Circumscription and Projection as Primitives of Logic Programming

Christoph Wernhard

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

Abstract. We pursue a representation of logic programs as classical first-order sentences. Different semantics for logic programs can then be expressed by the way in which they are wrapped into – semantically defined – operators for circumscription and projection. (Projection is a generalization of second-order quantification.) We demonstrate this for the stable model semantics, Clark's completion and a three-valued semantics based on the Fitting operator. To represent the latter, we utilize the polarity sensitiveness of projection, in contrast to second-order quantification, and a variant of circumscription that allows to express predicate minimization in parallel with maximization. In accord with the aim of an integrated view on different logic-based representation techniques, the material is worked out on the basis of first-order logic with a Herbrand semantics.

Table of Contents

1 Introduction

The multitude of semantics for logic programs is traditionally specified by a multitude of techniques: different rule languages, consequence operators, syntactic transformations like *reduct* and *completion*, and notions of model, two- and three-valued, for example. This makes it difficult to uncover relationships and transfer results between the semantics. It lets the long-term goal of a single logic-based system in which a variety of logic programming methods is simultaneously available appear quite fanciful. This work aims towards a unified and integrated view on different semantics for logic programs. We show a framework in which a logic program is represented by a classical first-order sentence, and several semantics for logic programs can be characterized by applying two further logic operators that are *defined in terms of classical semantics*: circumscription and projection.

A key observation is that semantics for logic programs involve circumscription in a way such that only certain *occurrences* of a predicate are affected, while others – basically those in the scope of negation as failure – stay unminimized. Indeed, as shown in [Lin91] and described in [Lif08], the stable models semantics can be characterized accordingly in terms of circumscription. From this point of view, the purpose of a rule syntax is just to indicate which occurrences are to be circumscribed. The alternative pursued here is to replace each "original" predicate by two replicas, one of them used in occurrences where circumscription should take effect. The formula then is classical, permitting for example simplifications that preserve classical equivalence.

Projection, a generalization of second-order quantification, can be used to control the interaction between the replicas. In general, projection is applied in the context of this work to express operations in a semantic way that are typically specified in syntactical terms, like systematic renaming of predicates and completion construction, where we refine a semantic characterization in [LL06].

We apply our framework to the stable model semantics, Clark's completion and a three-valued semantics based on the Fitting operator. The first two are distinguished just by the choice of circumscribed predicate occurrences, reflecting the characterization of Clark's completion in terms of stable models with negation as failure in the head described in [IS98]. The independence of syntactic constructions lets our framework quite naturally cover extensions of normal logic programs, including disjunctive heads and negation as failure in the head. In accord with the long-term goal of a unified logic-based knowledge processing system, the material in the paper is worked out for first-order logic with a Herbrand semantics, extended by circumscription and projection.

The paper is structured as follows: After notation and the used classical semantics have been specified in Sect. 2, projection and circumscription are introduced in Sect. 3. A view of logic programs as classical first-order sentences is described in Sect. 4. On this basis, it is shown in Sect. 5 how semantics for logic programs are expressed in terms of circumscription and projection. Specifically, the stable model semantics, Clark's completion, and a three-valued

semantics based on the Fitting operator are considered. In Sect. 6, the new characterization of the latter is related to the traditional definition, and a similar characterization of partial stable models is sketched. In the conclusion, further potential applications of this framework and a view on computational aspects are indicated.

Appendices provide technical material: Correctness of the reconstructions of the stable models semantics and Clark's completion is formally worked out in Appendix A and B, respectively. A further property of completion that is applied in the reconstruction of three-values semantics is shown in Appendix C. In Appendix D, a notational variant of the described framework that might facilitate its application to prove properties of semantics for logic programs is shown.

This report is an extended version of [Wer10a].

2 Notation and Preliminaries

Notation

We use the following symbols, also with sub- and superscripts, to stand for items of types as indicated in the following list (precise definitions of the types are given later on), considered implicitly as universally quantified in definition, proposition and theorem statements:

F, G, H – 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

We write the positive (negative, resp.) *literal* with atom A as $+A$ ($-A$, resp.). The *complement* of literal L is written $\overline{L}$. The *set of complements* of a set S of literals, that is, $\{\overline{L} | L \in S\}$, is written $\overline{S}$.

We call *predicate symbols* briefly *predicates*. We assume a fixed first-order signature with at least one constant. The sets of all ground terms, all ground literals, all positive ground literals, and all negative ground literals – with respect to this signature – are denoted by TERMS, ALL, POS, NEG, respectively. *Variables* are x, y, z, also with subscripts. The sequence $x_1, \ldots, x_n$, where n is the arity of predicate p, is abbreviated by $\overline{x}_p$.

