# Literal Projection and Circumscription – Extended Version, July 4, 2009 –

Christoph Wernhard

Technische Universität Dresden
`christoph.wernhard@tu-dresden.de`

**Abstract.** We develop a formal framework intended as a preliminary step for a single knowledge representation system that provides different representation techniques in a unified way. In particular we consider first-order logic extended by techniques for second-order quantifier elimination and non-monotonic reasoning. In this paper two independent results are developed. The background for the first result is literal projection, a generalization of second-order quantification which permits, so to speak, to quantify upon an arbitrary sets of ground literals, instead of just (all ground literals with) a given predicate symbol. We introduce an operator raise that is only slightly different from literal projection and can be used to define a generalization of predicate circumscription in a straightforward and compact way. We call this variant of circumscription *scope-determined*. Some properties of raise and scope-determined circumscription, also in combination with literal projection, are then shown. A previously known characterization of consequences of circumscribed formulas in terms of literal projection is generalized from propositional to first-order logic and proven on the basis of the introduced concepts. The second result developed in this paper is a characterization stable models in terms of circumscription. Unlike traditional characterizations, it does not recur onto syntactic notions like *reduct* and fixed-point construction. It essentially renders a recently proposed "circumscription-like" characterization in a compact way, without involvement of a non-classically interpreted connective.

## Table of Contents

## 1 Introduction

We develop a formal framework intended as a preliminary step for a single knowledge representation system that provides different representation techniques in a unified way. In particular we consider first-order logic extended by techniques for second-order quantifier elimination and non-monotonic reasoning.

Second-order quantifier elimination permits to express a large number of knowledge representation techniques (see for example [6]), including abduction, modularization of knowledge bases and the processing of circumscription. It is also closely related to knowledge compilation [14]. Variants of second-order quantifier elimination also appear under names such as *computation of uniform interpolants*, *forgetting*, and *projection*. Restricted to propositional formulas it is called *elimination of Boolean quantified variables*.

We focus here on a particular generalization of second-order quantifier elimination, the *computation of literal projection* [11, 12]. Literal projection generalizes second-order quantification by permitting, so to speak, to quantify upon an *arbitrary set of ground literals*, instead of just (all ground literals with) a given predicate symbol. Literal projection allows, for example, to express predicate quantification upon a predicate just in positive or negative polarity. Eliminating such a quantifier from a formula in negation normal form results in a formula that might still contain the quantified predicate, but only in literals whose polarity is complementary to the quantified one. This polarity dependent behavior of literal projection is essential for the relationship to non-monotonic reasoning that is investigated in this paper.

In particular, we consider circumscription and, based on it, the stable model semantics, which underlies many successful applications developed during the last decade. It is well-known that the processing of circumscription can be expressed as a second-order quantifier elimination task [1]. The formalization of circumscription investigated here does not just rely on literal projection as a generalization of second-order quantification, but utilizes the polarity dependent behavior of literal projection to obtain a particular straightforward and compact characterization. The concrete contributions of this paper are:

- The introduction of an operator raise that is only slightly different from literal projection and can be used to define a generalization of parallel circumscription with varied predicates in a straightforward and compact way.

  Like literal projection, the raise operator is defined in terms of semantic properties only, and is thus independent of syntactic properties or constructions. Some properties of this operator and circumscription, also in interaction with literal projection, are then shown (Sect. 3–6).

- The characterization of consequences of circumscribed formulas in terms of literal projection. We make a known result given in [8] more precise by providing a thorough proof and generalizing it from propositional to first-order formulas.

– A definition of answer sets according to the stable model semantics in terms of circumscription. Unlike the common definitions of stable models, it does not recur onto syntactic notions like *reduct* and fixed-point construction. It is essentially an adaption of the “circumscription-like” definition recently proposed in [4, 5]. In contrast to that definition, it does not involve a specially interpreted rule forming connective (Sect. 7).

The paper is structured as follows: Preliminaries are given in Section 2, including a description of the used semantic framework and a summary of background material on literal projection. In Sections 3–7 the proper contributions of this paper are described and formally stated. Proofs of propositions and theorems can be found in Appendix A. Details on the relationship of the introduced definition of stable models to characterizations in terms of *reduct* in Appendix B. This report is a revised and extended version of the workshop contribution [13].

## 2 Notation and Preliminaries

**Symbols.** We use the following symbols, also with sub- and superscripts, to stand for items of types as indicated in the following table (precise definitions of these types are given later on in this section). They are considered implicitly as universally quantified in definition, theorem and proposition statements.

| | |
|---|---|
| $F, G$ | – Formula |
| $L$ | – Literal |
| $S$ | – Set of ground literals (also called *literal scope*) |
| $M$ | – Consistent set of ground literals |
| $I, J, K$ | – Structure |
| $\beta$ | – Variable assignment |

**Notation.** Unless specially noted, we assume that a *first-order formula* is constructed from first-order literals, truth value constants $\top, \bot$, the unary connective $\neg$, binary connectives $\wedge, \vee$ and the first-order quantifiers $\forall$ and $\exists$. We write the positive (negative) literal with atom $A$ as $+A$ $(-A)$. Variables are $x$, $y$, $z$, also with subscripts. As meta-level notation with respect to this syntax we use implication $\rightarrow$, biconditional $\leftrightarrow$ and n-ary versions of the binary connectives.

A clause is a sentence of the form $\forall x_1 \ldots \forall x_n (L_1 \vee \ldots \vee L_m)$, where $n, m \geq 0$ and the $L_i$ for $i \in \{1, \ldots, m\}$ are literals. Since all variables in a clause are universally quantified, we sometimes do not write its quantifier prefix.

We assume a fixed first-order signature with at least one constant symbol. The sets of all ground terms and all ground literals, with respect to this signature, are denoted by $\mathsf{TERMS}$ and $\mathsf{ALL}$, respectively.

**The Projection Operator and Literal Scopes.** A *formula* in general is like a first-order formula, but in its construction two further operators, $\mathsf{project}(F, S)$ and $\mathsf{raise}(F, S)$, are permitted, where $F$ is a formula and $S$ specifies a set of ground literals. We call a set of ground literals in the role as argument to $\mathsf{project}$ or $\mathsf{raise}$ a *literal scope*. We do not define here a concrete syntax for specifying literal scopes and just speak of a *literal scope*, referring to the actual literal scope in a semantic context as well as some expression that denotes it in a syntactic context. The formula $\mathsf{project}(F, S)$ is called the *literal projection* of $F$ onto $S$. Literal projection generalizes existential second-order quantification [11] (see also Sect. 4 below). It will be further discussed in this introductory section (see [11, 12] for more thorough material). The semantics of the $\mathsf{raise}$ operator will be introduced later on in Sect. 3.

**Interpretations.** We use the notational variant of the framework of Herbrand interpretations described in [11]: An *interpretation* $\mathfrak{I}$ is a pair $\langle I, \beta \rangle$, where $I$ is a *structure*, that is, a set of ground literals that contains for all ground atoms $A$ exactly one of $+A$ or $-A$, and $\beta$ is a *variable assignment*, that is, a mapping of the set of variables into $\mathsf{TERMS}$.

**Satisfaction Relation and Semantics of Projection.** The satisfaction relation between interpretations $\mathfrak{I} = \langle I, \beta \rangle$ and formulas is defined by the clauses in Tab. 1, where $L$ matches a literal, $F, F_1, F_2$ match a formula, and $S$ matches a literal scope. In the table, two operations on variable assignments $\beta$ are used: If $F$ is a formula, then $F\beta$ denotes $F$ with all variables replaced by their image in $\beta$; If $x$ is a variable and $t$ a ground term, then $\beta\frac{t}{x}$ is the variable assignment that maps $x$ to $t$ and all other variables to the same values as $\beta$. Entailment and equivalence are straightforwardly defined in terms of the satisfaction relation. Entailment: $F_1 \models F_2$ holds if and only if for all $\langle I, \beta \rangle$ such that $\langle I, \beta \rangle \models F_1$ it holds that $\langle I, \beta \rangle \models F_2$. Equivalence: $F_1 \equiv F_2$ if and only if $F_1 \models F_2$ and $F_2 \models F_1$.

Intuitively, the literal projection of a formula $F$ onto scope $S$ is a formula that expresses about literals in $S$ the same as $F$, but expresses nothing about other literals. The projection is equivalent to a formula without the projection operator, in negation normal form, where all ground instances of literals occurring in it are members of the projection scope. The semantic definition of literal projection in Tab. 1 can be alternatively expressed as: An interpretation $\langle I, \beta \rangle$ satisfies $\mathsf{project}(F, S)$ if and only if there is a structure $J$ such that $\langle J, \beta \rangle$ satisfies $F$ and $I$ can be obtained from $J$ by replacing literals that are not in $S$ with their complements. This includes the special case $I = J$, where no literals are replaced.

**Table 1.** The Satisfaction Relation with the Semantic Definition of Literal Projection

| | | |
|---|---|---|
| $\langle I, \beta \rangle \models L$ | iff$_{\text{def}}$ | $L\beta \in I$ |
| $\langle I, \beta \rangle \models \top$ | | |
| $\langle I, \beta \rangle \not\models \bot$ | | |
| $\langle I, \beta \rangle \models \neg F$ | iff$_{\text{def}}$ | $\langle I, \beta \rangle \not\models F$ |
| $\langle I, \beta \rangle \models F_1 \land F_2$ | iff$_{\text{def}}$ | $\langle I, \beta \rangle \models F_1$ and $\langle I, \beta \rangle \models F_2$ |
| $\langle I, \beta \rangle \models F_1 \lor F_2$ | iff$_{\text{def}}$ | $\langle I, \beta \rangle \models F_1$ or $\langle I, \beta \rangle \models F_2$ |
| $\langle I, \beta \rangle \models \forall x\, F$ | iff$_{\text{def}}$ | for all $t \in \mathsf{TERMS}$ it holds that $\langle I, \beta\frac{t}{x} \rangle \models F$ |
| $\langle I, \beta \rangle \models \exists x\, F$ | iff$_{\text{def}}$ | there exists a $t \in \mathsf{TERMS}$ such that $\langle I, \beta\frac{t}{x} \rangle \models F$ |
| $\langle I, \beta \rangle \models \mathsf{project}(F, S)$ | iff$_{\text{def}}$ | there exists a $J$ such that $\langle J, \beta \rangle \models F$ and $J \cap S \subseteq I$ |

**Relation to Conventional Model Theory.** Literal sets as components of interpretations permit the straightforward definition of the semantics of literal projection given in the last clause in Tab. 1. The set of literals $I$ of an interpretation $\langle I, \beta \rangle$ is called *"structure"*, since it can be considered as representation of a structure in the conventional sense used in model theory: The domain is the set of ground terms. Function symbols $f$ with arity $n \geq 0$ are mapped to functions $f'$ such that for all ground terms $t_1, ..., t_n$ it holds that $f'(t_1, ..., t_n) = f(t_1, ..., t_n)$. Predicate symbols $p$ with arity $n \geq 0$ are mapped to $\{\langle t_1, ..., t_n \rangle \mid +p(t_1, ..., t_n) \in I\}$. Moreover, an interpretation $\langle I, \beta \rangle$ represents a conventional second-order interpretation [2] (if predicate variables are considered as distinguished predicate symbols): The structure in the conventional sense corresponds to $I$, as described above, except that mappings of predicate variables are omitted. The assignment is $\beta$, extended such that all predicate variables $p$ are mapped to $\{\langle t_1, ..., t_n \rangle \mid +p(t_1, ..., t_n) \in I\}$.

**Some More Notation.** The following table specifies symbolic notation for (i) the complement of a literal, (ii) the set of complement literals of a given set of literals, (iii) the set complement of a set of ground literals, (iv) the set of all positive ground literals, (v) the set of all negative ground literals, (vi) the set of all ground literals whose predicate symbol is from a given set, and (vii, viii) a structure that is like a given one, except that it assigns given truth values to a single given ground atom or to all ground atoms in a given set, respectively.

