A Characterization Result for Circumscribed Normal Logic Programs

Patrick Doherty and Witold Łukaszewicz and Andrzej Szałas

Abstract

Circumscription has been perceived as an elegant mathematical technique for modeling nonmonotonic and commonsense reasoning, but difficult to apply in practice due to the use of second-order formulas. One proposal for dealing with the computational problems is to identify classes of first-order formulas whose circumscription can be shown to be equivalent to a first-order formula. In previous work, we presented an algorithm which reduces certain classes of second-order circumscription axioms to logically equivalent first-order formulas. The basis for the algorithm is an elimination lemma due to Ackermann. In this paper, we capitalize on the use of a generalization of the Ackermann lemma in order to deal with a subclass of universal formulas used in normal logic programs. Our results subsume previous results by Kolaitis and Papadimitriou regarding a characterization of circumscribed definite logic programs which are first-order expressible. The means for distinguishing which programs are reducible is based on a boundedness criterion. The approach we use is to first reduce circumscribed normal logic programs to a fixpoint formula which is reducible if the program is bounded, otherwise not. In addition to a number of other extensions, we also present a fixpoint calculus which is shown to be sound and complete for bounded fixpoint formulas.

The first author has been supported in part by the Swedish Council for the Engineering Sciences (TFR). The second and third authors are affiliated with the Institute of Informatics, Warsaw University, and have been supported in part by KBN grant 3 P406 019 06.

IDA Technical Report 1995
LiTH-IDA-R-95-20
ISSN-0281-4250

Department of Computer
and Information Science,
Linköping University,
S-581 83 Linkping, Sweden

A Characterization Result for Circumscribed Normal Logic Programs

Patrick Doherty and Witold Łukaszewicz and Andrzej Szałas

Abstract

Circumscription has been perceived as an elegant mathematical technique for modeling nonmonotonic and commonsense reasoning, but difficult to apply in practice due to the use of second-order formulas. One proposal for dealing with the computational problems is to identify classes of first-order formulas whose circumscription can be shown to be equivalent to a first-order formula. In previous work, we presented an algorithm which reduces certain classes of second-order circumscription axioms to logically equivalent first-order formulas. The basis for the algorithm is an elimination lemma due to Ackermann. In this paper, we capitalize on the use of a generalization of the Ackermann lemma in order to deal with a subclass of universal formulas used in normal logic programs. Our results subsume previous results by Kolaitis and Papadimitriou regarding a characterization of circumscribed definite logic programs which are first-order expressible. The means for distinguishing which programs are reducible is based on a boundedness criterion. The approach we use is to first reduce circumscribed normal logic programs to a fixpoint formula which is reducible if the program is bounded, otherwise not. In addition to a number of other extensions, we also present a fixpoint calculus which is shown to be sound and complete for bounded fixpoint formulas.

1 Introduction

In recent years, a great deal of attention has been devoted to logics of "commonsense" reasoning. Circumscription [8], one of the more influential formalisms, has been perceived as an elegant mathematical technique for modeling nonmonotonic reasoning, but difficult to apply in practice. The circumscription technique transforms logical formulas by adding an additional requirement of minimality for selected predicates in a theory. The minimality requirement is enforced by adding a second-order formula to the theory. The addition of a second-order formula to a set of first-order formulas generally rules out the use of complete deductive systems for reasoning about the commonsense theories.

There have been a number of proposals for dealing with the computational problems, ranging from isolating classes of first-order formulas whose circumscription can be compiled into logic programs [3], to developing more direct specialized inference methods ([4, 10]) for similarly restricted classes of formulas. Yet another approach, and the one we will consider here, is to identify classes of first-order formulas whose circumscription can be shown to be equivalent to a first order formula. The obvious advantage of

this approach is that classical theorem-proving techniques can be used to reason about commonsense theories.

