar{x} \neq \bar{t}_n) \vee A(\bar{t} \leftarrow \bar{x}))$
$A(\bar{t}_1) \wedge \ldots \wedge A(\bar{t}_n)$
$\forall\bar{x}(A(\bar{t} \leftarrow \bar{x}) \vee \bar{x} \neq \bar{t})$
$A(\bar{t})$
Output: *Simplified Input Formula.*

Figure A.4. Phase 4: Simplification

The formula obtained as the result of the previous phase can often be substantially simplified using Proposition 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: [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

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

□

A.1.5. 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.

Notes

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, [4]).
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).
4. Rabinov requires n-simplicity here.
5. The second form considered in Lemma 1 is symmetric to the first one.
6. If the axiom were added to the theory, the DLS algorithm would fail due to unskolemization problems.
7. 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, [4]).
8. To increase the strength of the algorithm, it is essential to move as many existentially quantified variables as possible into the prefix of (A.1).

References

1. W. Ackermann. Untersuchungen über das Eliminationsproblem der Mathematischen logik. *Mathematische Annalen*, 110:390–413, 1935.
2. W. Ackermann. *Solvable Cases of the Decision Problem.* North-Holland, Amsterdam, 1954.
3. J Van Benthem. *Modal Logic and Classical Logic.* Bibliopolis, Napoli, 1983.
4. J. Van Benthem. Correspondence theory. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 2, pages 167–247. D. Reidel Publishing Co., 1984.
5. J. Van Benthem. Semantic parallels in natural language and computation. In H-D Ebbinghaus et al., editors, *Logic Colloquium, Granada 1987*, pages 331–375, 1989.
6. P. Doherty, W Łukaszewicz, and A. Szałas. A characterization result for circumscribed normal logic programs. Technical Report LITH-IDA-95-20, Department of Computer and Information Science, Linköping University, Linköping, Sweden, 1995.
7. P. Doherty, W Łukaszewicz, and A. Szałas. General Domain Circumscription and its First-Order Reduction. Technical Report LITH-IDA-95, Department of Computer and Information Science, Linköping University, Linköping, Sweden, 1995.
8. D. Gabbay and H. J. Ohlbach. Quantifier elimination in second-order predicate logic. Technical Report MPI-I-92-231, Max-Planck Institut für Informatik, Saarbrücken, Germany, 1992.
9. M. Gelfond and V. Lifschitz. Compiling circumscriptive theories into logic programs. In *Proc. 2nd Int'l Workshop on Non-Monotonic Reasoning*, volume 346 of *Lecture Notes in Artificial Intelligence*, pages 74–99, Berlin, 1989. Springer-Verlag.
10. M. L. Ginsberg. A circumscriptive theorem prover. *Artificial Intelligence*, 39:209–230, 1989.
11. G. N. Kartha and V. Lifschitz. A simple formalization of actions using circumscription. In *Proceedings of the 14th Int'l Joint Conference on Artificial Intelligence*, 1995.
12. P. Kolaitis and C. Papadimitriou. Some computational aspects of circumscription. In *AAAI-88: Proceedings of the 7th National Conference on Artificial Intelligence*, pages 465–469, 1988.
13. V. Lifschitz. Computing circumscription. In *Proceedings of the 9th Int'l Joint Conference on Artificial Intelligence*, volume 1, pages 121–127, 1985.
14. V. Lifschitz. Pointwise circumscription. In M. Ginsberg, editor, *Readings in Nonmonotonic Reasoning*, pages 179–193. Morgan Kaufmann, 1988.
15. V. Lifschitz. Circumscription. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Nonmonotonic Reasoning and Uncertain Reasoning*, volume 3 of *Handbook of Artificial Intelligence and Logic Programming.* Oxford University Press, 1994.
16. V. Lifschitz. Nested abnormality theories. *Artificial Intelligence*, 1995. To Appear.
17. L. Löwenheim. Über Möglichkeiten im Relativekalkül. *Mathematische Annalen*, pages 137–148, 1915.
18. J. McCarthy. Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1-2):27–39, 1980.
19. T. Przymusinski. An algorithm to compute circumscription. *Artificial Intelligence*, 38:49–73, 1991.
20. A. Rabinov. A generalization of collapsible cases of circumscription (research note). *Artificial Intelligence*, 38:111–117, 1989.
21. R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.
22. A. Szałas. On the correspondence between modal and classical logic: an automated approach. *Journal of Logic and Computation*, 3:605–620, 1993.