(i) If $A$ is an atom, then $\widetilde{+A} \stackrel{\text{def}}{=} -A$, and $\widetilde{-A} \stackrel{\text{def}}{=} +A$. The literal $\widetilde{L}$ is called the *complement* of $L$.
(ii) $\widetilde{S} \stackrel{\text{def}}{=} \{\widetilde{L} \mid L \in S\}$.
(iii) $\overline{S} \stackrel{\text{def}}{=} \mathsf{ALL} - S$.
(iv) $\mathsf{POS} \stackrel{\text{def}}{=} \{+A \mid +A \in \mathsf{ALL}\}$.
(v) $\mathsf{NEG} \stackrel{\text{def}}{=} \{-A \mid -A \in \mathsf{ALL}\}$.
(vi) $\hat{P}$ is the set of all ground literals whose predicate is $P$ or is in $P$, resp., where $P$ is a predicate symbol, or a tuple or set of predicate symbols.
(vii) $I[L] \stackrel{\text{def}}{=} (I - \{\widetilde{L}\}) \cup \{L\}$.
(viii) $I[M] \stackrel{\text{def}}{=} (I - \widetilde{M}) \cup M$.

**Literal Base and Related Concepts.** The *literal base* $\mathcal{L}(F)$ of a first-order formula $F$ in negation normal form is the set of all ground instances of literals in $F$. The following formal definition generalizes this notion straightforwardly for formulas that are not in negation normal form and possibly include the $\mathsf{project}$ and $\mathsf{raise}$ operator: $\mathcal{L}(L)$ is the set of all ground instances of $L$; $\mathcal{L}(\top) \stackrel{\text{def}}{=} \mathcal{L}(\bot) \stackrel{\text{def}}{=} \{\}$; $\mathcal{L}(\neg F) \stackrel{\text{def}}{=} \widetilde{\mathcal{L}(F)}$; $\mathcal{L}(F \otimes G) \stackrel{\text{def}}{=} \mathcal{L}(F) \cup \mathcal{L}(G)$ if $\otimes$ is $\land$ or $\lor$; $\mathcal{L}(\otimes x F) \stackrel{\text{def}}{=} \mathcal{L}(\otimes(F, S)) \stackrel{\text{def}}{=} \mathcal{L}(F)$ if $\otimes$ is a quantifier or the $\mathsf{project}$ or $\mathsf{raise}$ operator, respectively.

We call the set of ground literals "about which a formula expresses something" its *essential literal base*, made precise in Def. 1 (see [11, 12] for a more thorough discussion). It can be proven that essential literal base of a formula is a subset of its literal base. The essential literal base is independent of syntactic properties: equivalent formulas have the same essential literal base.

**Table 2.** Properties of Literal Projection

| | |
|---|---|
| (i) | $F \models \mathsf{project}(F, S)$ |
| (ii) | *If* $F_1 \models F_2$, *then* $\mathsf{project}(F_1, S) \models \mathsf{project}(F_2, S)$ |
| (iii) | *If* $F_1 \equiv F_2$, *then* $\mathsf{project}(F_1, S) \equiv \mathsf{project}(F_2, S)$ |
| (iv) | *If* $S_1 \supseteq S_2$, *then* $\mathsf{project}(F, S_1) \models \mathsf{project}(F, S_2)$ |
| (v) | $\mathsf{project}(\mathsf{project}(F, S_1), S_2) \equiv \mathsf{project}(F, S_1 \cap S_2)$ |
| (vi) | $F_1 \models \mathsf{project}(F_2, S)$ *if and only if* $\mathsf{project}(F_1, S) \models \mathsf{project}(F_2, S)$ |
| (vii) | $\mathsf{project}(F, \mathsf{ALL}) \equiv F$ |
| (viii) | $\mathsf{project}(F, \mathcal{L}(F)) \equiv F$ |
| (ix) | $\mathsf{project}(\top, S) \equiv \top$ |
| (x) | $\mathsf{project}(\bot, S) \equiv \bot$ |
| (xi) | $F$ *is satisfiable if and only if* $\mathsf{project}(F, S)$ *is satisfiable* |
| (xii) | $\mathcal{L}_{\mathcal{E}}(\mathsf{project}(F, S)) \subseteq S$ |
| (xiii) | $\mathcal{L}_{\mathcal{E}}(\mathsf{project}(F, S)) \subseteq \mathcal{L}_{\mathcal{E}}(F)$ |
| (xiv) | *If* $\mathsf{project}(F, S) \models F$, *then* $\mathcal{L}_{\mathcal{E}}(F) \subseteq S$ |
| (xv) | $\mathsf{project}(F, S) \equiv \mathsf{project}(F, \mathcal{L}(F) \cap S)$ |
| (xvi) | $F_1 \models F_2$ *if and only if* $\mathsf{project}(F_1, \mathcal{L}(F_2)) \models F_2$ |
| (xvii) | *If no instance of* $L$ *is in* $S$, *then* $\mathsf{project}(L, S) \equiv \top$ |
| (xviii) | *If all instances of* $L$ *are in* $S$, *then* $\mathsf{project}(L, S) \equiv L$ |
| (xix) | $\mathsf{project}(F_1 \lor F_2, S) \equiv \mathsf{project}(F_1, S) \lor \mathsf{project}(F_2, S)$ |
| (xx) | $\mathsf{project}(F_1 \land F_2, S) \models \mathsf{project}(F_1, S) \land \mathsf{project}(F_2, S)$ |
| (xxi) | *If* $\mathcal{L}(F_1) \cap \widetilde{\mathcal{L}(F_2)} \subseteq S \cap \widetilde{S}$ *then* |
| | $\mathsf{project}(F_1 \land F_2, S) \equiv \mathsf{project}(F_1, S) \land \mathsf{project}(F_2, S)$ |
| (xxii) | $\mathsf{project}(\exists x F, S) \equiv \exists x\ \mathsf{project}(F, S)$ |
| (xxiii) | $\mathsf{project}(\forall x F, S) \models \forall x\ \mathsf{project}(F, S)$ |

**Definition 1 (Essential Literal Base).** The *essential literal base* of a formula $F$, in symbols $\mathcal{L}_{\mathcal{E}}(F)$, is defined as $\mathcal{L}_{\mathcal{E}}(F) \overset{\text{def}}{=} \{L \mid L \in \mathsf{ALL}$ and there exists an interpretation $\langle I, \beta\rangle$ such that $\langle I, \beta\rangle \models F$ and $\langle I[\widetilde{L}], \beta\rangle \not\models F\}$.

**Properties of Literal Projection.** A summary of properties of literal projection is displayed in Tab. 2 and 3. Most of them follow straightforwardly from the semantic definition of project shown in Tab. 1 [12]. The more involved proof of Tab. 2.xxi (and the related Tab. 3.v) can be found in [11, 12]. The properties in Tab. 3 strengthen properties in Tab. 2, but apply only to formulas that satisfy a condition related to their essential literal base. These formulas are called $\mathcal{E}$-formulas and are defined as follows:

**Definition 2 ($\mathcal{E}$-Formula).** A formula $F$ is called *$\mathcal{E}$-formula* if and only if for all interpretations $\langle I, \beta\rangle$ and consistent sets of ground literals $M$ such that $\langle I, \beta\rangle \models F$ and $M \cap \mathcal{L}_{\mathcal{E}}(F) = \emptyset$ it holds that $\langle I[\widetilde{M}], \beta\rangle \models F$.

First-order formulas in negation normal form without existential quantifier – including propositional formulas and first-order clausal formulas – are $\mathcal{E}$-formulas. Being an $\mathcal{E}$-formula is a property that just depends on the semantics of a formula, that is, an equivalent to an $\mathcal{E}$-formula is also an $\mathcal{E}$-formula. See [11, 12] for more discussion.[1]

[1] An example that is not an $\mathcal{E}$-formula is the sentence $F \overset{\text{def}}{=} \forall x\ {+\mathsf{r}(x, \mathsf{f}(x))} \land \forall x \forall y(-\mathsf{r}(x, y) \lor +\mathsf{r}(x, \mathsf{f}(y))) \land \exists x \forall y(-\mathsf{r}(x, y) \lor +\mathsf{p}(y))$. Let the domain be the set

**Table 3.** Properties of Literal Projection for $\mathcal{E}$-Formulas $E$

| | | |
|---|---|---|
| (i) | $\mathsf{project}(E, \mathcal{L}_{\mathcal{E}}(E)) \equiv E$ | (strengthens Tab. 2.viii) |
| (ii) | $\mathcal{L}_{\mathcal{E}}(E) \subseteq S$ *if and only if* $\mathsf{project}(E, S) \equiv E$ | (strengthens Tab. 2.xiv) |
| (iii) | $\mathsf{project}(E, S) \equiv \mathsf{project}(E, \mathcal{L}_{\mathcal{E}}(E) \cap S)$ | (strengthens Tab. 2.xv) |
| (iv) | $F \models E$ *if and only if* $\mathsf{project}(F, \mathcal{L}_{\mathcal{E}}(E)) \models E$ | (strengthens Tab. 2.xvi) |
| (v) | *If* $\mathcal{L}_{\mathcal{E}}(E_1) \cap \widetilde{\mathcal{L}_{\mathcal{E}}(E_2)} \subseteq S \cap \widetilde{S}$ *then* $\mathsf{project}(E_1 \land E_2, S) \equiv \mathsf{project}(E_1, S) \land \mathsf{project}(E_2, S)$ | (strengthens Tab. 2.xxi) |

## 3 The Raise Operator

The following operator raise is only slightly different from literal projection and, as we will see later on, can be used to define a generalization of parallel circumscription with varied predicates in a straightforward and compact way.

**Definition 3 (Raise).**

$$\langle I, \beta \rangle \models \mathsf{raise}(F, S) \ \text{iff}_{\text{def}} \ \begin{array}{l}\text{there exists a } J \text{ such that}\\ \langle J, \beta \rangle \models F \text{ and}\\ J \cap S \subset I \cap S.\end{array}$$

The definition of raise is identical to that of literal projection (Tab. 1), with the exception that $J \cap S$ and $I \cap S$ are related by the *proper subset* instead of the *subset* relationship (assuming that condition $J \cap S \subseteq I$ in Tab. 1 is written equivalently as $J \cap S \subseteq I \cap S$).

The name "raise" suggests that a model $\langle I, \beta \rangle$ of $\mathsf{raise}(F, S)$ is not "the lowest" model of $F$, in the sense that there exists another model $\langle J, \beta \rangle$ of $F$ with the property $J \cap S \subset I \cap S$. An equivalent specification of the condition $J \cap S \subset I \cap S$ in the definition of raise provides further intuition on its effect: A literal scope $S$ can be partitioned into three disjoint subsets $S_p, S_n, S_{pn}$ such that $S_p$ ($S_n$) is the set of positive (negative) literals in $S$ whose complement is not in $S$, and $S_{pn}$ is the set of literals in $S$ whose complement is also in $S$. Within Def. 3, the condition $J \cap S \subset I \cap S$ can then be equivalently expressed by the conjunction of $J \cap (S_p \cup S_n) \subset I \cap (S_p \cup S_n)$ and $J \cap S_{pn} = I \cap S_{pn}$. That is, with respect to members of $S$ whose complement is not in $S$, the structure $J$ must be a proper subset of $I$, and with respect to the other members of $S$ it must be identical to $I$.

Proposition 1 below shows some properties of the raise operator: It is monotonic (Prop. 1.i). From this follows that it is a "semantic" operator in the sense that for equivalent arguments the values are equivalent too (Prop. 1.ii). Like projection, the raise operator distributes over disjunction (Prop. 1.iii). Proposition 1.iv follows from monotonicity. Proposition 1.v shows that for scopes that contain exactly the same atoms positively as well as negatively, raise is inconsistent. Propositions 1.vi and 1.vi show the interplay of raise with projection onto

of all terms $\mathsf{f}^n(\mathsf{a})$ where $n \geq 0$. For each member $T$ of the domain it can be verified that $+\mathsf{p}(T) \notin \mathcal{L}_{\mathcal{E}}(F)$. On the other hand, an interpretation that contains $-\mathsf{p}(T)$ for all members $T$ of the domain cannot be a model of $F$.

the same scope. Proposition 1.viii provides a characterization of *literal* projection in terms of raise and *atom* projection [11], a restricted form of projection where the polarity of the scope members is not taken into account, which can be expressed as literal projection onto scopes $S$ constrained by $S = \widetilde{S}$.