In previous work, Doherty et. al. [2], described 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 elimination algorithm (which will be referred to as the "DLS 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, only that the algorithm can not find one. The DLS algorithm is based on a lemma proved by Ackermann [1] in 1934. In addition, we have shown that the DLS algorithm essentially subsumes all previous results for reducing circumscription axioms [2]. Although a step in the right direction, the DLS algorithm is essentially limited[1] to those input formulas that are *separable* (see [8]). An obvious line of research is to investigate whether and in what ways the current algorithm can be extended to deal with classes of formulas which are non-separable.

The class of non-separable formulas has previously been investigated by a number of researchers. Lifschitz [7], who characterized the class of separable formulas, observed that there are non-separable formulas that can be reduced to equivalent first-order formulas and those that can not. In fact, Lifschitz presented an example of a non-separable circumscribed theory that is not first-order expressible, where only universal formulas consisting of a conjunction of function-free Horn clauses are used. Kolaitis and Papadimitriou [5] continued the investigation and were interested in finding a computationally useful characterization of universal formulas (of which the function-free Horn clauses are a subclass) that have a first-order circumscription. They approached the problem by establishing a connection between the circumscription of a conjunction of function-free Horn clauses and the convergence of the corresponding logic program. They showed that the circumscription of a conjunction of Horn clauses is first-order if and only if the corresponding program is bounded.

In this paper, we continue the line of research pursued by Kolaitis and Papadimitriou in the following manner. We first provide a re-characterization of the Kolaitis and Papadimitriou result in terms of the more general setting of the DLS algorithm described in Doherty et. al. [2]. Since we will focus on classes of formulas that are not separable, the DLS algorithm is inadequate. Fortunately, a generalization of this algorithm, described in Nonengart and Szalas [9], can be used for this purpose. Nonengart and Szalas consider the problem of finding automated techniques for modal logic correspondence theory. The idea is to find algorithms for the automatic synthesis of correspondence axioms expressed in classical logic.

Given a modal logic schema as input, in the best case, such algorithms return the corresponding first-order formula. Unfortunately, it is well known that certain modal schemas, such as Löb's schema, are not first-order definable. In cases where modal schemas can not be reduced to first-order formulas, the Ackermann technique is inadequate and sometimes results in infinite formulas. One way to deal with a larger class of modal schemas is to generalize the Ackermann technique by allowing fixpoint opera-

[1]In fact, the algorithm can deal with certain classes of non-separable formulas given additional heuristics.

tors within formulas output by the algorithm. Since unbounded logic programs behave in a similar manner, a similar approach can also be used to reduce circumscribed logic programs to a classical language extended with fixpoint operators. In the case where the circumscribed logic program given as input is bounded and can be put in the proper input form, the technique returns as output a fixpoint formula which can then be shown to be first-order definable. In the case where its input is not bounded, the technique returns a fixpoint formula.

In addition to providing the re-characterization, the original result shown by Kolaitis and Papadimitriou is extended in several respects.

- Our result applies to normal logic programs rather than definite logic programs, which are a subclass of normal logic programs.
- An explicit definition of the minimized predicate is returned as part of the output. This is the only place where fixpoint operators appear.
- Based on the use of a theorem concerning *separated* formulas (due to Lifschitz [8]), our result can be extended to normal logic programs with arbitrary "purely negative" parts for the minimized predicate.

Finally, we provide a basis for theorem proving for the bounded subclass of circumscribed normal logic programs by presenting a proof system for the fixpoint calculus which is shown to be both sound and complete for bounded fixpoint formulas.

The paper is organized as follows. In Section 3, a brief description of circumscription is provided together with the Ackermann Lemma which is used as a basis for the DLS algorithm and its generalization. In Section 4, a brief introduction to the fixpoint calculus is provided together with a central theorem which generalizes the Ackermann technique to the fixpoint operator extension. In Section 5, which presents the main results, the earlier Kolaitis and Papadimitriou result is described together with its generalization and re-characterization in terms of the fixpoint calculus. The main subsumption theorem is then presented together with a further generalization for arbitrary "purely negative parts" of the minimized predicate in a logic program. In Section 6, a proof theory for the fixpoint calculus is presented that is shown to be sound and complete for bounded fixpoint formulas. We then conclude with a discussion.