Formulas

We assume that a *formula* is constructed from first-order literals and the logic operators shown in the left column of Tab. 1. That is, we consider formulas of first-order logic, extended by an operator for syntactic equality ($\doteq$) and the two operators project and raise, which will be discussed in Sect. 3. As meta-level

Table 1. The Satisfaction Relation

$\langle I,\beta\rangle \models L$	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}	$\langle I,\beta\rangle \not\models F$
$\langle I,\beta\rangle \models F_1 \wedge F_2$	iff_{def}	$\langle I,\beta\rangle \models F_1$ and $\langle I,\beta\rangle \models F_2$
$\langle I,\beta\rangle \models F_1 \vee F_2$	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 $\langle I,\beta\frac{t}{x}\rangle \models F$
$\langle I,\beta\rangle \models t_1 \doteq t_2$	iff_{def}	$t_1\beta = t_2\beta$
$\langle I,\beta\rangle \models \mathsf{project}_S(F)$	iff_{def}	there exists a J such that $\langle J,\beta\rangle \models F$ and $J \cap S \subseteq I$
$\langle I,\beta\rangle \models \mathsf{raise}_S(F)$	iff_{def}	there exists a J such that $\langle J,\beta\rangle \models F$ and $J \cap S \subset I \cap S$

notation with respect to this syntax, we use versions of the binary connectives with arbitrary integers ≥ 0 as arity, implication ($\rightarrow$), converse implication ($\leftarrow$), equivalence ($\leftrightarrow$), writing positive literals just as atoms, sequences of variables as quantifier arguments, and omitting of universal quantifiers. A *sentence* is a formula without free variables. A *clausal sentence* is a sentence $\forall x_1 \ldots x_n F$, where F is an conjunction with arbitrary arity of disjunctions (*clauses*) with arbitrary arity of literals.

Classical Semantics

We use a notational variant of the framework of Herbrand interpretations: An *interpretation* 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. Formula F with all free variables replaced by their image in β is denoted by $F\beta$; the variable assignment that maps x to ground term t and all other variables to the same values as β is denoted by $\beta\frac{t}{x}$.

As explicated in [Wer08], the structure component I of an interpretation $\langle I,\beta\rangle$ represents a *structure* in the conventional sense used in model theory, and, moreover, an interpretation represents a *second-order interpretation* [EFT84], if predicate variables are considered as distinguished predicates.

The satisfaction relation between interpretations 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. 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 $F_2 \models F_1$.

3 Projection, Literal Scopes and Circumscription

The project operator, defined semantically in Tab. 1, is applied in the context of this paper to provide *semantic* characterizations of operations and properties

that are typically defined in syntactic terms: Clark's completion, extracting the subformula with the "converse rules" from Clark's completion, systematic renaming of predicates, and independence of a formula from given predicates. The formula $\mathsf{project}_S(F)$ is called the *projection* of formula F onto literal scope S. The *forgetting* in F about S is a variant of projection, where the scope is considered complementary:

Definition 1 (Forgetting).

$$\mathsf{forget}_S(F) \stackrel{\text{def}}{=} \mathsf{project}_{\mathsf{ALL}-S}(F).$$

We call a set of ground literals in the role as argument to projection a *literal scope*. When specifying literal scopes, we let a set of predicates stand for the set of all ground instances of literals whose predicate is in the set.

As an intuitive special case of projection, consider a literal scope S that contains the same atoms in positive as well as negative literals. The condition $J \cap S \subseteq I$ in the definition of project is then equivalent to $J \cap S = I \cap S$, that is, structures I and J are required to be equal as far as members of S are considered, but unrelated otherwise. Projection is a generalization of second-order quantification: if S is the set of all ground literals with a predicate other than p, then $\mathsf{project}_S(F)$ (or equivalently $\mathsf{forget}_{\{\mathsf{p}\}}(F)$) can be expressed by the second-order formula $\exists \mathsf{p}\, F$.

Beyond second-order quantification, the condition $J \cap S \subseteq I$ in the definition of project encodes a different effect on literals depending on whether they are positive or negative (w.r.t. to formulas that do not contain $\neg$). Hence, this variant of projection is also termed more specifically *literal projection*. Consider for example,

$$\mathsf{forget}_{\{+\mathsf{q},-\mathsf{q}\}}((+\mathsf{p} \vee -\mathsf{q}) \wedge (+\mathsf{q} \vee -\mathsf{r})) \tag{i}$$

which is equivalent to

$$(+\mathsf{p} \vee -\mathsf{r}), \tag{ii}$$

and, in contrast,

$$\mathsf{forget}_{\{+\mathsf{q}\}}((+\mathsf{p} \vee -\mathsf{q}) \wedge (+\mathsf{q} \vee -\mathsf{r})), \tag{iii}$$

which is equivalent to

$$((+\mathsf{p} \vee -\mathsf{q}) \wedge (+\mathsf{p} \vee -\mathsf{r})), \tag{iv}$$

where $-\mathsf{q}$ is retained. In the context of this paper, these effects are applied to specify a three-valued semantics for logic programs. Further material on projection can be found in [Wer08]. The other "nonstandard" operator defined in Tab. 1 is raise, which we apply to define *scope-determined circumscription* [Wer10b], a generalization of predicate circumscription [McC80]:

Definition 2 (Scope-Determined Circumscription).

$$\mathsf{circ}_S(F) \stackrel{\text{def}}{=} F \wedge \neg \mathsf{raise}_S(F).$$