**Proposition 1 (Properties of Raise).**

(i) *If* $F_1 \models F_2$, *then* $\mathsf{raise}(F_1, S) \models \mathsf{raise}(F_2, S)$.
(ii) *If* $F_1 \equiv F_2$, *then* $\mathsf{raise}(F_1, S) \equiv \mathsf{raise}(F_2, S)$.
(iii) $\mathsf{raise}(F_1 \lor F_2, S) \equiv \mathsf{raise}(F_1, S) \lor \mathsf{raise}(F_2, S)$.
(iv) $\mathsf{raise}(F_1 \land F_2, S) \models \mathsf{raise}(F_1, S) \land \mathsf{raise}(F_2, S)$.
(v) *If* $S = \widetilde{S}$, *then* $\mathsf{raise}(F, S) \equiv \bot$.
(vi) $\mathsf{raise}(\mathsf{project}(F, S), S) \equiv \mathsf{raise}(F, S)$.
(vii) $\mathsf{project}(\mathsf{raise}(F, S), S) \equiv \mathsf{raise}(F, S)$.
(viii) $\mathsf{project}(F, S) \equiv \mathsf{project}(F, S \cup \widetilde{S}) \lor \mathsf{raise}(F, S)$.

## 4 Definition of Circumscription in Terms of Raise

The following definition specifies a characterization of circumscription in terms of raise, as we will first outline informally and then show more precisely.

**Definition 4 (Scope-Determined Circumscription).** The *scope-determined circumscription* of formula $F$ with respect to literal scope $S$, in symbols $\mathsf{circ\text{-}s}(F, S)$, is a formula that involves the raise operator and is defined as:

$$\mathsf{circ\text{-}s}(F, S) \stackrel{\text{def}}{=} F \land \neg\mathsf{raise}(F, S).$$

The name *scope-determined* indicates that a literal scope, that is, a set of ground literals, is used to determine what is circumscribed. *Parallel circumscription of predicate constants $P$ in sentence $F$ with varied predicate constants $Z$* [9] can be expressed as the special case of scope-determined circumscription onto a scope that is the set of all ground literals $L$ such that either

1. $L$ is positive and its predicate is in $P$, or
2. The predicate of $L$ is neither in $P$ nor in $Z$.

In other words, the scope contains the circumscribed predicates just positively (the positive literals according to item 1.), and the "fixed" predicates in full (all positive as well as negative literals according to item 2.). Since the literal scope $S$ in $\mathsf{circ\text{-}s}(F, S)$ can be an arbitrary set of literals, scope-determined circumscription is more general than parallel circumscription with varied predicates: Model maximization conditions can be expressed by means of scopes that contain negative literals but not their complements. Furthermore, it is possible to express minimization, maximization and variation conditions that apply only to a subset of the instances of a predicate.

We now make precise how scope-determined circumscription relates to the established definition of predicate circumscription by means of second-order quantification [9, 1, 6]. The following definition specifies a second-order sentence

$\mathsf{CIRC}[F; P; Z]$ that is called *parallel circumscription of predicate constants $P$ in $F$ with varied predicate constants $Z$* in [9] and is straightforwardly equivalent to the sentence called *second-order circumscription of $P$ in $F$ with variable $Z$* in [1, 6]:

**Definition 5 (Second-Order Circumscription).** Let $F$ be a first-order sentence and let $P, P', Z, Z'$ be mutually disjoint tuples of distinct predicate symbols such that: $P = p_1, \ldots, p_n$ and $P' = p'_1, \ldots, p'_n$ where $n \geq 0$; both $Z$ and $Z'$ have the same length $\geq 0$; members of $P'$ and $P$ with the same index, as well as members of $Z'$ and $Z$ with the same index, are of the same arity; and $P'$ and $Z'$ do not contain predicate symbols in $F$. Let $F'$ be the formula that is obtained from $F$ by replacing each predicate symbol that is in $P$ or $Z$ by the predicate symbol with the same index in $P'$ or $Z'$, respectively. For $i \in \{1, \ldots, n\}$ let $\overline{x}_i$ stand for $x_1, \ldots, x_k$, where $k$ is the arity of predicate symbol $p_i$. Let $P' < P$ stand for

$$\bigwedge_{i=1}^{n} \forall \overline{x}_i (p'_i(\overline{x}_i) \rightarrow p_i(\overline{x}_i)) \land \neg \bigwedge_{i=1}^{n} \forall \overline{x}_i (p'_i(\overline{x}_i) \leftrightarrow p_i(\overline{x}_i))).$$

Considering the predicate symbols in $P'$ and $Z'$ as predicate variables, the *second-order circumscription of $P$ in $F$ with variable $Z$*, written $\mathsf{CIRC}[F; P; Z]$, is then defined as:

$$\mathsf{CIRC}[F; P; Z] \overset{\text{def}}{=} F \land \neg\exists P', Z' \ (F' \land P' < P).$$

Existential second-order quantification can be straightforwardly expressed with literal projection: $\exists p\ G$ corresponds to $\mathsf{project}(G, S)$, where $S$ is the set of all ground literals with a predicate other than $p$. From Tab. 2.xv it can be derived that also a smaller projection scope is sufficient: $\mathsf{project}(G, S)$ is equivalent to $\mathsf{project}(G, S')$ for all subsets $S'$ of $S$ that contain those literals of $S$ whose predicate symbol occurs in $G$. Accordingly, $\mathsf{CIRC}[F; P; Z]$ can be expressed straightforwardly in terms of literal projection instead of the second-order quantification:

**Definition 6 (Second-Order Circumscription in Terms of Projection).** Let $F$ be a first-order formula and let $P, P', Z, Z'$ be tuples of predicate symbols as specified in the definition of $\mathsf{CIRC}$ (Def. 5). Let $Q$ be the set of predicate symbols in $F$ that are neither in $P$ nor in $Z$. Then $\mathsf{CIRC\text{-}PROJ}[F; P; Z]$ is a formula with the projection operator, defined as:

$$\mathsf{CIRC\text{-}PROJ}[F; P; Z] \overset{\text{def}}{=} F \land \neg\mathsf{project}(F' \land P' < P,\ \hat{P} \cup \hat{Q}).$$

The $Q$ parameter in Def. 6 is the set of the "fixed" predicates. The set of literals $(\hat{P} \cup \hat{Q})$ suffices as projection scope, since the quantified body of the right conjunct of $\mathsf{CIRC}[F; P; Z]$, that is, $(F' \land P' < P)$, contains – aside of the quantified predicate symbols from $P', Z'$ – just predicate symbols that are in $P$ or in $Q$.

The following theorem makes precise how second-order circumscription can be expressed with scope-determined circumscription. Its proof formally relates second-order circumscription expressed by projection (Def. 6) with circumscription defined in terms of of the $\mathsf{raise}$ operator (Def. 4).

**Theorem 1 (Second-Order and Scope-Determined Circumscription).** *Let $F$ be a first-order formula and let $P, P', Z, Z'$ be tuples of predicate symbols as specified in the definition of* $\mathsf{CIRC}$ *(Def. 5). Let $Q$ be the set of predicate symbols in $F$ that are neither in $P$ nor in $Z$. Then*

$$\mathsf{CIRC\text{-}PROJ}[F; P; Z] \equiv \mathsf{circ\text{-}s}(F, (\hat{P} \cap \mathsf{POS}) \cup \hat{Q}).$$

## 5 Well-Foundedness

As discussed in [9], circumscription can in general only be applied usefully to a sentence $F$ if all models of $F$ extend some model of $F$ that is minimal with respect to the circumscribed predicates. The concept of well-foundedness [9] makes this property precise. We show that it can be expressed in a compact way in terms of projection. This characterization facilitates to prove properties of circumscription that have well-foundedness as a precondition, as for example Prop. 3 and Theorem 2 below.

**Definition 7 (Well-Founded Formula).** $F$ is called *well-founded with respect to $S$* if and only if

$$F \models \mathsf{project}(\mathsf{circ\text{-}s}(F, S), S).$$

In this definition, the literal scope $S$ can be an arbitrary set of literals, corresponding to variants of circumscription as indicated in Sect. 4. We now explicate how this definition renders the definition of well-foundedness in [9], which is defined for the special case of circumscription of a single predicate $p$ with varied predicates $Z$. That definition is as as follows (adapted to our notation): Let $F$ be a first-order sentence, $p$ be predicate symbol and $Z$ be a tuple of predicate symbols. The sentence $F$ is called *well-founded with respect to* $(p; Z)$ if for every model $\mathfrak{I}$ of $F$ there exists a model $\mathfrak{J}$ of $\mathsf{CIRC}[F; p; Z]$ such that $\mathfrak{I}$ and $\mathfrak{J}$ differ only in how they interpret $p$ and $Z$ and the extent of $p$ in $J$ is a (not necessarily strict) subset of its extent in $I$. We can convert this definition straightforwardly into our semantic framework: Let $Q$ be the set of predicate symbols in $F$ that are different from $p$ and not in $Z$. The sentence $F$ is then well-founded with respect to $(p; Z)$ if for all interpretations $\langle I, \beta \rangle$ such that $\langle I, \beta \rangle \models F$ there exists an interpretation $\langle J, \beta \rangle$ such that (1.) $\langle J, \beta \rangle \models \mathsf{CIRC\text{-}PROJ}[F; p; Z]$, (2.) $J \cap \hat{p} \cap \mathsf{POS} \subseteq \cap I$, and (3.) $J \cap \hat{Q} = I \cap \hat{Q}$. The $\mathsf{project}$ operator allows to express this converted definition compactly: Let $S$ be the literal scope $((\hat{p} \cap \mathsf{POS}) \cup \hat{Q})$. By Theorem 1, $\mathsf{CIRC\text{-}PROJ}[F; p; Z]$ is equivalent to $\mathsf{circ\text{-}s}(F, S)$. Furthermore, given that $I$ and $J$ are structures and $\hat{Q} = \widetilde{\hat{Q}}$, the conjunction of items (2.) and (3.) above is equivalent to $J \cap S \subseteq I$. By the definition of $\mathsf{project}$ (Tab. 1), the statement that there exists an interpretation $\langle J, \beta \rangle$ satisfying items (1.)–(3.) can be expressed as $\langle I, \beta \rangle \models \mathsf{project}(\mathsf{circ\text{-}s}(F, S), S)$.

## 6 Interplay of Projection and Circumscription

The following proposition shows properties of projection nested within circumscription. It is independent of the *well-founded* property.

**Proposition 2 (Circumscribing Projections).**
(i) $\mathsf{circ\text{-}s}(F, S) \models \mathsf{circ\text{-}s}(\mathsf{project}(F, S), S)$.
(ii) $\mathsf{circ\text{-}s}(\mathsf{project}(F, S), S) \models \mathsf{circ\text{-}s}(\mathsf{project}(F, S \cup \widetilde{S}), S)$.

In the special case where $S \cup \widetilde{S} = \mathsf{ALL}$, which holds for example if $S = \mathsf{POS}$, the two entailments Prop. 2.i and Prop. 2.ii can be combined to the equivalence $\mathsf{circ\text{-}s}(\mathsf{project}(F, S), S) \equiv \mathsf{circ\text{-}s}(F, S)$. From this equivalence, it can be derived that two formulas which express the same about positive literals (that is, have equivalent projections onto $\mathsf{POS}$) have the same minimal models (that is, have equivalent circumscriptions for scope $\mathsf{POS}$).

The following proposition concerns circumscription nested within projection. It is a straightforward consequence of the definition of *well-founded* along with Tab. 2.vi and 2.ii.

**Proposition 3 (Projecting Circumscriptions).** *If $F$ is well-founded with respect to $S$, then*

$$\mathsf{project}(\mathsf{circ\text{-}s}(F, S), S) \equiv \mathsf{project}(F, S).$$

From this proposition follows that if two well-founded formulas have equivalent circumscriptions for some scope, then also their projections onto that scope are equivalent. With properties of projection, it also follows that if $S$ is a positive literal scope (that is, $S \subseteq \mathsf{POS}$) then $\mathsf{project}(\mathsf{circ\text{-}s}(F, \mathsf{POS}), S) \equiv \mathsf{project}(F, S)$. This equivalence can be applied to provide a straightforward justification for applying methods to compute minimal models also to projection computation onto positive scopes: We consider methods that compute for a given input formula $F$ an output formula $F'$ that satisfies syntactic criteria (for example correspondence to a tableau) which permit projection computation with low computational effort, such that projection computation is in essence already performed by computing $F'$. Assume that the output formula has the same minimal models as the input formula, that is, $\mathsf{circ\text{-}s}(F', \mathsf{POS}) \equiv \mathsf{circ\text{-}s}(F, \mathsf{POS})$. If $F'$ is well-founded, for positive literal scopes $S$ it then follows that $\mathsf{project}(F', S) \equiv \mathsf{project}(F, S)$. A tableau constructed by the hyper tableau calculus can indeed be viewed as representation of such a formula $F'$ [14].