2 Preliminaries

2.1 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 $\neg(\bar{x} = \bar{t})$.

2.2 Definitions

We will be primarily concerned with the Horn clause subclass of universal formulas. The following definitions will be used for defining logic programs.

Definition 2.1 A first-order sentence T is said to be *existential* (*universal*) iff it is of the form $\exists \overline{x}.\ T_1$ (resp. $\forall \overline{x}.\ T_1$), where T_1 is quantifier free. ■

Definition 2.2 *(Horn Clause)* A *Horn clause (w.r.t. predicate P)* is an expression of the form:

$$\forall \bar{x} \forall \bar{z}(A(\bar{x}, \bar{z}, P) \supset P(\bar{x})),$$

where $A(\bar{x}, \bar{z}, P)$ is a conjunction of positive atomic formulas, possibly including P. ■

Definition 2.3 *(Logic Program)* A *definite logic program (w.r.t. P)* is a conjunction of Horn clauses (w.r.t. P) of the form:

$$\bigwedge_{i=1}^{k} \forall \bar{x} \forall \overline{z_i}\ (A_i(\bar{x}, \overline{z_i}, P) \supset P(\bar{x})). \tag{1}$$

A *normal logic program* is a formula of the form (1) except that atoms other than P may be negative in $A_i(\bar{x}, \bar{z}, P)$, for $i \leq k$. ■

It is easily observed that any logic program is equivalent to a formula of the form:

$$\forall \bar{x}[(\bigvee_{i=1}^{k} \exists \overline{z_i}\ A_i(\bar{x}, \overline{z_i}, P)) \supset P(\bar{x})]. \tag{2}$$

3 Circumscription

Since we primarily focus on the Horn clause subclass of universal formulas, we restrict ourselves to the following non-variable form of second-order circumscription.

Definition 3.1 (Second-Order Circumscription) Let $\overline{P}$ be a tuple of distinct predicate constants and let $T(\overline{P})$ be a sentence. The *second-order circumscription* of $\overline{P}$ in $T(\overline{P})$ written $Circ_{SO}(T; \overline{P})$, is the sentence

$$T(\overline{P}) \wedge \forall \overline{\Phi} \neg\ [T(\overline{\Phi}) \wedge \overline{\Phi} < \overline{P}] \tag{3}$$

where $\overline{\Phi}$ is a tuple of variables similar to $\overline{P}$. ■

Observe that (3) can be rewritten as

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

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.

$$\forall \overline{\Phi}\,[T(\overline{\Phi}) \wedge [\overline{\Phi} \leq \overline{P}]] \supset [\overline{P} \leq \overline{\Phi}]]. \tag{4}$$

Definition 3.2 A predicate variable Φ occurs *positively* (resp. *negatively*) in a formula A if the conjunctive normal form of A contains a subformula of the form $\Phi(\bar{t})$ (resp. $\neg\Phi(\bar{t})$). A formula A is said to be *positive* (resp. *negative*) w.r.t. Φ iff all occurrences of Φ in A are positive (resp. negative). ■

Definition 3.3 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

$$\bigvee_{i=1}^{m} [B_i(\overline{\phi}) \wedge (\overline{U}_i \leq \overline{\phi})] \tag{5}$$

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 $\overline{\phi}$.■

3.1 DLS Algorithm

In this section, we briefly describe the DLS algorithm mentioned in the introduction. Its complete formulation can be found in [2]. The algorithm was originally formulated in a weaker form in [12], in the context of modal logics. It is based on Ackermann's techniques developed in connection with the elimination problem (see [1]).The DLS algorithm is based on the following lemma, proved by Ackermann in 1934 (see [1]). The proof can also be found in [12].