The argument S is also a literal scope, which then provides a uniform interface for expressions combining projection and circumscription. Superficially, raise is very similar to project: Consider Tab. 1. The condition $J \cap S \subseteq I$ in the definition of project is equivalent to $J \cap S \subseteq I \cap S$. Just by replacing the subset relation ($\subseteq$) with strict subset ($\subset$), the definition of raise is obtained. If F is a sentence over disjoint sets of predicates P, Q and Z, then the *parallel predicate circumscription of P in F with fixed Q and varied Z* [Lif94], traditionally written CIRC$[F; P; Z]$, is expressed as

$$\mathsf{circ}_{(P \cap \mathsf{POS}) \cup Q}(F). \tag{v}$$

Recall that in specifications of literal scopes, we let a set of predicates stand for the set of all ground instances of literals whose predicate is in the set. The scope $(P \cap \mathsf{POS}) \cup Q$ thus is the set of all *positive* ground literals with a circumscribed predicate, and *all* ground literals with a fixed predicate. While circumscription traditionally just allows to express predicate *minimization*, scope-determined circumscription symmetrically permits to express *maximization* by scopes containing just *negative* ground literals with predicates to be maximized. In the context of this paper, parallel minimization and maximization is applied to specify a three-valued semantics for logic programs.

4 Logic Programs as Classical Sentences

A logic program is typically understood as a set of rules of the form:

$$A_1 \mid \ldots \mid A_k \mid \mathsf{not} A_{k+1} \mid \ldots \mid \mathsf{not} A_l \;\leftarrow\; A_{l+1}, \ldots, A_m, \mathsf{not} A_{m+1}, \ldots, \mathsf{not} A_n. \tag{vi}$$

This involves logic operators which do not belong to classical first-order logic. To represent a logic program as a classical first-order sentence, we assume that the set of all predicates can be partitioned into *predicate groups*, that is, disjoint sets of equal cardinality. The idea is that each "original predicate" is replicated once in each group. The respective copy of the "original" p in predicate group P is then written p^P. If P and Q are two predicate groups, we say that p^P and p^Q are *corresponding* predicates, assuming that they have the same arity, which we also call *arity of p*. We transfer the notation p^P to atoms and literals: A^P (L^P) stands for an atom (literal) whose predicate is in predicate group P. Formally, the partitioning into predicate groups can be modeled by means of a total ordering $<_{\mathsf{pred}}$ on predicates such that p denotes the position of p^P within predicate group P sorted according to $<_{\mathsf{pred}}$. Corresponding predicates then have the same positions within their respective group. The set of all such positions p is written PREDS.

Definition 3 (Predicate Groups $\mathcal{C}, \mathcal{F}, \mathcal{O}$). The symbols $\mathcal{C}, \mathcal{F}, \mathcal{O}$ denote three different predicate groups.

Predicate groups $\mathcal{C}, \mathcal{F}, \mathcal{O}$ are used to express logic programs. Roughly, the group indicates whether a predicate occurrence should be *circumscribed* (group $\mathcal{C}$), should be *fixed* with respect to circumscription (group $\mathcal{F}$), or is yet *open* (group

$\mathcal{O}$), that is, further operations are applied that place it into group $\mathcal{C}$ or $\mathcal{F}$ at a later stage.

Definition 4 (Rule Clause, Raw Rule Clause).
(i) A *rule clause* is a clause of the form

$$+A_1^{\mathcal{C}} \vee \ldots \vee +A_k^{\mathcal{C}} \vee -A_{k+1}^{\mathcal{F}} \vee \ldots \vee -A_l^{\mathcal{F}} \vee -A_{l+1}^{\mathcal{C}} \vee \ldots \vee -A_m^{\mathcal{C}} \vee +A_{m+1}^{\mathcal{F}} \vee \ldots \vee +A_n^{\mathcal{F}},$$

where $n \geq m \geq l \geq k \geq 0$.
(ii) A *raw rule clause* is like a rule clause, except that the literals with indexes from $l+1$ to m are from predicate group $\mathcal{O}$ instead of $\mathcal{C}$.

Based on Def. 4, a logic program can be understood as a clausal sentence with rule clauses or raw rule clauses. In both cases, a logic program is then just a *classical first-order sentence* that meets certain restrictions. ([Raw] rule clauses can contain universal variables.) When we say that a [raw] rule clause *corresponds* to a rule of the form (vi), we assume that the [raw] rule clause has predicates from groups as indicated by matching (vi) with Def. 4.

The *head* of a [raw] rule clause is the disjunction of those of its literals whose index is less or equal to l, its *body* is the conjunction of the complements of its literals with index greater than l. A [raw] rule clause can express a *normal rule* (if $k = l = 1$), *integrity constraint* (if $k = l = 0$), *disjunctive rule* (if $k = l > 1$) and a *rule with negation as failure in the head* (if $l > k$). The class of rules in *general extended disjunctive programs (GEDP)* considered in [IS98] is however strictly more general: In rules of the form (vi), GEDP would allow also *negated* atoms in place of the atoms A_i, for $i \in \{1, \ldots, n\}$.

Predicate Renaming

Definition 6 below gives a semantic account of systematically replacing predicates from one group P by their correspondents from another group Q. First we define of shorthands for formulas that will be used at several places in the sequel.

Definition 5 (Predicate Inclusion). Let P, Q be predicate groups.
(i) $P \geq Q \stackrel{\text{def}}{=} \forall \overline{x} \bigwedge_{p \in \mathsf{PREDS}} (+p^P(\overline{x}_p) \vee -p^Q(\overline{x}_p))$.
(ii) $P = Q \stackrel{\text{def}}{=} P \geq Q \wedge Q \geq P$.