*Literal forgetting* is a variant of literal projection that can be defined as $\mathsf{forget}(F, S) \overset{\text{def}}{=} \mathsf{project}(F, \overline{S})$ and is investigated for propositional logic in [8]. It is shown there that circumscription, or more precisely the formulas that are entailed by circumscriptions, can be characterized in terms of literal forgetting. Two such characterizations are given as Proposition 22 in [8], where the simpler one applies if the literal base of the entailed formula is restricted in a certain way.

These characterizations are rendered here in terms of literal projection as Theorem 2.ii and 2.iii below, where we generalize and make more precise the statements given in [8] in the following four respects: (1.) We generalize the characterizations to first-order logic. (2.) We use preconditions on the entailed formulas that do not refer to their literal base, a syntactic notion. This is discussed more in depth after the theorem statement. (3.) We provide a thorough proof. The proof given in [8] just shows the characterizations as straightforward consequence of [10, Theorems 2.5 and 2.6], for which in turn no proof is given, neither in [10], nor in [7] which is referenced by [10]. (4.) We add a third basic variant (Theorem 2.i) for consequents that are stronger restricted than in Theorem 2.ii.

This basic variant is actually a straightforward generalization of Proposition 12 in [9], which is introduced as capturing the intuition that, under the assumption of well-foundedness, a circumscription provides no new information about the fixed predicates, and only "negative" additional information about the circumscribed predicates.

**Theorem 2 (Consequences of Circumscription).** *If $F$ is well-founded with respect to $S$, then*

$$\mathsf{circ\text{-}s}(F,S) \models G$$

*is equivalent to at least one of the following entailments, depending on additional preconditions on $G$:*

(i) $F \models G$, *if* $G \equiv \mathsf{project}(G,S)$;

(ii) $F \models \mathsf{project}(F \wedge G, S)$, *if* $G \equiv \mathsf{project}(G, S \cup \widetilde{S})$;

(iii) $F \models \mathsf{project}(F \wedge \neg\mathsf{project}(F \wedge \neg G, S), S)$.

Theorems 2.ii and 2.i involve preconditions on $G$ which are expressed somewhat abstractly with the $\mathsf{project}$ operator. This is a way to generalize more concrete preconditions which come in two variants: One for *$\mathcal{E}$-formulas* $G$, based on the semantic notion of *essential* literal base ($\mathcal{L}_{\mathcal{E}}$), and the other for formulas $G$ in general, based on the syntactic notion of literal base ($\mathcal{L}$). We have seen a similar split into variants before, with properties of projection that are displayed in Table 3 and 2, respectively.

If $G$ is an $\mathcal{E}$-formula, then by Tab. 3.ii the precondition $G \equiv \mathsf{project}(G,S)$ of Theorem 2.i is equivalent to $\mathcal{L}_{\mathcal{E}}(G) \subseteq S$. For arbitrary formulas $G$, by Tab. 2.viii and 2.xv the precondition $G \equiv \mathsf{project}(G,S)$ is implied by the existence of a formula $G'$ such that $G' \equiv G$ and $\mathcal{L}(G') \subseteq S$. The precondition $G \equiv \mathsf{project}(G, S \cup \widetilde{S})$ of Theorem 2.ii can of course be expressed analogously in two such variants.

## 7 Answer Sets with Stable Model Semantics

In [4, 5] a characterization of stable models in terms of a formula translation that is *similar* to the second-order circumscription has been presented. Roughly, it differs from circumscription in that only certain *occurrences* of predicates are circumscribed, which are identified by their position with respect to a non-classical rule forming operator. We develop a variant of this characterization of stable models that is *in terms of* circumscription. It involves no non-classical operators. Instead, to indicate occurrences be circumscribed, it puts atoms into term position, wrapped by one of two special predicates.

We let the symbols $\circ$ and $\bullet$ denote these predicates. They are unary, and we write them without parentheses – for example $\bullet \mathsf{p}(\mathsf{a})$. With them, we now formally define a notion of *logic program*. Its correspondence to the more conventional view of a logic program as formed by non-classical operators will then be indicated.

**Definition 8 (Logic Program).**
(i) A *rule clause* is a clause[2] of the form

$$\bigvee_{i=1}^{m} -\circ A_i \ \vee \bigvee_{i=1}^{n} +\bullet B_i \ \vee \bigvee_{i=1}^{o} +\circ C_i \ \vee \bigvee_{i=1}^{p} -\bullet D_i,$$

where $k, m, n, o, p \geq 0$ and the subscripted $A, B, C, D$ are terms.
(ii) For a rule clause of the form specified in (8.i), the rule clause $(\bigvee_{i=1}^{m} -\circ A_i \vee \bigvee_{i=1}^{n} +\bullet B_i)$ is called its *negated body*, and the rule clause $(\bigvee_{i=1}^{o} +\circ C_i \vee \bigvee_{i=1}^{p} -\bullet D_i)$ is called its *head*.
(iii) A *logic program* is a conjunction of rule clauses.
(iv) The symbol $\mathsf{SYNC}$ stands for the formula $\forall x(+\bullet x \leftrightarrow +\circ x)$.

Conventionally, logic programs are typically notated by means of a special syntax with truth value constants $(\top, \bot)$, conjunction (,), disjunction (;), negation as failure ($\mathsf{not}$) and rule forming $(\rightarrow)$ as connectives. A rule clause according to (Def. 8.i) is then written as a rule of the form

$$A_1, \ldots, A_m, \mathsf{not}\, B_1, \ldots, \mathsf{not}\, B_n \rightarrow C_1; \ldots; C_o; \mathsf{not}\, D_1; \ldots; \mathsf{not}\, D_p, \qquad \text{(i)}$$

where $m, n, o, p \geq 0$ and the subscripted $A, B, C, D$ are atoms. If $m = n = 0$, then the left side of the rule is $\top$; if $o = p = 0$, then the right side is $\bot$.

The following definition specifies a formula $\mathsf{ans}(F)$ whose models are exactly those interpretations that represent an answer set of $F$ according to the stable model semantics.

**Definition 9 (Answer Set).** For all formulas $F$ with $\bullet$ and $\circ$ as only predicate symbols:

$$\mathsf{ans}(F) \overset{\text{def}}{=} \mathsf{circ\text{-}s}(F, \mathsf{POS} \cup \hat{\bullet}) \wedge \mathsf{SYNC}.$$

In the definition of $\mathsf{ans}(F)$, the scope of the circumscription, $(\mathsf{POS} \cup \hat{\bullet})$, is equal to $((\hat{\circ} \cap \mathsf{POS}) \cup \hat{\bullet})$ which matches the right side of Theorem 1, indicating that $\mathsf{ans}(F)$ can also be expressed in terms of second-order circumscription.

We now explicate the relationship of the characterization of stable models by $\mathsf{ans}(F)$ to the characterization in [4, 5], and justify in this way that $\mathsf{ans}(F)$ indeed characterizes stable models. Relationships to reduct-based characterizations of answer sets are shown in Appendix B. We limit our considerations to logic programs according to Def. 8.iii, which are clausal sentences (the characterization in [4, 5] applies also to nonclausal sentences).

Let $F$ be a logic program. Let $P = p_1, \ldots, p_n$ be the function symbols of the principal terms in $F$ (that is, the predicate symbols if the wrapper predicates

[2] Recall that a *clause* as specified in Sect. 2 may contain universally quantified variables. The implicit quantifier prefixes of clauses are not written in this definition.

$\circ$ and $\bullet$ are dropped). Let $P' = p'_1, \ldots, p'_n$ and $Q = q_1, \ldots, q_n$ be tuples of distinct predicate symbols which are disjoint with each other and with $P$. We use the following notation to express variants of $F$ that are obtained by replacing predicate symbols:

- We write $F$ also as $F[\circ, \bullet]$, to indicate that $\circ$ and $\bullet$ occur in it.
- The formula $F[U, V]$, where $U = u_1, \ldots, u_n$ and $V = v_1, \ldots, v_n$ are tuples of predicate symbols is $F[\circ, \bullet]$ with all atoms $\circ(p_i(\overline{t}))$ replaced by $u_i(\overline{t})$ and all atoms $\bullet(p_i(\overline{t}))$ replaced by $u_i(\overline{t})$, where $\overline{t}$ matches the respective argument terms. As a special case, $F[P, P]$ is then $F[\circ, \bullet]$ with all atoms of the form $\circ A$ or $\bullet A$ replaced by $A$.

Let $\mathsf{cnv}(F)$ denote $F$ converted into the syntax of logic programs with non-classical operators used by [4, 5] (an explicit such conversion is given in Appendix B as Def. B3.iii). Let $\mathsf{SM}(\mathsf{cnv}(F))$ be the second-order sentence that characterizes the stable models of $\mathsf{cnv}(F)$ according to [4, 5]. The following equivalence can be verified, where $P' < P$ has the same meaning as in Def. 5:

$$\mathsf{SM}(\mathsf{cnv}(F)) \equiv F[P, P] \land \neg\exists P'(F[P', P] \land P' < P). \tag{ii}$$

The right side of equivalence (ii) is not a second-order circumscription, since $P$ occurs in $F[P', P]$ as well as in $P' < P$. To fit it into the circumscription scheme, we replace the occurrences of $P$ in $F[P', P]$ by $Q$ and add the requirement that $P$ and $Q$ are equivalent: Let $(P \leftrightarrow Q)$ stand for $\bigwedge_{i=1}^{n}(p_i(\overline{x}_i) \leftrightarrow q_i(\overline{x}_i))$, where $\overline{x}_i$ has the same meaning as in Def. 5. The following equivalences then hold:

$$\begin{aligned} & \mathsf{SM}(\mathsf{cnv}(F)) \land (P \leftrightarrow Q) && \text{(iii)} \\ \equiv\ & F[P, Q] \land \neg\exists P'(F[P', Q] \land P' < P) \land (P \leftrightarrow Q) && \text{(iv)} \\ \equiv\ & \mathsf{CIRC}[F[P, Q]; P; \emptyset] \land (P \leftrightarrow Q). && \text{(v)} \end{aligned}$$

To get rid of the biconditionals $(P \leftrightarrow Q)$ in (iii), projection can be employed: From $\mathsf{SM}(\mathsf{cnv}(F)) \equiv \mathsf{project}(\mathsf{SM}(\mathsf{cnv}(F)) \land (P \leftrightarrow Q), \hat{P})$ it follows that

$$\mathsf{SM}(\mathsf{cnv}(F)) \equiv \mathsf{project}(\mathsf{CIRC}[F[P, Q]; P; \emptyset] \land (P \leftrightarrow Q), \hat{P}). \tag{vi}$$

Based on equivalence (vi), the correspondence of $\mathsf{ans}(F)$ to $\mathsf{SM}(\mathsf{cnv}(F))$ can be shown by proving that for two interpretations that are related in a certain way the one is a model of $\mathsf{SM}(\mathsf{cnv}(F))$ if and only if the other is a model of $\mathsf{ans}(F)$: Let $I$ be a structure over $P$ and $Q$ as predicate symbols. Define $I'$ as the structure obtained from $I$ by replacing all atoms $p_i(A)$ with $\circ(p_i(A))$ and all atoms $q_i(A)$ with $\bullet(q_i(A))$. Define $I''$ as the structure that contains the same literals with predicate $\bullet$ as $I'$ and contains $+\circ(A)$ $(-\circ(A))$ whenever it contains $+\bullet(A)$ $(-\bullet(A))$. Thus the literals with predicate $\circ$ are chosen in $I''$ such that it satisfies $\mathsf{SYNC}$. The following statements are then equivalent:

$$\langle I, \beta\rangle \models \mathsf{SM}(\mathsf{cnv}(F)). \quad \text{(vii)}$$
$$\langle I, \beta\rangle \models \mathsf{project}(\mathsf{CIRC}[F[P,Q]; P; \emptyset] \land (P \leftrightarrow Q), \hat{P}). \quad \text{(viii)}$$
$$\langle I', \beta\rangle \models \mathsf{project}(\mathsf{CIRC}[F[\circ,\bullet]; \circ; \emptyset] \land \mathsf{SYNC}, \hat{\circ}). \quad \text{(ix)}$$
$$\langle I', \beta\rangle \models \mathsf{project}(\mathsf{CIRC}[F; \circ; \emptyset] \land \mathsf{SYNC}, \hat{\circ}). \quad \text{(x)}$$
$$\langle I'', \beta\rangle \models \mathsf{CIRC}[F; \circ; \emptyset] \land \mathsf{SYNC}. \quad \text{(xi)}$$
$$\langle I'', \beta\rangle \models \mathsf{circ\text{-}s}(F, \mathsf{POS} \cup \hat{\bullet}) \land \mathsf{SYNC}. \quad \text{(xii)}$$
$$\langle I'', \beta\rangle \models \mathsf{ans}(F). \quad \text{(xiii)}$$

## 8 Conclusion

We have introduced an operator raise which can be used to define circumscription in a compact way. The definition of that operator – in a semantic framework where structures are represented by sets of literals – is identical to that of literal projection, except that a set inclusion is replaced by a proper set inclusion.