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

$$\exists\Phi\forall\bar{x}[\Phi(\bar{x}) \vee A(\bar{x},\bar{z})] \wedge B(\Phi \leftarrow \neg\Phi) \;\equiv\; B(\Phi \leftarrow A(\bar{x},\bar{z})) \tag{6}$$

$$\exists\Phi\forall\bar{x}[\neg\Phi(\bar{x}) \vee A(\bar{x},\bar{z})] \wedge B(\Phi) \;\equiv\; B(\Phi \leftarrow A(\bar{x},\bar{z})) \tag{7}$$

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

The DLS algorithm is based on eliminating second-order quantifiers of the input formula using a combination of applications of Lemma 3.1 together with various syntactic transformations which preserve equivalence. One of the main restrictions of the DLS algorithm is that it does not allow clauses which contain both positive and negative occurrences of the predicate symbol which is being eliminated. In other words, formulas input to the DLS algorithm must essentially be separable. In Section 6, the use of Lemma 3.1 will be generalized to deal with input consisting of non-separable formulas, among others.

4 Fixpoint calculus

Let $\mathcal{L}_\mathcal{I}$ be the classical first-order logic. In order to define the *fixpoint calculus* $\mathcal{L}_\mathcal{F}$, we extend $\mathcal{L}_\mathcal{I}$ by allowing the least fixpoint operator $\mu\Phi.A(\Phi)$, where A is positive w.r.t. Φ. We abbreviate a formula of the form $\neg\mu\neg\Phi.\neg A(\neg\Phi)$ by $\nu\Phi.A(\Phi)$. It is sometimes convenient to indicate the individual variables that are bound by the fixpoint operators μ and ν. We write $\mu.\Phi(\bar{x})$ and $\nu.\Phi(\bar{x})$ to indicate that the tuple $\bar{x}$ of variables is bound by a fixpoint operator.

Every formula $A(\Phi)$ which is positive w.r.t. Φ is monotone. Consequently, by the Knaster & Tarski fixpoint theorem, we are assured that the fixpoints we consider are well defined. Moreover, the fixpoints have the following nice characterization:

$$\mu\Phi.A(\Phi) \;\equiv\; \bigvee_{\beta\in\alpha} A^\beta(\bot) \tag{8}$$

for an ordinal α.

Definition 4.1 The least ordinal α for which equivalence (8) holds is called the *closure ordinal for* $A(\Phi)$.

A fixpoint formula is called *bounded* iff it contains only fixpoint operators with finite closure ordinals. ■

Note that $\mu\Phi(\bar{x}).A(\Phi)$ is the least (w.r.t. $\leq$) formula $B(\bar{x})$ such that

$$B(\bar{x}) \;\equiv\; A(\Phi \leftarrow B(\bar{x})).$$

The following theorem is proved in [9]. It is used as a basis for our characterization result in Section 5.

Theorem 4.1 Assume that all occurrences of the predicate variable Φ in the formula B bind only variables.

- if A and B are negative w.r.t. Φ then the closure ordinal for $A(\neg\Phi)$ is less than or equal to ω, and

$$\exists\Phi\forall\bar{y}\;[\Phi(\bar{y})\vee A(\neg\Phi)]\wedge[B(\neg\Phi)] \;\equiv\; B[\neg\Phi \leftarrow \nu\neg\Phi(\bar{y}).A(\neg\Phi)] \tag{9}$$

- if A and B are positive w.r.t. Φ then the closure ordinal for $A(\Phi)$ is less than or equal to ω, and

$$\exists\Phi\forall\bar{y}[\neg\Phi(\bar{y})\vee A(\Phi)]\wedge[B(\Phi)] \;\equiv\; B[\Phi \leftarrow \nu\Phi(\bar{y}).A(\Phi)] \tag{10}$$

where the above substitutions exchange the variables bound by fixpoint operators by the corresponding actual variables of the substituted predicate.■

Note both the similarity with Lemma 3.1 and that the restriction of separability can be relaxed due to Theorem 4.1.