Definition 6 (Predicate Renaming in Terms of Projection). Let P, Q, R be predicate groups. Then

$$\mathsf{rename}_{P \mapsto Q}(F) \stackrel{\text{def}}{=} \mathsf{forget}_P(F \wedge P = Q).$$

The notation

$$\mathsf{rename}_{[P_1 \mapsto P_2, \ \ldots, \ P_{n-1} \mapsto P_n]}(F)$$

is a shorthand for $\mathsf{rename}_{P_{n-1} \mapsto P_n}(\ldots(\mathsf{rename}_{P_1 \mapsto P_2}(F))\ldots)$.

The formula $\mathsf{rename}_{P \mapsto Q}(F)$ is equivalent to F with all occurrences of predicates from P replaced by their respective corresponding predicates from Q.

5 Semantics for Logic Programs via Circumscription and Projection

Based on the representation of a logic program as a clausal first-order sentence with raw rule clauses, three well-known semantics for logic programs – the stable model semantics, the classical models of Clark's completion, and the three-valued minimal models obtained with the Fitting operator – can be characterized in terms of circumscription and projection:

Definition 7 (Semantics for Logic Programs Reconstructed). Let F be a formula over $\mathcal{C} \cup \mathcal{F} \cup \mathcal{O}$.

(i) $\mathsf{ans\text{-}stable}(F) \stackrel{\text{def}}{=} \mathsf{rename}_{\mathcal{F} \mapsto \mathcal{C}}(\mathsf{circ}_{(\mathcal{C} \cap \mathsf{POS}) \cup \mathcal{F}}(\mathsf{rename}_{\mathcal{O} \mapsto \mathcal{C}}(F)))$.

(ii) $\mathsf{ans\text{-}completion}(F) \stackrel{\text{def}}{=} \mathsf{rename}_{\mathcal{F} \mapsto \mathcal{C}}(\mathsf{circ}_{(\mathcal{C} \cap \mathsf{POS}) \cup \mathcal{F}}(\mathsf{rename}_{\mathcal{O} \mapsto \mathcal{F}}(F)))$.

(iii) $\mathsf{ans\text{-}fitting}(F) \stackrel{\text{def}}{=} \mathsf{circ}_{(\mathcal{C} \cap \mathsf{POS}) \cup (\mathcal{F} \cap \mathsf{NEG})}(\mathcal{F} \geq \mathcal{C} \wedge \mathsf{rename}_{\mathcal{O} \mapsto \mathcal{C}}(F) \wedge F^*)$,

where $F^* = \mathsf{rename}_{[\mathcal{C} \mapsto \mathcal{O},\ \mathcal{F} \mapsto \mathcal{C},\ \mathcal{O} \mapsto \mathcal{F}]}(\mathsf{forget}_{\mathcal{C} \cap \mathsf{POS}}(\mathsf{circ}_{(\mathcal{C} \cap \mathsf{POS}) \cup \mathcal{O} \cup \mathcal{F}}(F)))$.

The definientia are formulas of first-order logic extended with project (recall that rename is a shorthand for a formula with project) and circ as additional operators. For ans-stable and ans-completion, the involved projection could also be expressed as second-order quantification, as indicated in Sect. 3, and the involved scope-determined circumscription corresponds to parallel predicate circumscription of $\mathcal{C}$ with fixed $\mathcal{F}$. For ans-fitting, in contrast, proper generalizations of second-order quantification and parallel predicate circumscription are utilized: The scope $\mathcal{C} \cap \mathsf{POS}$ of the forgetting is just about *positive literals* with a predicate from $\mathcal{C}$. The scope $(\mathcal{C} \cap \mathsf{POS}) \cup (\mathcal{F} \cap \mathsf{NEG})$ of the outer circumscription expresses minimization of $\mathcal{C}$ in parallel with *maximization* of $\mathcal{F}$.

Semantics for logic programs are usually specified in terms of sets of atoms (*answer sets*), or "partial interpretations", that is, consistent sets of literals, representing a three-valued assignment of atoms to truth values: *true* (*false*) for the atoms of positive (negative) literals in the set, and *undefined* for the remaining atoms. In contrast, semantics for logic programs are specified in Def. 7 as classical models. For ans-stable, such a classical model $\langle I, \beta \rangle$ corresponds to the answer set $\{A \mid +A^{\mathcal{C}} \in I\}$. Predicates from $\mathcal{F}$ are not considered for the answer set, reflecting that $\mathcal{F}$ is forgotten by the outer rename. For ans-fitting, $\langle I, \beta \rangle$ corresponds to the partial interpretation

$$\{+A \mid +A^{\mathcal{C}} \in I\} \cup \{-A | -A^{\mathcal{F}} \in I\}. \tag{vii}$$

The characterization of stable models in terms of circumscription (Def. 7.i) originates from [Lin91] and is described as *"definition F"* in [Lif08] for logic programs over $\mathcal{C} \cup \mathcal{F}$ (in our notation). We use the third group $\mathcal{O}$ for mapping to other semantics. In [FLL07] a characterization of stable models in terms of a formula translation that is *similar* to predicate circumscription has been presented. Roughly, it differs from circumscription in that only certain *occurrences* of predicates are circumscribed. In this respect it is like the approach pursued

here. However, in [FLL07] these occurrences are identified by their syntactic position within formulas from a fragment of classical propositional logic – to the effect, that classically equivalent programs might not be equivalent when considered as logic programs. A formal proof of the correspondence of ans-stable to the original characterization of stable models [GL88] and variants of it is shown in Appendix A.