An approach to an intuitive understanding of the raise operator is to consider minimization as passed through from the "object language level" (the extents of predicates is minimized) to the "meta level" of the semantic framework: Raise expresses that model agreement conditions are minimized. Accordingly, the predicate minimization conditions (commonly abbreviated by $P' < P$ in definitions of circumscription) have not to be made explicit with the raise operator, but are "built-in". In addition, the approach to "minimize model agreement conditions" effects that the raise operator straightforwardly covers certain generalizations of circumscription: Raise has – aside of a formula – just a set of literals as argument, such that, depending on the composition of this set, not only parallel circumscription with varied predicates can be expressed, but also predicate maximization conditions. Moreover, also minimization, maximization and agreement conditions can be expressed that apply only to a subset of the instances of a predicate.

The characterization of circumscription in terms of the raise operator is immediately useful to prove properties of circumscription in a streamlined way. The introduced semantic framework with the project and raise operators is a basis for future research, including the further elaboration of common and differing properties of both operators, the exploration of applications that involve combinations of circumscription and projection, and the investigation of possibilities for transferring and interleaving methods for non-monotonic reasoning, such as computation of stable models, with methods for second-order quantifier elimination and the closely related projection computation.

## References

1. P. Doherty, W. Łukaszewicz, and A. Szałas. Computing circumscription revisited: A reduction algorithm. *J. Autom. Reason.*, 18(3):297–338, 1997.
2. H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Einführung in die mathematische Logik*. Spektrum Akademischer Verlag, Heidelberg, 4th edition, 1996.
3. P. Ferraris. Answer sets for propositional theories. In *LPNMR'05*, pages 119–131, 2005.
4. P. Ferraris, J. Lee, and V. Lifschitz. A new perspective on stable models. In *IJCAI-07*, pages 372–379, 2007.
5. P. Ferraris, J. Lee, and V. Lifschitz. Stable models and circumscription. 2009. To appear; Draft retrieved on May 17th 2009 from https://www.cs.utexas.edu/users/otto/papers/smcirc.pdf.
6. D. M. Gabbay, R. A. Schmidt, and A. Szałas. *Second-Order Quantifier Elimination: Foundations, Computational Aspects and Applications*. CollegePublications, 2008.
7. M. Gelfond, H. Przymusinska, and T. Przymusinski. The extended closed world assumption and its relationship to parallel circumscription. In *ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pages 133–139, 1986.
8. J. Lang, P. Liberatore, and P. Marquis. Propositional independence – formula-variable independence and forgetting. *JAIR*, 18:391–443, 2003.
9. V. Lifschitz. Circumscription. In *Handbook of Logic in AI and Logic Programming*, volume 3, pages 298–352. Oxford University Press, 1994.
10. T. Przymusinski. An algorithm to compute circumscription. *Artificial Intelligence*, 83:59–73, 1989.
11. C. Wernhard. Literal projection for first-order logic. In *JELIA 08*, pages 389–402, 2008.
12. C. Wernhard. *Automated Deduction for Projection Elimination.* Number 324 in Dissertationen zur Künstlichen Intelligenz (DISKI). AKA/IOS Press, 2009.
13. C. Wernhard. Literal projection and circumscription. In *Int. Workshop on First-Order Theorem Proving, FTP'09.* University of Oslo, 2009.
14. C. Wernhard. Tableaux for projection computation and knowledge compilation. In *TABLEAUX 2009*, pages 325–340, 2009.

# Appendix

## A Proofs

In general, the proofs are presented formally by means of a table that shows numbered proof steps in symbolic form and is followed by verbal justifications. We use the notation described in Sect. 2.

### A.1 Proof of the Proposition in Section 3

**Proposition A1 (An Auxiliary Property of Structures).**

$$J \cap S \subseteq I \textit{ if and only if } I \cap \widetilde{S} \subseteq J.$$

*Proof.* Easy to verify from the definition of structure as a set of ground literals that contains for all ground atoms $A$ exactly one of $+A$ or $-A$. □

**Proposition 1 (Properties of $\mathsf{Raise}$).**

(1.i) *If* $F_1 \models F_2$, *then* $\mathsf{raise}(F_1, S) \models \mathsf{raise}(F_2, S)$.

(1.ii) *If* $F_1 \equiv F_2$, *then* $\mathsf{raise}(F_1, S) \equiv \mathsf{raise}(F_2, S)$.

(1.iii) $\mathsf{raise}(F_1 \lor F_2, S) \equiv \mathsf{raise}(F_1, S) \lor \mathsf{raise}(F_2, S)$.

(1.iv) $\mathsf{raise}(F_1 \land F_2, S) \models \mathsf{raise}(F_1, S) \land \mathsf{raise}(F_2, S)$.

(1.v) *If* $S = \widetilde{S}$, *then* $\mathsf{raise}(F, S) \equiv \bot$.

(1.vi) $\mathsf{raise}(\mathsf{project}(F, S), S) \equiv \mathsf{raise}(F, S)$.

(1.vii) $\mathsf{project}(\mathsf{raise}(F, S), S) \equiv \mathsf{raise}(F, S)$.

(1.viii) $\mathsf{project}(F, S) \equiv \mathsf{project}(F, S \cup \widetilde{S}) \lor \mathsf{raise}(F, S)$.

*Proof.*

(1.i) Straightforward from the definition of $\mathsf{raise}$.

(1.ii) Follows from Prop. 1.i.

(1.iii) Straightforward from the definition of $\mathsf{raise}$.

(1.iv) Follows from Prop. 1.i.

(1.v) If $S = \widetilde{S}$, then there do not exist structures $I, J$ such that $J \cap S \subset I \cap S$ is satisfied, which is a constraint in the definition of $\mathsf{raise}$.

(1.vi) Right-to-left follows from Prop. 1.i and Tab. 2.i. Left to right:

(1) $\langle I, \beta \rangle \models \mathsf{raise}(\mathsf{project}(F, S), S)$.
(2) There exists a $J$ such that:
$\langle J, \beta \rangle \models \mathsf{project}(F, S)$ and $J \cap S \subset I \cap S$.
(3) There exist $J, K$ such that:
$\langle K, \beta \rangle \models F$, $K \cap S \subseteq J$, and $J \cap S \subset I \cap S$.
(4) There exists a $K$ such that:
$\langle K, \beta \rangle \models F$ and $K \cap S \subset I \cap S$.
(5) $\langle I, \beta \rangle \models \mathsf{raise}(F, S)$.

Let $\langle I, \beta \rangle$ be an interpretation such that (1) holds. Step (4) follows from (3). The remaining steps follow from expanding/contracting definitions of $\mathsf{raise}$ and $\mathsf{project}$.

(1.vii) Right-to-left follows from Tab. 2.i. Left-to-right:

(1) $\langle I, \beta \rangle \models \mathsf{project}(\mathsf{raise}(F, S), S)$.
(2) There exists a $J$ such that:
$\langle J, \beta \rangle \models \mathsf{raise}(F, S)$ and $J \cap S \subseteq I$.
(3) There exist $J, K$ such that:
$\langle K, \beta \rangle \models F$, $K \cap S \subset J \cap S$, and $J \cap S \subseteq I$.
(4) There exists a $K$ such that:
$\langle K, \beta \rangle \models F$ and $K \cap S \subset I \cap S$.
(5) $\langle I, \beta \rangle \models \mathsf{raise}(F, S)$.

Let $\langle I, \beta \rangle$ be an interpretation such that (1) holds. Step (4) follows from (3). The remaining steps follow from expanding/contracting definitions of raise and project.

(1.viii)

(1) $\langle I, \beta \rangle \models \mathsf{project}(F, S)$ iff
(2) There exists a $J$ such that
$\langle J, \beta \rangle \models F$ and $J \cap S \subseteq I$ iff
(3) There exists a $J$ such that
$\langle J, \beta \rangle \models F$ and $J \cap S \subseteq I$ and $(I \cap S \subseteq J$ or $I \cap S \not\subseteq J)$ iff
(4) There exists a $J$ such that
$\langle J, \beta \rangle \models F$ and $J \cap S \subseteq I$ and $J \cap \widetilde{S} \subseteq I$
or
There exists a $J$ such that
$\langle J, \beta \rangle \models F$ and $J \cap S \subset I \cap S$ iff
(5) $\langle I, \beta \rangle \models \mathsf{project}(F, S \cup \widetilde{S}) \lor \mathsf{raise}(F, S)$.

Let $\langle I, \beta \rangle$ be an interpretation such that (1) holds. In showing equivalence of (4) to (3), Prop. A1 is used to justify that $I \cap S \subseteq J$ if and only if $J \cap \widetilde{S} \subseteq I$. The remaining equivalences are straightforward or follow from expanding/contracting definitions of raise and project.

□

## A.2 Proof of the Theorem in Section 4

**Theorem 1 (Second-Order Circumscription Expressed by circ-s).** *Let $F$ be a first-order formula and let $P, P', Z, Z'$ be tuples of predicate symbols as specified in the definition of* CIRC *(Def. 5). Let $Q$ be the set of predicate symbols in $F$ that are neither in $P$ nor in $Z$. Then*

$$\mathsf{CIRC\text{-}PROJ}[F; P; Z] \equiv \mathsf{circ\text{-}s}(F, (\hat{P} \cap \mathsf{POS}) \cup \hat{Q}).$$

*Proof.* Let $F, P, P', Q$ be as specified in the preconditions of the theorem. We formally show the following equivalence which, considering the definitions of CIRC and circ-s, straightforwardly entails the theorem:

$$\mathsf{project}(F' \land P' {<} P,\ \hat{P} \cup \hat{Q}) \ \equiv\ \mathsf{raise}(F, (\hat{P} \cap \mathsf{POS}) \cup \hat{Q}). \qquad \text{(xiv)}$$

We use the following additional symbolic notation: Let $S[R\backslash R']$ for a set of literals $S$ and tuples of predicate symbols $R, R'$ with equal length and without duplicate members stand for $S$ with each predicate symbol that occurs in $R$ at a certain position in the tuple replaced by the predicate symbol that is in $R'$ at that position.

The left-to-right direction of equivalence (xiv) can then be shown as follows:

(3) $\langle I, \beta\rangle \models \mathsf{project}(F' \land P'{<}P,\ \hat{P} \cup \hat{Q})$.
(4) $\langle J, \beta\rangle \models F'$.
(5) $\langle J, \beta\rangle \models P'{<}\,P$.
(6) $J \cap (\hat{P} \cup \hat{Q}) \subseteq I$.
(7) $(J \cap \hat{P'})[P'\backslash P] \cap \mathsf{POS} \subset J \cap \hat{P} \cap \mathsf{POS}$.
(8) $J \cap \hat{P} \subseteq I \cap \hat{P}$.
(9) $(J \cap \hat{P'})[P'\backslash P] \cap \mathsf{POS} \subset I \cap \hat{P} \cap \mathsf{POS}$.
(10) $K \stackrel{\text{def}}{=} (J - \hat{P}) \cup ((J \cap \hat{P'})[P'\backslash P])$.
(11) $\langle K, \beta\rangle \models F$.
(12) $K \cap \hat{Q} \subseteq I$.
(13) $K \cap \hat{P} \cap \mathsf{POS} \subset I \cap \hat{P} \cap \mathsf{POS}$.
(14) $K \cap ((\hat{P} \cap \mathsf{POS}) \cup \hat{Q}) \subset I \cap ((\hat{P} \cap \mathsf{POS}) \cup \hat{Q})$.
(15) $\langle I, \beta\rangle \models \mathsf{raise}(F, (\hat{P} \cap \mathsf{POS}) \cup \hat{Q})$.