5 A characterization result

In this section, we first present the original result of Kolaitis and Papadimitriou who characterize the class of circumscribed function-free definite logic programs which are first-order expressible. We then provide a re-characterization of circumscribed normal logic programs in terms of the extended fixpoint language. Since definite logic programs are a subclass of normal logic programs, the new characterization applies to the class of logic programs considered by Kolaitis and Papadimitriou. We then present a theorem which characterizes the class of circumscribed normal logic programs which are first-order expressible and which subsumes the Kolaitis and Papadimitriou result.

5.1 Kolaitis and Papadimitriou result

Kolaitis and Papadimitriou [5] consider the circumscription of function-free definite logic programs and prove the following result:

Theorem 5.1 Let P be a predicate and $\Pi(P)$ be a definite logic program of the form,

$$\forall \bar{x}[(\bigvee_{i=1}^{k} \exists \overline{z_i}\ A_i(\bar{x}, \overline{z_i}, P)) \supset P(\bar{x})]. \tag{11}$$

Then $Circ_{SO}(\Pi(P), P)$ is equivalent to a first-order formula iff
$\mu P(\bar{x}).[\bigvee_{i=1}^{k} \exists \overline{z_i}\ A_i(\bar{x}, \overline{z_i}, P)]$ is bounded. ■

5.2 A characterization result

Before presenting the main result, we first provide a characterization of circumscribed normal logic programs in terms of the extended fixpoint language.

Lemma 5.1 Let P be a predicate and $\Pi(P)$ be a normal logic program of the form

$$\forall \bar{x}[(\bigvee_{i=1}^{k} \exists \overline{z_i}\ A_i(\bar{x}, \overline{z_i}, P)) \supset P(\bar{x})]. \tag{12}$$

Then $Circ_{SO}(\Pi(P), P)\ \equiv$

$$[\Pi(P) \wedge \forall \bar{x}(P(\bar{x}) \equiv \mu\Phi(\bar{x}). \bigvee_{i=1}^{k} \exists \overline{z_i} A_i(\bar{x}, \overline{z_i}, \Phi))] \tag{13}$$

Proof
Consider the second order part of $Circ_{SO}(\Pi(P), P)$. According to (4), it is equivalent to

$$\forall \Phi\ [\Pi(\Phi) \wedge [\Phi \leq P]] \supset [P \leq \Phi]]. \tag{14}$$

In order to apply Theorem 4.1, we first have to negate this formula, obtaining

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

which is equivalent to

$$\exists \Phi\ \Pi(\Phi) \wedge \forall \bar{x}[\neg \Phi(\bar{x}) \vee P(\bar{x})] \wedge \exists \bar{x}[P(\bar{x}) \wedge \neg \Phi(\bar{x})]. \tag{15}$$

Replacing $\Pi(\Phi)$ by its definition (12) in (15) results in the equivalent formula

$$\begin{aligned} &\exists \Phi\ \forall \bar{x}[\neg (\bigvee_{i=1}^{k} \exists \overline{z_i}\ A_i(\bar{x}, \overline{z_i}, \Phi)) \vee \Phi(\bar{x})] \wedge \\ &\quad \forall \bar{x}[\neg \Phi(\bar{x}) \vee P(\bar{x})] \wedge \exists \bar{x}[P(\bar{x}) \wedge \neg \Phi(\bar{x})]. \end{aligned} \tag{16}$$