Equivalence of ans-completion to the syntactically defined Clark's completion [Cla78] is shown in Appendix B along the approach of [LL06], but generalized to first-order logic and refined by utilizing predicate groups: Head literals are distinguished by placing them in $\mathcal{C}$, which allows to prove *equivalence* of semantic and syntactic characterizations, whereas the related Proposition 4 in [LL06] just makes the weaker statement that the semantically defined completion of F_1 is equivalent to the syntactically defined Clark's completion of F_2 for *some* F_2 *that is equivalent to* F_1.

The formula F in Def. 7 is over $\mathcal{C} \cup \mathcal{F} \cup \mathcal{O}$. In ans-stable and ans-completion, it is subjected to renaming the predicates from $\mathcal{O}$ to either $\mathcal{C}$ or $\mathcal{F}$, respectively, which is actually the only difference between these semantics. For F that are just over $\mathcal{C} \cup \mathcal{F}$ both semantics are identical. The characterization of Clark's completion in terms of stable models of programs with negation as failure in the head, described by means of a program transformation in [IS98], thus can be rendered by the following equivalence:

$$\text{If } F \text{ is over } \mathcal{C} \cup \mathcal{F} \cup \mathcal{O}, \text{ then} \qquad \text{(viii)}$$
$$\mathsf{ans\text{-}completion}(F) \equiv \mathsf{ans\text{-}stable}(\mathsf{rename}_{\mathcal{O} \mapsto \mathcal{F}}(F)).$$

Based on a fixed-point characterization of the models of Clark's completion as so-called *supported models* [ABW88], it has been shown in [MS92] that a stable model of a *normal* logic program (i.e. with rules of the form (vi) where $k = l = 1$) is also a minimal model of its Clark completion. For more general classes of logic programs, analogous properties can be proven on the basis of Def. 7 (Prop. D2): If F is over $\mathcal{C} \cup \mathcal{F} \cup \mathcal{O}$ and $F \equiv \mathsf{forget}_{\mathcal{O} \cap \mathsf{POS}}(F)$, then

$$\mathsf{ans\text{-}stable}(F) \models \mathsf{ans\text{-}completion}(F), \qquad \text{(ix)}$$

and, if in addition, $F \equiv \mathsf{forget}_{\mathcal{F} \cap \mathsf{NEG}}(F)$, then

$$\mathsf{ans\text{-}stable}(F) \models \mathsf{circ}_{\mathcal{C} \cap \mathsf{POS}}(\mathsf{ans\text{-}completion}(F)). \qquad \text{(x)}$$

6 Three-Valued Semantics Based on the Fitting Operator

In [Fit85] a consequence operator Φ (*Fitting operator*) is introduced which is applied to construct three-valued interpretations M, represented by consistent sets of ground literals. For a ground program F with rules of the form (vi), constrained by $k = l = 1$ (i.e. normal rules), the value of the Fitting operator can be described as follows: The body of a rule is *true* with respect to M, if and only if each of its literals is contained in M. It is *false* with respect to M if and

only if the complement of at least one of its literals is in M. For a given M, the Fitting operator yields the union of (1.) the set of all positive literals $+A$ such that there exists a rule of F with head $+A$ whose body is true with respect to M, and (2.) the set of all negative literals $-A$ such all rules of F with head $+A$ have a body that is false with respect to M. The minimal fixed point (minimal w.r.t. set inclusion of the consistent literal sets M) of the Fitting operator then represents a (partial) model, the result of the program, and thus might be called "answer set" according to "Fitting's semantics".

To show that ans-fitting (Def. 7.iii) corresponds to this semantics, we reconstruct it in our framework. We use interpretations over the union of the two predicate groups $\mathcal{C}$ and $\mathcal{F}$ to represent the consistent literal sets expressing three-valued or interpretations. Structures I such that

$$\langle I, \beta \rangle \models \mathcal{F} \geq \mathcal{C} \tag{xi}$$

(assignment β is irrelevant for (xi) since $\mathcal{F} \geq \mathcal{C}$ does not contain free variables) are mapped with the following one-to-one correspondence to such literal sets M:

Definition 8 (Representation of Three-Valued Interpretations).

(i) $\mathsf{litset}(I) \stackrel{\text{def}}{=} \{+A \mid +A^{\mathcal{C}} \in I\} \cup \{-A \mid -A^{\mathcal{F}} \in I\}$.

(ii) $\mathsf{litset}^{-1}(M) \stackrel{\text{def}}{=} \{+A^{\mathcal{C}} \mid +A \in M\} \cup \{-A^{\mathcal{C}} \mid +A \notin M\} \cup \{+A^{\mathcal{F}} \mid -A \notin M\} \cup \{-A^{\mathcal{F}} \mid -A \in M\}$.

Minimization with respect to set inclusion of the literal sets M can be expressed by scope-determined circumscription onto the scope

$$(\mathcal{C} \cap \mathsf{POS}) \cup (\mathcal{F} \cap \mathsf{NEG}), \tag{xii}$$

since $\mathsf{litset}(I) \subseteq \mathsf{litset}(J)$ if and only if $I \cap ((\mathcal{C} \cap \mathsf{POS}) \cup (\mathcal{F} \cap \mathsf{NEG})) \subseteq J$. Circumscribing onto this scope effects that predicates from $\mathcal{C}$ are minimized, and, in parallel, predicates from $\mathcal{F}$ are *maximized*, which can not be directly expressed by conventional predicate circumscription.

