Literal Projection and Circumscription

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. Background of the work 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. In this paper, an operator raise is introduced that is only slightly different from literal projection and can be used to define a generalization of circumscription in a straightforward and compact way. Some properties of this operator and of circumscription defined in terms of it, 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. A characterization of answer sets according to the stable model semantics in terms of circumscription is given. This characterization 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 specially interpreted connective.

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 [5]), including abduction, modularization of knowledge bases and the processing of circumscription. It is also closely related to knowledge compilation [13]. 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* [10, 11]. 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 [7] more precise and generalize it from propositional to first-order formulas. In the extended report version of this paper [12] we provide a thorough proof (Sect. 6).

– 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 [3, 4]. 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 as well as more details on the relationship of the introduced definition of stable models to other characterizations can be found in the extended report version of this paper [12].

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
A	– Atom
L	– Literal
S	– Set of ground literals (also called *literal scope*)
M	– Consistent set of ground literals
I, J, K	– Structure
β	– 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 $\land, \lor$ 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 $\forall x_1 \ldots \forall x_n (L_1 \lor \ldots \lor L_m)$, 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 TERMS and 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 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 [10] (see also Sect. 4 below). It will be further discussed in this introductory section (see [10, 11] for more thorough material). The semantics of the raise operator will be introduced later on in Sect. 3.

Interpretations. We use the notational variant of the framework of Herbrand interpretations described in [10]: 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 β is a *variable assignment*, that is, a mapping of the set of variables into 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 β are used: If F is a formula, then $F\beta$ denotes F with all variables replaced by their image in β; 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 β. 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_{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_{def}	$\langle I, \beta \rangle \not\models F$
$\langle I, \beta \rangle \models F_1 \land F_2$	iff_{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_{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_{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_{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_{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 β, 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) $\widetilde{+A} \stackrel{\text{def}}{=} -A$; $\widetilde{-A} \stackrel{\text{def}}{=} +A$. 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 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 $\wedge$ or $\vee$; $\mathcal{L}(\otimes x F) \stackrel{\text{def}}{=} \mathcal{L}(\otimes(F, S)) \stackrel{\text{def}}{=} \mathcal{L}(F)$ if $\otimes$ is a quantifier or the $\mathsf{project}$ or 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 [10, 11] 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.

Definition 1 (Essential Literal Base). The *essential literal base* of a formula F, in symbols $\mathcal{L}_{\mathcal{E}}(F)$, is defined as $\mathcal{L}_{\mathcal{E}}(F) \stackrel{\text{def}}{=} \{L \mid L \in \mathsf{ALL}$ 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 $\mathsf{project}$ shown in Tab. 1 [11]. The more involved proof of Tab. 2.xxi (and the related Tab. 3.v) can be found in [10, 11]. 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 [10, 11] for more discussion.[1]

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)	$\mathsf{project}(\forall x F, S) \models \forall x\ \mathsf{project}(F, S)$

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)

[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 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.

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).

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

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.

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 the same scope. Proposition 1.viii provides a characterization of *literal* projection in terms of raise and *atom* projection [10], 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 formula $\mathsf{circ}(F, S)$ that provides a characterization of circumscription in terms of raise, as we will first outline informally and then show more precisely.

Definition 4 (Circumscription).

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

In this notation, the *parallel circumscription of predicate constants P in sentence F with varied predicate constants Z* [8] is expressed as $\mathsf{circ}(F, S)$, where S 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 S 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 in $\mathsf{circ}(F, S)$ can be an arbitrary sets of literals, $\mathsf{circ}(F, S)$ 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 circ relates to the established definition of circumscription by means of second-order quantification [8, 1, 5]. 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 [8] and is straightforwardly equivalent to the sentence called *second-order circumscription of P in F with variable Z* in [1, 5]:

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 ≥ 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 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:

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

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 circ. Its proof in [12] formally relates second-order circumscription expressed by projection (Def. 6) with circumscription defined in terms of of the raise operator (Def. 4).

Theorem 1 (Second-Order Circumscription Expressed by circ). *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*

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

5 Well-Foundedness

As discussed in [8], 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 [8] 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

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

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 [8], 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 $\mathsf{circ}(F, S)$. 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 $\langle I, \beta\rangle \models \mathsf{project}(\mathsf{circ}(F, S), S)$.