The application of Theorem 4.1 to (16) results in the formula

$$\begin{aligned} &\forall \bar{x}[\nu \neg \Phi(\bar{x}).[\neg \bigvee_{i=1}^{k} \exists \overline{z_i}\ A_i(\bar{x}, \overline{z_i}, \Phi)] \vee \Phi(\bar{x})] \wedge \\ &\quad \exists \bar{x}[P(\bar{x}) \wedge \nu \neg \Phi(\bar{x}).[\neg \textstyle\bigvee_{i=1}^{k} \exists \overline{z_i}\ A_i(\bar{x}, \overline{z_i}, \Phi)]]. \end{aligned} \tag{17}$$

In order to re-obtain a formula equivalent to (14), (17) has to be negated. This results in

$$\begin{aligned} &\forall \bar{x}[(\mu \Phi(\bar{x}). \bigvee_{i=1}^{k} \exists \overline{z_i}\ A_i(\bar{x}, \overline{z_i}, \Phi)) \supset P(\bar{x})] \ \supset \\ &\quad \forall \bar{x}[P(\bar{x}) \supset \mu \Phi(\bar{x}). \textstyle\bigvee_{i=1}^{k} \exists \overline{z_i}\ A_i(\bar{x}, \overline{z_i}, \Phi)]. \end{aligned} \tag{18}$$

Now by application of the Park rule (23) to formula (12) we can infer that $\Pi(P)$ implies

$$\forall \bar{x}[(\mu P(\bar{x}). \bigvee_{i=1}^{k} \exists \overline{z_i}\ A_i(\bar{x}, \overline{z_i}, P)) \supset P(\bar{x})],$$

i.e. $\Pi(P)$ together with (18) is equivalent to (13). ■

Observe that Lemma 5.1 provides an explicit definition of the minimized predicate P in the circumscribed logic program. In addition, the only place that a fixpoint operator appears is in the explicit definition for the minimized predicate.

The following theorem subsumes Theorem 5.1 of Kolaitis and Papadimitriou and is substantially stronger.

Theorem 5.2 Let P be a predicate and $\Pi(P)$ be a normal logic program of the form given in Lemma 5.1. Then $Circ_{SO}(\Pi(P), P)$ is equivalent to a first-order formula iff the fixpoint formula

$$\mu \Phi(\bar{x}).[\bigvee_{i=1}^{k} \exists \overline{z_i}\ A_i(\bar{x}, \overline{z_i}, \Phi)] \tag{19}$$

is bounded.

Proof
If formula (19) is bounded then it is equivalent to a finite disjunction, i.e. to a first-order formula. Thus by equivalence (13) of Lemma 5.1, $Circ_{SO}(\Pi(P), P)$ is equivalent to a first-order formula.

Assume that formula (19) is not bounded and suppose that it is equivalent to a first-order formula, say A. Thus

$$\bigvee_{m\in\omega} [\bigvee_{i=1}^{k} \exists\overline{z_i}\ A_i(\bar{x}, \overline{z_i}, \bot)]^m \ \equiv\ A. \tag{20}$$

This implies that

$$\{\bigvee_{i=1}^{k} \exists\overline{z_i}\ A_i(\bar{x}, \overline{z_i}, \bot)]^m :\ m \in \omega\} \ \models\ A,$$

where $\models$ is the semantic consequence relation of the classical first-order logic. By compactness of $\models$ we conclude that

there is a finite $S \subseteq \{\bigvee_{i=1}^{k} \exists\overline{z_i}\ A_i(\bar{x}, \overline{z_i}, \bot)]^m :\ m \in \omega\}$ such that $S \models A$.

Obviously, A implies the conjunction of all formulas of S thus A is equivalent to the conjunction of all formulas of S, i.e. formula (19) is bounded. ■

Observe that due to the dual form (10) of Theorem 4.1, P can be replaced by $\neg P$, in Theorem 5.2.

5.3 Extending the characterization result

In this section, we extend the characterization result in Theorem 5.2 to normal logic programs with arbitrary "purely negative parts". The extension is based on a result by Lifschitz in [8] where he proposed an optimization of circumscription formulas, based on the following proposition (Proposition 3.3.1 of [8]).