The Fitting operator is – like the original form of Clark's completion – applied to normal logic programs, that is, sets of rules of the form (vi) where $k = l = 1$. Such a program corresponds to a clausal sentence with rule clauses that are over $\mathcal{F}$ except for a single positive literal over $\mathcal{C}$. For Clark's completion, in a first "preprocessing" step, such a sentence is transformed to an equivalent, possibly nonclausal, sentence of a second particular form, which is then the basis for the proper completion transformation. A suitable such second form will be specified in Def. 9 below. We call it *normal completion input sentence*, since any clausal sentence with rule clauses constrained by $k = l = 1$ is equivalent to such a sentence, obtainable by straightforward rewriting with equivalences, including

$$+p(t_1,\ldots,t_n) \vee G \equiv \forall x_1 \ldots x_n \, +p(x_1,\ldots,x_n) \vee \neg x_1 \doteq t_1 \vee \ldots \vee \neg x_n \doteq t_n \vee G, \tag{xiii}$$

where $x_1, \ldots, x_n$ are variables not occurring in $t_1, \ldots, t_n, G$.

Definition 9 (Normal Completion Input Sentence). A sentence F is called a *normal completion input sentence* if it is over $\mathcal{C} \cup R$, with R being a set of predicates not in $\mathcal{C}$, and is of the form

$$\forall \overline{x} \, (\bigwedge_{p \in \mathsf{PREDS}} (+p^{\mathcal{C}}(\overline{x}_p) \vee G_p(\overline{x}_p))),$$

where
(1.) $\overline{x}$ is $x_1, \ldots, x_k$, with k being the maximal arity of all members of PREDS,
(2.) $G_p(\overline{x}_p)$ are formulas whose free variables are in $\overline{x}_p$, and
(3.) $G_p(\overline{x}_p)$ does not contain predicates from $\mathcal{C}$.

In traditional terminology, a subformula $+p^{\mathcal{C}}(\overline{x}_p)$ of a normal completion input sentence corresponds to a head, and $G_p(\overline{x}_p)$ to the negated disjunction of all bodies of clauses with head $+p^{\mathcal{C}}(\overline{x}_p)$. For a normal completion input sentence F over $\mathcal{C} \cup \mathcal{F}$, Clark's completion of F can then be defined as $\mathsf{rename}_{\mathcal{F} \mapsto \mathcal{C}}(F \wedge F^*)$, where F^* is the *syntactic completion addendum* of F, defined as follows:

Definition 10 (Syntactic Completion Addendum). Let F be a normal completion input sentence with syntactic constituents as specified in Def. 9. The following sentence is called the *syntactic completion addendum* of F:

$$\forall \overline{x} \, (\bigwedge_{p \in \mathsf{PREDS}} (-p^{\mathcal{C}}(\overline{x}_p) \vee \neg G_p(\overline{x}_p))).$$

Let F be a normal completion input sentence over $\mathcal{C} \cup \mathcal{F} \cup \mathcal{O}$. Recall the definition of ans-fitting (Def. 7.iii):

$$\mathsf{ans\text{-}fitting}(F) \stackrel{\text{def}}{=} \mathsf{circ}_{(\mathcal{C} \cap \mathsf{POS}) \cup (\mathcal{F} \cap \mathsf{NEG})}(\mathcal{F} \geq \mathcal{C} \wedge \mathsf{rename}_{\mathcal{O} \mapsto \mathcal{C}}(F) \wedge F^*),$$

where

$$F^* = \mathsf{rename}_{[\mathcal{C} \mapsto \mathcal{O},\ \mathcal{F} \mapsto \mathcal{C},\ \mathcal{O} \mapsto \mathcal{F}]}(\mathsf{forget}_{\mathcal{C} \cap \mathsf{POS}}(\mathsf{circ}_{(\mathcal{C} \cap \mathsf{POS}) \cup \mathcal{O} \cup \mathcal{F}}(F))).$$

The outer circumscription has the scope specified above in (xii) and thus effects minimization to the smallest models with respect to the three-valued view of interpretations. The argument formula of this circumscription consists of three conjuncts. The first one is (xi) which excludes interpretations without a consistent three-valued correspondence. The other ones correspond to the positive and negative consequences, respectively, of the Fitting operator.

Assume that the normal completion input sentence F has been obtained in a "preprocessing" step, as outlined above, from an equivalent clausal sentence with raw clauses, representing a conjunction F_0 of rules of the form (vi), constrained by $k = l = 1$, and such that all heads have just mutually distinct variables as argument terms (in presence of equivalence (xiii), the last condition is w.l.o.g.). Let p be some member of PREDS. The formula $G_p(\overline{x}_p)$ is then a constituent of F as specified in Def. 9. The second conjunct $\mathsf{rename}_{\mathcal{O} \mapsto \mathcal{C}}(F)$ is the original logic program, with $\mathcal{O}$ renamed to $\mathcal{C}$, as in ans-stable. It can be shown that $\langle I, \beta \rangle \models$

$\mathcal{F} \geq \mathcal{C} \wedge \mathsf{rename}_{\mathcal{O}\mapsto\mathcal{C}}(\neg G_p(\overline{x}_p))$ if and only if there is a rule R with head predicate p in F_0 such that the body of its ground instance $R\beta$ is *true* with respect to $\mathsf{litset}(I)$. The subformulas $(+p^{\mathcal{C}}(\overline{x}_p) \vee \mathsf{rename}_{\mathcal{O}\mapsto\mathcal{C}}(\neg G_p(\overline{x}_p)))$ in $\mathsf{rename}_{\mathcal{O}\mapsto\mathcal{C}}(F)$ then allow to infer positive literals with predicate $p^{\mathcal{C}}$, corresponding to positive consequences of the Fitting operator.