Let $\langle I, \beta\rangle$ be an interpretation such that (3) holds. By the definition of project, then there exists a structure $J$ such that (4)–(6) hold. Step (7) is equivalent to (5). Step (8) follows from (6). Step (9) follows from (8) and (7). Let $K$ be as defined in (10). Step (11) then follows from (4); step (12) from (6); and step (13) from (9). Step (14) follows from (13) and (12). Finally, step (15) follows from (14) and (11) along with definition of raise. We now show the right-to-left direction of equivalence (xiv):

(16) $\langle I, \beta\rangle \models \mathsf{raise}(F, (\hat{P} \cap \mathsf{POS}) \cup \hat{Q})$.
(17) $\langle J, \beta\rangle \models F$.
(18) $J \cap ((\hat{P} \cap \mathsf{POS}) \cup \hat{Q}) \subset I \cap ((\hat{P} \cap \mathsf{POS}) \cup \hat{Q})$.
(19) $J \cap \hat{P} \cap \mathsf{POS} \subset I \cap \hat{P} \cap \mathsf{POS}$.
(20) $K \stackrel{\text{def}}{=} (J - (\hat{P} \cup \hat{P'})) \cup (I \cap \hat{P}) \cup ((J \cap \hat{P})[P\backslash P'])$.
(21) $K \cap \hat{P} = I \cap \hat{P}$.
(22) $\langle K, \beta\rangle \models F'$.
(23) $K \cap \hat{Q} \subseteq I$.
(24) $(K \cap \hat{P'})[P'\backslash P] \cap \mathsf{POS} \subset I \cap \hat{P} \cap \mathsf{POS}$.
(25) $(K \cap \hat{P'})[P'\backslash P] \cap \mathsf{POS} \subset K \cap \hat{P} \cap \mathsf{POS}$.
(26) $\langle K, \beta\rangle \models P'{<}\,P$.
(27) $K \cap (\hat{P} \cup \hat{Q}) \subseteq I$.
(28) $\langle I, \beta\rangle \models \mathsf{project}(F' \land P'{<}P,\ \hat{P} \cup \hat{Q})$.

Let $\langle I, \beta\rangle$ be an interpretation such that (16) holds. By the definition of raise, then there exists a structure $J$ such that (17) and (18) hold. Step (19) follows from (18), since $\hat{Q} = \widetilde{\hat{Q}}$. Let $K$ be as defined in (20). Then, step (21) is immediate;

step (22) follows from (17); step (23) from (18); and step (24) from (19). Step (25) follows from (24) and (21). Step (26) is equivalent to (25). Step (27) follows from (23) and (21). Finally, step (28) follows from (27) and (22) along with the definition of $\mathsf{project}$.

□

## A.3 Proofs of the Propositions and the Theorem in Section 6

**Proposition 2 (Circumscribing Projections).**

(i) $\mathsf{circ\text{-}s}(F, S) \models \mathsf{circ\text{-}s}(\mathsf{project}(F, S), S)$.

(ii) $\mathsf{circ\text{-}s}(\mathsf{project}(F, S), S) \models \mathsf{circ\text{-}s}(\mathsf{project}(F, S \cup \widetilde{S}), S)$.

*Proof.*

(2.i)

$$\begin{array}{lll}
(1) & \mathsf{circ\text{-}s}(F, S) & \equiv \\
(2) & F \land \lnot\mathsf{raise}(F, S) & \models \\
(3) & \mathsf{project}(F, S) \land \lnot\mathsf{raise}(\mathsf{project}(F, S), S) & \equiv \\
(4) & \mathsf{circ\text{-}s}(\mathsf{project}(F, S)) &
\end{array}$$

That step (2) entails (3) follows from Tab. 2.i and Prop. 1.vi. The equivalences follow from expanding/contracting the definition of $\mathsf{circ\text{-}s}$.

(2.ii)

$$\begin{array}{lll}
(1) & \mathsf{circ\text{-}s}(\mathsf{project}(F, S), S) & \equiv \\
(2) & \mathsf{project}(F, S) \land \lnot\mathsf{raise}(\mathsf{project}(F, S), S) & \equiv \\
(3) & (\mathsf{project}(F, S \cup \widetilde{S}) \lor \mathsf{raise}(F, S)) \land \lnot\mathsf{raise}(\mathsf{project}(F, S), S) & \models \\
(4) & (\mathsf{project}(F, S \cup \widetilde{S}) \lor \mathsf{raise}(\mathsf{project}(F, S \cup \widetilde{S}), S)) \land & \\
 & \lnot\mathsf{raise}(\mathsf{project}(F, S \cup \widetilde{S}), S) & \equiv \\
(5) & \mathsf{project}(F, S \cup \widetilde{S}) \land \lnot\mathsf{raise}(\mathsf{project}(F, S \cup \widetilde{S}), S) & \equiv \\
(6) & \mathsf{circ\text{-}s}(\mathsf{project}(F, S \cup \widetilde{S}), S). &
\end{array}$$

Equivalence of (3) to (2) follows from Prop. 1.viii. That (3) entails (4) follows from Tab. 2.iv and Prop. 1.i. Step (5) is logically equivalent to (4). The remaining equivalences follows from expanding/contracting the definition of $\mathsf{circ\text{-}s}$.

□

**Proposition 3 (Projecting Circumscriptions).** *If $F$ is well-founded with respect to $S$, then*

$$\mathsf{project}(\mathsf{circ\text{-}s}(F, S), S) \equiv \mathsf{project}(F, S).$$

*Proof.* Left-to-right follows from Tab. 2.ii. Right-to-left from Tab. 2.vi.

□

The proof of Theorem 2 is preceded by the following auxiliary propositions: Prop. A2 is used to prove Prop. A3, which is referenced in the proof of Theorem 2.ii. Prop. A4 is referenced in the proof of Theorem 2.iii.

**Proposition A2.** *If $F \models G$ then*

$$\mathsf{project}(F, S) \land \neg\mathsf{raise}(G, S) \models \mathsf{project}(F, S \cup \widetilde{S}).$$

*Proof.*

(1) $F \models G$.
(2) $\mathfrak{I} \models \mathsf{project}(F, S) \land \neg\mathsf{raise}(G, S)$.
(3) $\mathsf{raise}(F, S) \models \mathsf{raise}(G, S)$.
(4) $\mathfrak{I} \models (\mathsf{project}(F, S \cup \widetilde{S}) \lor \mathsf{raise}(F, S)) \land \neg\mathsf{raise}(G, S)$.
(5) $\mathfrak{I} \models \mathsf{project}(F, S \cup \widetilde{S})$.

Assume (1) and let $\mathfrak{I}$ be an interpretation such that (2) holds. Step (3) follows from (1) and Prop. 1.i. Step (4) follows from (2) and Prop. 1.viii. Step (5) follows from (4) and (3).

□

**Proposition A3.** *If $F \models G$ then*

$$\mathsf{project}(F, S) \land \mathsf{circ\text{-}s}(G, S) \models \mathsf{project}(F, S \cup \widetilde{S}).$$

*Proof.* Follows from Prop. A2, since by the definition of circ-s it holds that $\mathsf{circ\text{-}s}(G, S) \models \neg\mathsf{raise}(G, S)$.

□

**Proposition A4.**

$$\mathsf{project}(F, S \cup \widetilde{S}) \land \neg\mathsf{raise}(G, S) \models \mathsf{project}(F \land \neg\mathsf{raise}(G, S), (S \cap \widetilde{S})).$$

*Proof.*

(1) $\langle I, \beta \rangle \models \mathsf{project}(F, S \cup \widetilde{S})$.
(2) $\langle I, \beta \rangle \models \neg\mathsf{raise}(G, S)$.
(3) $\langle J, \beta \rangle \models F$.
(4) $J \cap (S \cup \widetilde{S}) \subseteq I$.
(5) $J \cap S = I \cap S$.
(6) For all $K$ such that $\langle K, \beta \rangle \models G$ and $K \cap S \subseteq I \cap S$ it holds that $K \cap S = I \cap S$.
(7) For all $K$ such that $\langle K, \beta \rangle \models G$ and $K \cap S \subseteq J \cap S$ it holds that $K \cap S = J \cap S$.
(8) $\langle J, \beta \rangle \models \neg\mathsf{raise}(G, S)$.
(9) $\langle I, \beta \rangle \models \mathsf{project}(F \land \neg\mathsf{raise}(G, S), (S \cap \widetilde{S}))$.

Let $\langle I, \beta \rangle$ be an interpretation such that (1) and (2) holds. By expanding the definition of project, from (1) follows that there exists a structure $J$ such that (3) and (4) hold. Step (5) follows from (4) and Prop. A1. By expanding the definition of raise, step (6) follows from (2). Step (7) – which is identical to (6), except that $I \cap S$ is replaced by $J \cap S$ – from (6) and (5). Step (8) follows from (7) and contracting the definition of raise. Step (9), finally, from (8), (4) and (3) and contracting the definition of project.

□

**Theorem 2 (Consequences of Circumscription).** *If $F$ is well-founded with respect to $S$, then*

$$\mathsf{circ\text{-}s}(F, S) \models G$$

*is equivalent to at least one of the following entailments, depending on additional preconditions on $G$:*

(i) $F \models G$, *if* $G \equiv \mathsf{project}(G, S)$*;*

(ii) $F \models \mathsf{project}(F \land G, S)$, *if* $G \equiv \mathsf{project}(G, S \cup \widetilde{S})$*;*

(iii) $F \models \mathsf{project}(F \land \neg\mathsf{project}(F \land \neg G, S), S)$.

*Proof.*

(2.i)

(1) $\mathsf{circ\text{-}s}(F, S) \models G$ iff
(2) $\mathsf{project}(\mathsf{circ\text{-}s}(F, S), S) \models G$ iff
(3) $\mathsf{project}(F, S) \models G$ iff
(4) $F \models G$.

Let $F, G$ and $S$ be as specified in the preconditions of the theorem. Then equivalence of (2) to (1) and of (4) to (3) follows from the precondition $G \equiv \mathsf{project}(G, S)$ and Tab. 2.vi. Equivalence of (3) to (2) follows from Prop. 3 and the precondition that $F$ is well-founded with respect to $S$.

(2.ii) Let $F, G$ and $S$ be as specified in the preconditions of the theorem. Left-to-right:

(1) $\mathsf{circ\text{-}s}(F, S) \models G$.
(2) $\mathsf{circ\text{-}s}(F, S) \models F \land G$.
(3) $\mathsf{circ\text{-}s}(F, S) \models \mathsf{project}(F \land G, S)$.
(4) $F \models \mathsf{project}(F \land G, S)$.

Assume (1). Step (2) follows from (1), since $\mathsf{circ\text{-}s}(F) \models F$. Step (3) follows from (2) and Tab. 2.i. Step (4) follows from (3) and Theorem 2.i, whose preconditions are met: That $F$ is well-founded with respect to $S$ is also here a precondition, and $\mathsf{project}(F \land G, S) \equiv \mathsf{project}(\mathsf{project}(F \land G, S), S)$ follows from Tab. 2.v.

Right-to-left:

(5) $F \models \mathsf{project}(F \land G, S)$.
(6) $\mathsf{circ\text{-}s}(F, S) \models \mathsf{project}(F \land G, S)$.
(7) $\mathsf{circ\text{-}s}(F, S) \models \mathsf{project}(F \land G, S \cup \widetilde{S}))$.
(8) $\mathsf{circ\text{-}s}(F, S) \models \mathsf{project}(G, S \cup \widetilde{S})$.
(9) $\mathsf{circ\text{-}s}(F, S) \models G$.

Assume (5). Step (6) follows from (5), since $\mathsf{circ\text{-}s}(F, S) \models F$. Step (7) follows from (5) and Prop. A3. Step (8) follows from (7) and Tab. 2.xx. Step (9) follows from (8) and the precondition $G \equiv \mathsf{project}(G, S \cup \widetilde{S})$.

(2.iii) Let $F, G, S$ as specified in the preconditions of the theorem. We formally prove the following equivalence:

$$\mathsf{circ\text{-}s}(F,S) \models G \quad \text{if and only if} \quad \mathsf{circ\text{-}s}(F,S) \models \neg\mathsf{project}(F \land \neg G, S). \qquad \text{(xv)}$$

The theorem straightforwardly follows from equivalence (xv) and Theorem 2.ii, whose preconditions are satisfied: That $F$ is well-founded with respect to $S$ is also here a precondition. The second precondition $\neg\mathsf{project}(F \land \neg G, S) \equiv \mathsf{project}(\neg\mathsf{project}(F \land \neg G, S), S \cup \widetilde{S})$ is an instance of the following general property which can be proven from the definition of $\mathsf{project}$: For all formulas $F$ it holds that

$$\mathsf{project}(\neg\mathsf{project}(F,S), S \cup \widetilde{S}) \equiv \neg\mathsf{project}(F,S). \qquad \text{(xvi)}$$

The left-to-right direction of equivalence (xv) can be shown as follows:

(1) $\mathsf{circ\text{-}s}(F,S) \models \neg\mathsf{project}(F \land \neg G, S)$.
(2) $\mathsf{circ\text{-}s}(F,S) \models \neg(F \land \neg G)$.
(3) $\mathsf{circ\text{-}s}(F,S) \models G$.