Proposition 5.1 If $B(P)$ is negative w.r.t. P, then the circumscription $Circ_{SO}(A(P)\wedge B(P), P)$ is equivalent to $Circ_{SO}(A(P), P)\wedge B(P)$. ■

This proposition allows us to move the "purely negative part" of a formula being circumscribed outside the circumscription if the negative part can be separated from the positive part of the formula being circumscribed. Consequently our main theorem can be generalized to formulas with arbitrary "purely negative parts". This is shown in the following theorem.

Theorem 5.3 Theorem 5.2 holds for $\Pi(P)$ of the form $\Pi_1(P)\wedge\Pi_2(P)$, where Π_1 is a normal logic program of the form

$$\forall\bar{x}[(\bigvee_{i=1}^{k} \exists\overline{z_i}\ A_i(\bar{x}, \overline{z_i}, P)) \supset P(\bar{x})], \tag{21}$$

and $\Pi_2(P)$ is negative w.r.t. P. ■

Observe that in the above theorem, $\Pi_2(P)$ is arbitrary and does not have to be a logic program.

6 A proof system for bounded fixpoint calculus

The following section is somewhat independent from the previous sections with the exception of the use of the Park rule in the proof of Theorem 5.2. We describe a sound and complete proof system for the bounded fixpoint calculus which should be viewed as providing a possible basis for theorem proving in the extended logical language used in the paper.

Consider any sound and complete proof system $\mathcal{S}$ for the classical first-order logic. By $\mathcal{S}_\mu$ we shall mean $\mathcal{S}$ augmented with the following axiom (22) and inference rule (23):

$$\vdash_{\mathcal{S}_\mu} \ \mu\Phi.A(\Phi) \ \equiv \ A(\mu\Phi.A(\Phi)) \tag{22}$$

$$A(I) \supset I \ \vdash_{\mathcal{S}_\mu} \ \mu\Phi.A(\Phi) \ \supset \ I \qquad \text{(Park rule)} \tag{23}$$

This proof system is an obvious generalization of known proof systems for the propositional μ-calculus (see, e.g., [6]). It seems weak. On the other hand, as we show below, it is sound and complete for bounded fixpoint formulas. Some more general (however not effective) proof systems, sound and complete for any fixpoint formulas, are to be found e.g. in [11, 13].

Theorem 6.1 The Proof system $\mathcal{S}_\mu$ is sound and complete for the fixpoint calculus with bounded fixpoint formulas.

Proof
Soundness of $\mathcal{S}_\mu$ can be proved by applying standard techniques.

In order to show completeness of $\mathcal{S}_\mu$ it suffices to prove that for any classical first-order formulas I, $A(\Phi)$, if fixpoint formula $\mu\Phi.A(\Phi)$ is bounded then

$$\models \mu\Phi.A(\Phi) \ \equiv \ I \text{ implies } \vdash_{\mathcal{S}_\mu} \mu\Phi.A(\Phi) \ \equiv \ I.$$

(In such a case any bounded fixpoint formula can be proved equivalent to a first-order formula. This allows us to eliminate fixpoint operators from formulas and reduce the reasoning to classical first-order calculus).

Assume that $\models \mu\Phi.A(\Phi) \ \equiv \ I$. First let us show that $\vdash_{\mathcal{S}_\mu} \mu\Phi.A(\Phi) \ \supset \ I$. Since $\mu\Phi.A(\Phi)$ is bounded, it is equivalent to $\bigvee_{i\leq n} A^i(\bot)$, for some $n \in \omega$. Thus $\bigvee_{i\leq n} A^i(\bot)$ is a fixpoint of $A(\Phi)$, so we have:

$$\models A(\bigvee_{i\leq n} A^i(\bot)) \ \supset \ \bigvee_{i\leq n} A^i(\bot). \tag{24}$$