Analogously, $\langle I, \beta\rangle \models \mathcal{F} \geq \mathcal{C} \wedge \mathsf{rename}_{[\mathcal{F}\mapsto\mathcal{C},\ \mathcal{O}\mapsto\mathcal{F}]}(G_p(\overline{x}_p))$ if and only if for all rules R with head predicate p in F_0 it holds that the body of the ground instance $R\beta$ is *false* with respect to $\mathsf{litset}(I)$. Subformulas of the form $(-p^{\mathcal{F}}(\overline{x}_p) \vee \mathsf{rename}_{[\mathcal{F}\mapsto\mathcal{C},\ \mathcal{O}\mapsto\mathcal{F}]}(G_p(\overline{x}_p)))$ then allow to infer negative literals with predicate $p^{\mathcal{F}}$, corresponding to negative consequences of the Fitting operator. Sentence F^* is the universally quantified conjunction of these subformulas, for each predicate p from PREDS. It is equivalent to the syntactic completion addendum of F (Def. 10), subjected to switching group assignments $\mathcal{C}$ and $\mathcal{F}$, and renaming $\mathcal{O}$ to $\mathcal{F}$. This switching and renaming is expressed by rename applied to $[\mathcal{C} \mapsto \mathcal{O},\ \mathcal{F} \mapsto \mathcal{C},\ \mathcal{O} \mapsto \mathcal{F}]$. The circumscription in F^* is equivalent to the conjunction of F and its syntactic completion addendum, which follows from Theorems B1 and B2 in Appendix B. The forgetting about $\mathcal{C} \cap \mathsf{POS}$ serves to extract an equivalent to the syntactic completion addendum from this circumscription, as shown in detail in Theorem C1 in Appendix C. *Literal* projection is utilized there to preserve the negative literals from $\mathcal{C}$ in the addendum, but forget about the positive literals from $\mathcal{C}$ in the original formula, and with them the whole original formula.

A further prominent semantics for logic programs with three-valued models is the *partial stable model semantics*. In [JNS+06] a characterization of partial stable models as stable models of a translated program is given (tracing back to earlier work [Sch95]). Based on a reconstruction of the syntactic transformation Tr(P) of [JNS+06] in terms of rename, and on the characterization of stable models by $\mathsf{ans\text{-}stable}$, partial stable models can be characterized in our framework as shown in the following definition. The three-valued (i.e. partial) interpretations are represented there in the same way as shown above for the Fitting semantics.

Definition 11 (Partial Stable Models Reconstructed). Let F be a formula over $\mathcal{C} \cup \mathcal{F} \cup \mathcal{O}$. Let $\mathcal{C}'$ and $\mathcal{F}'$ be two additional predicate groups, different from each other and from $\mathcal{C}, \mathcal{F}, \mathcal{O}$.

$$\mathsf{ans\text{-}partial\text{-}stable}(F) \stackrel{\text{def}}{=} \mathsf{rename}_{[\mathcal{C}'\mapsto\mathcal{C},\mathcal{F}'\mapsto\mathcal{F}]}(\mathsf{circ}_{((\mathcal{C}\cup\mathcal{F})\cap\mathsf{POS})\cup\mathcal{C}'\cup\mathcal{F}'}(\mathcal{F}\geq\mathcal{C}\wedge F_1\wedge F_2)),$$

where $F_1 = \mathsf{rename}_{[\mathcal{O}\mapsto\mathcal{C},\mathcal{F}\mapsto\mathcal{F}']}(F)$ and $F_2 = \mathsf{rename}_{[\mathcal{O}\mapsto\mathcal{C},\mathcal{F}\mapsto\mathcal{C}',\mathcal{C}\mapsto\mathcal{F}]}(F)$.

7 Conclusion

We investigated a representation of logic programs as classical first-order sentences that are wrapped into the semantically defined additional operators circumscription and projection, in different ways, rendering different established semantics of logic programs. The generality of our framework indicates interesting spaces that have yet to be explored: Our characterizations of semantics for

logic programs apply to broad formula classes. The scopes of circumscription and projection in the characterizations of semantics could be modified, or additional applications of projection could be merged in, to express, for example, models that are "stable only with respect to some atoms", and to restrict answer sets to atoms that are relevant for the user [EW08,GKS09].