Assume (1). Step (2) follows from (1) and Tab. 2.i. Step (3) follows from (2) since $\mathsf{circ\text{-}s}(F,S) \models F$.

The right-to-left direction of equivalence (xv) can be shown as follows:

(1) $\mathsf{circ\text{-}s}(F,S) \models G$.
(2) $\mathsf{project}(\mathsf{circ\text{-}s}(F,S) \land \neg G, S \cup \widetilde{S}) \models \bot$.
(3) $\mathsf{project}(F \land \neg G \land \neg\mathsf{raise}(F,S), S \cup \widetilde{S}) \models \bot$.
(4) $\mathsf{project}(F \land \neg G, S \cup \widetilde{S}) \land \neg\mathsf{raise}(F,S) \models \bot$.
(5) $\mathsf{project}(F \land \neg G, S) \land \neg\mathsf{raise}(F \land \neg G, S) \land \neg\mathsf{raise}(F,S) \models \bot$.
(6) $\mathsf{project}(F \land \neg G, S) \land \neg\mathsf{raise}(F,S) \models \bot$.
(7) $\neg\mathsf{raise}(F,S) \models \neg\mathsf{project}(F \land \neg G, S)$.
(8) $\mathsf{circ\text{-}s}(F,S) \models \neg\mathsf{project}(F \land \neg G, S)$.

Assume (1). Step (2) follows from (1) and Tab. 2.x. Step (3) follows from (2) by expanding the definition of $\mathsf{circ\text{-}s}$. Step (4) follows from (3) and Prop. A4. Step (5) follows from (4) and Prop. 1.viii. Step (6) follows from (5) and Prop. 1.i. Step (7) is equivalent to (6). Finally, step (8) follows from (7) and the definition of $\mathsf{circ\text{-}s}$.

□

## B Reduct-Based Notions of Answer Sets

Answer sets according to the stable model semantics are traditionally described as fixed points, in terms of a *reduct* operation that maps a formula and an interpretation to a simpler formula. In this section, we give such a definition and show in Theorem B1 its equivalence to answer sets according to Def. 9. We then reconstruct in terms of this notion of *reduct* a specific variant of *reduct* from [3] which is used in [4, 5] to justify the “circumscription-like” characterization of stable models that we have compared in Sect. 7 to Def. 9. This appendix is then concluded by Theorem B2, which shows correctness of this reconstruction and thus gives insights into the relationships between the discussed characterizations of answer sets.

**Notation in this Appendix.** In this Appendix B we use notation and symbols as specified in Sects. 2 and 7, but with two constraining assumptions: (1.) Unless specially noted otherwise, the only predicate symbols in formulas and interpretations are $\circ$ and $\bullet$. (2.) We assume that formulas are *ground*. Since the variable assignment $\beta$ of an interpretation $\langle I, \beta\rangle$ is then irrelevant, we write the interpretation just as structure $I$. The material in this section should transfer straightforwardly to formulas with variables (and similarly, infinite conjunctions represented by clause sets), but we have not worked this out.

With Def. B2, we give a reduct-based characterization of answer sets according to the stable model semantics. This definition is preceded by Def. B1, which specifies the symbolic notation $F|_M$ for *restriction*, as we call the straightforward replacement of literals in a formula $F$ by truth value constants according to a consistent set of literals $M$. The definition of *reduct* (Def. B2.i) is then specified in terms of restriction. Theorem B1 then formally states the equivalence of the reduct-based definition of answer sets (Def. B2) with the circumscription-based (Def. 9).

**Definition B1 (Restriction $F|_L$, $F|_M$).** The *restriction* of a formula $F$ by a consistent set of literals $M$, in symbols $F|_M$, is $F$ with all literals that are members of $M$ replaced by $\top$, and all literals whose complement is a member of $M$ replaced by $\bot$.

The following properties of restriction will be used later on in proofs.

**Proposition B5 (Properties of Restriction $F|_L$, $F|_M$).**
(i) $I \models F|_M$ *if and only if* $I[M] \models F$.
(ii) $\neg(F|_M) = (\neg F)|_M$.
(iii) $F|_{I\cap S}$ *is satisfiable if and only if* $I \models \mathsf{project}(F, \widetilde{S})$.

*Proof.* (B5.i) and (B5.ii) are easy to see. (B5.iii) can be shown as follows:

(1) There exists a $J$ such that $J \models F|_{I\cap S}$ iff
(2) There exists a $J$ such that $J[I \cap S] \models F$ iff
(3) There exists a $J$ such that $J \models F$ and $I \cap S \subseteq J$ iff
(4) There exists a $J$ such that $J \models F$ and $J \cap \widetilde{S} \subseteq I$ iff
(5) $I \models \mathsf{project}(F, \widetilde{S})$.

Equivalence of (2) to (1) follows from Prop. B5.i. Equivalence of (4) to (3) from Prop. A1. Equivalence of (5) to (4) follows from contracting the definition of project.

□

**Definition B2 (Answer Set in Terms of Reduct).** For all consistent sets of ground literals $M \subseteq (\mathsf{NEG} \cap \hat{\circ})$ define:

(i) $\mathsf{reduct}(F, M, I) \overset{\text{def}}{=} F|_{I \cap (M \cup \hat{\bullet})}$.

(ii) $I \models \mathsf{ans}_{\mathsf{red}}(F, M)$ iff$_{\text{def}}$ $I \models \mathsf{circ\text{-}s}(\mathsf{reduct}(F, M, I), \mathsf{POS} \cup \hat{\bullet}) \land \mathsf{SYNC}$.

The $M$ parameter in Def. B2.i and B2.ii is not commonly found in characterizations of answer sets. Its value can be any set of negative literals with predicate $\circ$, for instance $\emptyset$ or $(\mathsf{NEG} \cap \hat{\circ})$. It is used to model different notions of *reduct* from the literature. Actually, as Theorem B1 shows, the value of $M$ is irrelevant for the correspondence between different characterizations of answer sets.

**Proposition B6 (Properties of Reduct).** *For all $M \subseteq (\mathsf{NEG} \cap \circ)$ it holds that*

(i) $I \models \mathsf{reduct}(F, M, J)$ *if and only if* $I[J \cap (M \cup \hat{\bullet})] \models F$.
(ii) $I \models \mathsf{reduct}(F, M, I)$ *if and only if* $I \models F$.
(iii) $\neg\mathsf{reduct}(F, M, I) = \mathsf{reduct}(\neg F, M, I)$.

*Proof.* (B6.i) follows from Prop. B5.i; (B6.ii) from Prop. B6.i; (B6.iii) from Prop. B5.ii. □

**Theorem B1 (Equivalence of $\mathsf{Ans}_{\mathsf{red}}$ and $\mathsf{Ans}$).** *For all $M \subseteq (\mathsf{NEG} \cap \hat{\circ})$ it holds that*

$$\mathsf{ans}_{\mathsf{red}}(F, M) \equiv \mathsf{ans}(F).$$

*Proof.*

(1) $I \models \mathsf{ans}_{\mathsf{red}}(F, M)$ iff
(2) $I \models \mathsf{circ\text{-}s}(\mathsf{reduct}(F, M, I), \mathsf{POS} \cup \hat{\bullet}) \land \mathsf{SYNC}$ iff
(3) $I \models \mathsf{reduct}(F, M, I) \land \neg\mathsf{raise}(\mathsf{reduct}(F, M, I), \mathsf{POS} \cup \hat{\bullet}) \land \mathsf{SYNC}$ iff
(4) $I \models F \land \neg\mathsf{raise}(\mathsf{reduct}(F, M, I), \mathsf{POS} \cup \hat{\bullet}) \land \mathsf{SYNC}$ iff
(5) $I \models F \land \neg\mathsf{raise}(F, \mathsf{POS} \cup \hat{\bullet}) \land \mathsf{SYNC}$ iff
(6) $I \models \mathsf{circ\text{-}s}(F, \mathsf{POS} \cup \hat{\bullet}) \land \mathsf{SYNC}$ iff
(7) $I \models \mathsf{ans}(F)$

Let $I$ be an interpretation such that (1) holds. Equivalence of (4) to (3) follows from Prop. B6.ii. The remaining equivalences are straightforward to see, with exception of equivalence of (5) to (4), which we show now:

(8) $\mathsf{raise}(\mathsf{reduct}(F, M, I), \mathsf{POS} \cup \hat{\bullet})$ iff
(9) There exists a $J$ such that
$J \models \mathsf{reduct}(F, M, I)$ and $J \cap (\mathsf{POS} \cup \hat{\bullet}) \subset I \cap (\mathsf{POS} \cup \hat{\bullet})$ iff
(10) There exists a $J$ such that
$J[I \cap (M \cup \hat{\bullet})] \models F$ and $J \cap (\mathsf{POS} \cup \hat{\bullet}) \subset I \cap (\mathsf{POS} \cup \hat{\bullet})$ iff
(11) There exists a $J$ such that
$J \models F$ and $J \cap (\mathsf{POS} \cup \hat{\bullet}) \subset I \cap (\mathsf{POS} \cup \hat{\bullet})$ iff
(12) $\mathsf{raise}(F, \mathsf{POS} \cup \hat{\bullet})$

Equivalence of (10) to (9) follows from Prop. B6.i. Equivalence of (11) to (10) can be shown as follows: By (10) it holds that $J \cap (\mathsf{POS} \cup \hat{\bullet}) \subseteq I$, which by Prop. A1 is equivalent to $I \cap (\mathsf{NEG} \cup \hat{\bullet}) \subseteq J$. Since $M \subseteq \mathsf{NEG}$ it then holds that $I \cap (M \cup \hat{\bullet}) \subseteq J$. Thus $J[I \cap (M \cup \hat{\bullet})] = J$. □

We now show the relationship of *reduct* as specified in Def. B2.i to the notion of *reduct* in [3], which is used in [4, 5] to justify the "circumscription-like" characterization of stable models that we have discussed in Sect. 7.

The *formulas* considered in [3] are based on classical propositional logic with a restricted syntax: Constructed from *atoms*, the truth value constant $\bot$, binary connectives $\wedge, \vee$, and implication $\rightarrow$. That is, negative literals, explicit negation, and the truth value constant $\top$ are omitted. Implication $(F \rightarrow G)$ has a special ambiguous meaning: On the one hand, it is semantically understood like classic implication, that is, as equivalent to $(\neg F \vee G)$. On the other hand it is used for two special purposes: As rule forming operator and to express negation as failure.

The correspondence of $\mathsf{reduct}$ to the notion of *reduct* of [3] is formally stated with Theorem B2 below. This correspondence is shown with two limitations: First, only logic programs as specified in Def. 8.iii, which are *clausal*, are considered as formulas. This class of includes disjunctive heads and negation as failure in heads. The work in [3–5] applies also to nonclausal formulas. The second limitation is that only reductions with respect to interpretations that are models of the original program are taken into account. This should not have any practical impact, since an interpretation that fails to model the original program cannot be an answer set anyways.

We now start working towards Theorem B2. The function $\mathsf{fp}$, defined in the following, specifies a mapping from logic programs to formulas as considered in [3]. Given $\mathsf{SYNC}$ and considering implications $(F \rightarrow G)$ as classically equivalent to $(\neg F \vee G)$, this function preserves equivalence, which is then stated as Prop. B7. The function $\mathsf{fe}$, on which $\mathsf{fp}$ is based, is not total, but defined as far as required to prove Theorem B2 and for the discussion in Sect. 7. In values of $\mathsf{fp}$ all atoms have $\circ$ as predicate symbol, none has $\bullet$. A formula that literally matches the syntax of [3] would be obtained from a value of $\mathsf{fp}$ by replacing all the atoms $\circ(A)$ with $A$, but for showing Theorem B2 this "unwrapping" is not required.