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

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

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 $\mathsf{circ}(\mathsf{project}(F, S), S) \equiv \mathsf{circ}(F, S)$. From this equivalence, it can be derived that two formulas which express the same about positive literals (that is, have equivalent projections onto POS) have the same minimal models (that is, have equivalent circumscriptions for scope 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*

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

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 $\mathsf{project}(\mathsf{circ}(F, \mathsf{POS}), S) \equiv \mathsf{project}(F, S)$. 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, $\mathsf{circ}(F', \mathsf{POS}) \equiv \mathsf{circ}(F, \mathsf{POS})$. 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' [13].

Literal forgetting is a variant of literal projection that can be defined as $\mathsf{forget}(F, S) \stackrel{\text{def}}{=} \mathsf{project}(F, \overline{S})$ and is investigated for propositional logic in [7]. 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 [7], 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 [7] in the following four respects: (1.) We use weaker requirements on the entailed formulas by referring to the *essential* literal base instead of the (syntactic) literal base. The respective requirements on the syntactic literal base follow, since it is a subset of the essential literal base (see Sect. 2). (2.) We provide a thorough proof in [12]. The proof given in [7] just shows the characterizations as straightforward consequence of [9, Theorems 2.5 and 2.6], for which in turn no proof is given, neither in [9], nor in [6] which is referenced by [9]. (3.) We generalize the characterizations to first-order logic. (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 [8], 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*

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

is equivalent to (at least) one of the following statements, depending on $\mathcal{L}_{\mathcal{E}}(G)$:

(i) $F \models G$, *if* $\mathcal{L}_{\mathcal{E}}(G) \subseteq S$;

(ii) $F \models \mathsf{project}(F \land G, S)$, *if* $\mathcal{L}_{\mathcal{E}}(G) \subseteq (S \cup \widetilde{S})$;

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

7 Answer Sets with Stable Model Semantics

In [3, 4] 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

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

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 $(\bigvee_{i=1}^{m} -\circ A_i \lor \bigvee_{i=1}^{n} +\bullet B_i)$ is called its *negated body*, and the rule clause $(\bigvee_{i=1}^{o} +\circ C_i \lor \bigvee_{i=1}^{p} -\bullet D_i)$ is called its *head*.
(iii) A *logic program* is a conjunction of rule clauses.
(iv) The symbol 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 (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 $\bot$.

[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.

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 logic programs F:

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

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 [3, 4], and justify in this way that $\mathsf{ans}(F)$ indeed characterizes stable models. More detailed proofs and relationships to reduct based characterizations of answer sets can be found in [12]. We limit our considerations to logic programs according to Def. 8.iii, which are clausal sentences (the characterization in [3, 4] 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 $\circ$ 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 nonclassical operators used by [3, 4] (see [12] for an explicit such conversion). Let $\mathsf{SM}(\mathsf{cnv}(F))$ be the second-order sentence that characterizes the stable models of $\mathsf{cnv}(F)$ according to [3, 4]. The following equivalence can be verified, where $P' {<} P$ has the same meaning as in Def. 5:

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

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:

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

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

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

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 SYNC. The following statements are then equivalent:

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

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, J. Lee, and V. Lifschitz. A new perspective on stable models. In *IJCAI-07*, pages 372–379, 2007.
4. 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.
5. D. M. Gabbay, R. A. Schmidt, and A. Szałas. *Second-Order Quantifier Elimination: Foundations, Computational Aspects and Applications.* CollegePublications, 2008.
6. 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.
7. J. Lang, P. Liberatore, and P. Marquis. Propositional independence – formula-variable independence and forgetting. *JAIR*, 18:391–443, 2003.
8. V. Lifschitz. Circumscription. In *Handbook of Logic in AI and Logic Programming*, volume 3, pages 298–352. Oxford University Press, 1994.
9. T. Przymusinski. An algorithm to compute circumscription. *Artificial Intelligence*, 83:59–73, 1989.
10. C. Wernhard. Literal projection for first-order logic. In *JELIA 08*, pages 389–402, 2008.
11. C. Wernhard. *Automated Deduction for Projection Elimination.* Number 324 in Dissertationen zur Künstlichen Intelligenz (DISKI). AKA/IOS Press, 2009.
12. C. Wernhard. Literal projection and circumscription – extended version. Technical report, Technische Universität Dresden, 2009. Available from http://cs.christophwernhard.com/papers/projection-circumscription.pdf.
13. C. Wernhard. Tableaux for projection computation and knowledge compilation. In *TABLEAUX 2009*, pages 325–340, 2009.