Formula (24) is a classical first-order formula, thus

$$\vdash_{\mathcal{S}_\mu} A(\bigvee_{i\leq n} A^i(\bot)) \ \supset \ \bigvee_{i\leq n} A^i(\bot),$$

i.e., by application of Park rule (23),

$$\vdash_{\mathcal{S}_\mu} \mu\Phi.A(\Phi) \ \supset \ \bigvee_{i\leq n} A^i(\bot).$$

Of course, $\vdash_{\mathcal{S}_\mu} \bigvee_{i\leq n} A^i(\bot) \ \supset \ I$, thus also $\vdash_{\mathcal{S}_\mu} \mu\Phi.A(\Phi) \ \supset \ I$.

In order to show that $\vdash_{\mathcal{S}_\mu} I \supset \mu\Phi.A(\Phi)$ it suffices to note that for any $k \in \omega$,

$$\vdash_{\mathcal{S}_\mu} A^k(\bot) \supset \mu\Phi.A(\Phi).$$

The proof of this claim is carried out by induction on k. If $k = 0$ then the result is obvious, since $A^0(\bot) \equiv \bot$. Now assume the claim holds for some $k \in \omega$. We thus have that

$$\vdash_{\mathcal{S}_\mu} A^k(\bot) \supset \mu\Phi.A(\Phi).$$

Since A is positive (thus monotone) w.r.t. Φ, it follows that

$$\vdash_{\mathcal{S}_\mu} A(A^k(\bot)) \supset A(\mu\Phi.A(\Phi)), \text{ i.e. } \vdash_{\mathcal{S}_\mu} A^{k+1}(\bot) \supset A(\mu\Phi.A(\Phi)).$$

By axiom (22) we have that $\vdash_{\mathcal{S}_\mu} A(\mu\Phi.A(\Phi)) \supset \mu\Phi.A(\Phi)$ which proves the claim.

The result now follows from boundedness of $\mu\Phi.A(\Phi)$. ■

7 Conclusions

We have provided a characterization result for the class of circumscribed normal logic programs which are first-order expressible. The approach used is based on a generalization of a lemma due to Ackermann for eliminating second-order quantifiers. In our case, the elimination technique results in a fixpoint formula which if bounded is first-order definable, and if not, may still be used for reasoning in a suitable fixpoint calculus. One advantage of the approach is that the characterization integrates nicely with an existing algorithm for reducing certain classes of second-order circumscription axioms to equivalent first-order formulas. In addition, the characterization provides an explicit definition of the minimized predicate, where the use of fixpoints is restricted to just the definition. Even if certain circumscribed formulas are not first-order expressible, this approach still allows reasoning about such theories in a suitable fixpoint calculus. It remains to be investigated just how far the characterization results can be extended and how practical the use of fixpoint calculi are in application domains that use circumscription as a technique.

References

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

[2] Doherty, P., Łukaszewicz, W., Szałas, A. (1994) *Computing Circumscription Revisited. A Reduction Algorithm*, Technical Report LiTH-IDA-R-94-42, Linköping University, 1994. Also in, Proc. 14th IJCAI, 1995, Montreal, Canada.[2]

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

[2]URL: `http://www.ida.liu.se/labs/rkllab/people/patdo/`.

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

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

[6] Kozen, D. (1983) *Results on the Propositional μ-calculus*, Theoretical Computer Science, **27**, 333-354.

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

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

[9] Nonnengart, A., Szałas, A. (1995) *A Fixpoint Approach to Second-Order Quantifier Elimination with Applications to Correspondence Theory*, Report of Max-Planck-Institut für Informatik, MPI-I-95-2-007, Saarbrücken, Germany.

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

[11] Szałas, A. (1992) *Axiomatizing Fixpoint Logics*, Information Processing Letters, **41**, 175-180.

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

[13] Szałas, A. (1995) *On Natural Deduction in First-Order Fixpoint Logics*, Fundamenta Informaticae (to appear).