**Definition B3 (Translation of Logic Programs to Ferraris' Syntax).**

(i) The function $\mathsf{fe}$ maps a formula (constructed from literals with $\circ$ or $\bullet$ as predicate symbol, $\top$, $\bot$, $\wedge$, and $\vee$) to a formula corresponding to the syntax considered in [3] (constructed from positive literals with $\circ$ as predicate symbol, $\bot$, $\vee$, $\wedge$, and $\rightarrow$):

$$\begin{array}{rcll}
\mathsf{fe}(+\circ A) & \stackrel{\text{def}}{=} & +\circ A, & \text{where } A \text{ is a term,}\\
\mathsf{fe}(-\bullet A) & \stackrel{\text{def}}{=} & +\circ A \rightarrow \bot, & \text{where } A \text{ is a term,}\\
\mathsf{fe}(\top) & \stackrel{\text{def}}{=} & \bot \rightarrow \bot, & \\
\mathsf{fe}(\bot) & \stackrel{\text{def}}{=} & \bot, & \\
\mathsf{fe}(F \wedge G) & \stackrel{\text{def}}{=} & \mathsf{fe}(F) \wedge \mathsf{fe}(G), & \\
\mathsf{fe}(F \vee G) & \stackrel{\text{def}}{=} & \mathsf{fe}(F) \vee \mathsf{fe}(G). &
\end{array}$$

(ii) The function $\mathsf{fc}$ maps a rule clause to a formula corresponding to the syntax considered in [3] (as described in Def. B3.i):

$$\mathsf{fc}((\bigvee_{i=1}^{n} L) \vee H) \stackrel{\text{def}}{=} \mathsf{fe}(\bigwedge_{i=1}^{n} \widetilde{L}) \rightarrow \mathsf{fe}(H),$$

where $((\bigvee_{i=1}^{n} L) \vee H)$ is a rule clause with negated body $(\bigvee_{i=1}^{n} L)$ and head $H$ for some $n \geq 0$. (As special case, if $n = 0$, the negated body is $\bot$ and $\bigwedge_{i=1}^{n} \widetilde{L} = \top$. Similarly if the head $H$ contains no literals it is $\bot$.)

(iii) The function $\mathsf{fp}$ maps a logic program to a set of formulas which correspond to the syntax considered in [3] (as described in Def. B3.i):

$$\mathsf{fp}(\bigwedge_{i=1}^{n} F_i) \stackrel{\text{def}}{=} \{\mathsf{fc}(F_i) | i \in \{1, \ldots, n\}\}, \text{ where } n \geq 0.$$

**Proposition B7.** *If $F$ is a formula such that $\mathsf{fe}(F)$ is defined, then $(\mathsf{fe}(F) \wedge \mathsf{SYNC}) \equiv (F \wedge \mathsf{SYNC})$. If $F$ is a rule clause, then $(\mathsf{fc}(F) \wedge \mathsf{SYNC}) \equiv (F \wedge \mathsf{SYNC})$. If $F$ is logic program, then $(\bigwedge_{C \in \mathsf{fp}(F)} C \wedge \mathsf{SYNC}) \equiv (F \wedge \mathsf{SYNC})$.*

*Proof.* Easy to see from Def. B3.

□

The following Def. B4 directly reproduces the notion of reduct of [3]. Def. B4.i specifies an auxiliary mapping from sets of atoms which represent interpretations in [3] to interpretations as we represent them, which additionally takes into account that in values of $\mathsf{fp}$ each atom is "wrapped" with the $\circ$ predicate.

**Definition B4 (Ferraris' Reduct).**

(i) If $X$ is a set of ground atoms, then

$$\mathsf{interp}(X) \stackrel{\text{def}}{=} \{+\circ A | \circ A \in X\} \cup \{+\bullet A | \circ A \in X\} \cup \{-\circ A | \circ A \in \mathsf{ALL} - X\} \cup \{-\bullet A | \circ A \in \mathsf{ALL} - X\}.$$

(ii) The reduct $F^X$ of a formula $F$ relative to a set of ground atoms $X$ is the formula obtained from $F$ by replacing every outermost subformula $G$ such that $\mathsf{interp}(X) \not\models G$ with $\bot$.

(iii) The reduct $\Phi^X$ of a set of formulas $\Phi$ relative to a set of ground atoms $X$ is $\{F^X | F \in \Phi\}$.

The reduct $F^X$ depends on syntactic properties of $F$. That is, for classically equivalent $F$ and $G$, the reducts $F^X$ and $G^X$ are not necessarily classically equivalent. Theorem B2, which we are approaching, states that $F^X$ can be expressed in terms of our generic $\mathsf{reduct}$ (Def. B2.i) by instantiating $N$ with $(\mathsf{NEG} \cap \hat{\circ})$. Definition B5 gives this instantiation a name:

**Definition B5 ($\mathsf{Reduct}_{\mathsf{fer}}$).**

$$\mathsf{reduct}_{\mathsf{fer}}(F, I) \stackrel{\text{def}}{=} \mathsf{reduct}(F, \mathsf{NEG} \cap \hat{\circ}, I).$$

The statement of Theorem B2 is now preceded by a lemma to prove it, Prop. B9, which in turn is proven making use of the following lemma:

**Proposition B8 (Satisfiability of Reduct$_{\mathsf{fer}}$).** *If* $\mathcal{L}(F) \subseteq (\mathsf{POS} \cup \hat{\bullet})$*, then*

$$\mathsf{reduct}_{\mathsf{fer}}(F, I) \text{ is satisfiable if and only if } I \models F.$$

*Proof.* From Prop. B5.iii follows that $\mathsf{reduct}_{\mathsf{fer}}(F, I)$ is satisfiable if and only if $I \models \mathsf{project}(F, \mathsf{POS} \cup \hat{\bullet})$. If $\mathcal{L}(F) \subseteq (\mathsf{POS} \cup \hat{\bullet})$, then from properties of projection (Tab. 2.viii and 2.xv) it follows that $\mathsf{project}(F, \mathsf{POS} \cup \hat{\bullet}) \equiv F$.

□

**Proposition B9.** *If* $\mathcal{L}(F) \subseteq (\mathsf{POS} \cup \hat{\bullet})$*,* $\mathsf{fe}(F)$ *is defined, and* $X$ *is a set of atoms such that* $\mathsf{interp}(X) \models \mathsf{SYNC}$*, then*

$$\mathsf{fe}(F)^X \equiv \mathsf{reduct}_{\mathsf{fer}}(F, \mathsf{interp}(X)).$$

*Proof.* The proof is by induction on formulas. Let $I$ be the interpretation $\mathsf{interp}(X)$. If $I \not\models F$, by Prop. B7 it holds that $I \not\models \mathsf{fe}(F)$. Thus $\mathsf{fe}(F)^X = \bot$ and from Prop. B8 follows $\mathsf{reduct}_{\mathsf{fer}}(F, I) \equiv \bot$. Hence $\mathsf{fe}(F)^X = \bot \equiv \mathsf{reduct}_{\mathsf{fer}}(F, I)$. If $I \models F$, cases corresponding to the definition clauses of $\mathsf{fe}$ have to be distinguished. To verify the base cases, it is helpful to recall that the definition of $\mathsf{reduct}_{\mathsf{fer}}(F, I)$ expands as follows:

$$\mathsf{reduct}_{\mathsf{fer}}(F, I) = \mathsf{reduct}(F, \mathsf{NEG} \cap \hat{\circ}, I) = F|_{I \cap ((\mathsf{NEG} \cap \hat{\circ}) \cup \hat{\bullet})}. \qquad \text{(xvii)}$$

We have to consider the following cases:

- If $F = +\circ A$, $F = \top$, or $F = \bot$, then $\mathsf{fe}(F)^X \equiv F^X = F = \mathsf{reduct}_{\mathsf{fer}}(F, I)$.
- If $F = -\bullet A$, then $\mathsf{fe}(F)^X = (+\circ A \rightarrow \bot)^X = (\bot \rightarrow \bot) \equiv \top = \mathsf{reduct}_{\mathsf{fer}}(F, I)$.
- If $F = (F_1 \otimes F_2)$, where $\otimes$ is $\wedge$ or $\vee$, then $\mathsf{fe}(F)^X = (\mathsf{fe}(F_1)^X \otimes \mathsf{fe}(F_1)^X)$ and $\mathsf{reduct}_{\mathsf{fer}}(F, I) = (\mathsf{reduct}_{\mathsf{fer}}(F_1, I) \otimes \mathsf{reduct}_{\mathsf{fer}}(F_2, I))$. From the induction assumptions $\mathsf{fe}(F_1)^X \equiv \mathsf{reduct}_{\mathsf{fer}}(F_1, I)$ and $\mathsf{fe}(F_2)^X \equiv \mathsf{reduct}_{\mathsf{fer}}(F_2, I)$ then follows $\mathsf{fe}(F)^X \equiv \mathsf{reduct}_{\mathsf{fer}}(F, I)$.

□

**Theorem B2 (Ferraris' Reduct in Terms of the Generic Reduct).** *If* $X$ *is a set of atoms such that* $\mathsf{interp}(X) \models (F \wedge \mathsf{SYNC})$*, then*

$$\bigwedge_{C \in \mathsf{fp}(F)^X} C \;\equiv\; \mathsf{reduct}(F, \mathsf{NEG} \cap \hat{\circ}, \mathsf{interp}(X)).$$

*Proof.* Let $F$ and $X$ be as specified in the precondition of the theorem. Let $I$ be the interpretation $\mathsf{interp}(X)$. Recall that the right side of the theorem can be expressed more compact as $\mathsf{reduct}_{\mathsf{fer}}(F, I)$. From the definitions of $F^X$, $\mathsf{fp}$ and $\mathsf{reduct}$ it can be verified that for all rule clauses $C$ it holds that $\mathsf{fc}(C)^X$ is a set member in $\mathsf{fp}(F)^X$ if and only if $\mathsf{reduct}_{\mathsf{fer}}(C, I)$ is a conjunct in $\mathsf{reduct}_{\mathsf{fer}}(F, I)$.

The theorem thus follows if for all clauses $C$ in $F$ it holds that

$$\mathsf{fc}(C)^X \equiv \mathsf{reduct}_{\mathsf{fer}}(C, I). \tag{xviii}$$

We now show equivalence (xviii). Let $C = ((\bigvee_{i=1}^{n} L) \lor H)$ be a clause in $F$, with negated body $(\bigvee_{i=1}^{n} L)$ and head $H$ for some $n \geq 0$ in $F$. (As a special case, negated body and head can be empty, as described in Def. B3.ii). From the precondition $I \models F$ follows $I \models C$. With the precondition $I \models \mathsf{SYNC}$ and Prop. B7 then follows $I \models \mathsf{fc}(C)$. Thus $\mathsf{fc}(C)^X = (\mathsf{fc}(\bigvee_{i=1}^{n} L) \lor H)^X = (\mathsf{fe}(\bigwedge_{i=1}^{n} \widetilde{L}) \rightarrow \mathsf{fe}(H))^X = (\mathsf{fe}(\bigwedge_{i=1}^{n} \widetilde{L})^X \rightarrow \mathsf{fe}(H)^X) \equiv (\neg\mathsf{fe}(\bigwedge_{i=1}^{n} \widetilde{L})^X \lor \mathsf{fe}(H)^X)$. So equivalence (xviii) is implied by the following equivalence:

$$\neg\mathsf{fe}(\bigwedge_{i=1}^{n} \widetilde{L})^X \lor \mathsf{fe}(H)^X \equiv \mathsf{reduct}_{\mathsf{fer}}(\bigvee_{i=1}^{n} L \lor H, I), \tag{xix}$$

and, since $\mathsf{reduct}_{\mathsf{fer}}$ distributes over disjunction, also by the conjunction of the following two equivalences:

$$\neg\mathsf{fe}(\bigwedge_{i=1}^{n} \widetilde{L})^X \equiv \mathsf{reduct}_{\mathsf{fer}}(\bigvee_{i=1}^{n} L, I) \tag{xx}$$

$$\mathsf{fe}(H)^X \equiv \mathsf{reduct}_{\mathsf{fer}}(H, I). \tag{xxi}$$

Equivalence (xx) holds if and only if $\mathsf{fe}(\bigwedge_{i=1}^{n} \widetilde{L})^X \equiv \neg\mathsf{reduct}_{\mathsf{fer}}(\bigvee_{i=1}^{n} L, I)$, and thus, by Prop. B6.iii if and only if $\mathsf{fe}(\bigwedge_{i=1}^{n} \widetilde{L})^X \equiv \mathsf{reduct}_{\mathsf{fer}}(\bigwedge_{i=1}^{n} \widetilde{L}, I)$, which follows from Prop. B9. Equivalence (xxi) follows directly from Prop. B9.

□