A computational approach to the processing of operators for circumscription and projection is "elimination", analogous to second-order quantifier elimination: Computing for a given formula that involves the operator (second-order quantifier, resp.) an equivalent formula without the operator (second-order quantifier, resp.). Indeed, methods for the computation of circumscription and projection can essentially be considered as methods for second-order quantifier elimination [GSS08,Wer08,Wer09]. Our framework thus indicates that methods for processing logic programs could be seen in this context: On one hand, established methods for second-order quantifier elimination might be applied to process logic programs, which might be especially interesting for nonground programs. On the other hand, known efficient techniques for processing logic programs with specific semantics get embedded in a wider context when seen as particular efficient second-order quantifier elimination methods for constrained inputs.

Acknowledgements. I am obliged to anonymous referees of an earlier version for suggestions to improve the presentation and bringing important related works [MS92,Sch95,JNS+06] to attention.

References

[ABW88] K. R. Apt, H. A. Blair, and A. Walker. Towards a theory of declarative knowledge. In Jack Minker, editor, *Foundations of deductive databases and logic programming*, pages 89–148. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[Cla78] Keith L. Clark. Negation as failure. In Herve Gallaire and Jack Minker, editors, *Logic and Databases*, pages 292–322. Plenum Press, New York, 1978.

[EFT84] H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer, New York, 1984.

[EW08] Thomas Eiter and Kewen Wang. Semantic forgetting in answer set programming. *Artificial Intelligence*, 172:1644–1672, 2008.

[Fit85] Melvin Fitting. A Kripke-Kleene semantics for logic programs. *Journal of Logic Programming*, 2(4):295–312, 1985.

[FLL07] Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz. A new perspective on stable models. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI-07*, pages 372–379, 2007.

[GKS09] Martin Gebser, Benjamin Kaufmann, and Torsten Schaub. Solution enumeration for projected Boolean search problems. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 6th International Conference, CPAIOR 2009*, volume 5547 of *LNCS*, pages 71–86. Springer, 2009.

[GL88] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *Logic Programming, Proceedings of the Fifth International Conference and Symposium, ICLP/SLP 1988*, pages 1070–1080. MIT Press, 1988.

[GSS08] D. M. Gabbay, R. A. Schmidt, and A. Szałas. *Second-Order Quantifier Elimination: Foundations, Computational Aspects and Applications*. College Publications, London, 2008.

[IS98] Katsumi Inoue and Chiaki Sakama. Negation as failure in the head. *Journal of Logic Programming*, 35(1):39–78, 1998.

[JNS+06] Tomi Janhunen, Ilkka Niemelä, Dietmar Seipel, Patrik Simons, and Jia-Huai You. Unfolding partiality and disjunctions in stable model semantics. *ACM Transactions on Computational Logic*, 7(1):1–37, 2006.

[Lif94] Vladimir Lifschitz. Circumscription. In *Handbook of Logic in AI and Logic Programming*, volume 3, pages 298–352. Oxford University Press, Oxford, 1994.

[Lif08] Vladimir Lifschitz. Twelve definitions of a stable model. In *Logic Programming: 24th International Conference, ICLP 2008*, volume 5366 of *LNCS*, pages 37–51. Springer, 2008.

[Lin91] Fangzhen Lin. *A Study of Nonmonotonic Reasoning*. PhD thesis, Stanford University, 1991.

[LL06] Johyung Lee and Fangzhen Lin. Loop formulas for circumscription. *Artificial Intelligence*, 170:160–185, 2006.

[McC80] John McCarthy. Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[MS92] W. Markek and V. S. Subrahmanian. The relationship between stable, supported, default and autoepistemic semantics for general logic programs. *Theoretical Computer Science*, 103:365–386, 1992.

[Sch95] John S. Schlipf. The expressive powers of the logic programming semantics. *Journal of Computer and System Sciences*, 51(1):64–86, 1995.

[Wer08] Christoph Wernhard. Literal projection for first-order logic. In *Logics in Artificial Intelligence: 11th European Conference, JELIA 08*, volume 5293 of *LNAI*, pages 389–402. Springer, 2008.

[Wer09] Christoph Wernhard. Tableaux for projection computation and knowledge compilation. In *Automated Reasoning with Analytic Tableaux and Related Methods: International Conference, TABLEAUX 2009*, volume 5607 of *LNAI*, pages 325–340. Springer, 2009.

[Wer10a] Christoph Wernhard. Circumscription and projection as primitives of logic programming. In *Technical Communications of the International Conference on Logic Programming, ICLP'10*, LIPIcs, 2010. To appear.

[Wer10b] Christoph Wernhard. Literal projection and circumscription. In *Proceedings of the 7th International Workshop on First-Order Theorem Proving, FTP'09*, volume 556 of *CEUR Workshop Proceedings*, 2010. Available from http://ceur-ws.org/Vol-556.

Appendix

A Correctness of the Reconstruction of Stable Models

In this appendix, we formally show the correspondence of the reconstruction of the stable models semantics in terms of circumscription and projection (Def. 7.i) with the traditional definition in terms of *reduct*. This correspondence is stated as Theorem A1, the main result of this appendix.

We consider logic programs represented by formulas over $\mathcal{C} \cup \mathcal{F}$, which includes clausal sentences with rule clauses – and thus normal logic programs with various generalizations, as described in Sect. 4. For these formulas, we define an operator ans-stable-via-reduct (Def. A2) that models the original definition of stable models [GL88], abstracted and generalized in some respects. The equivalence of ans-stable to ans-stable-via-reduct, which justifies that ans-stable indeed represents the stable models semantics, is then stated as Theorem A1. First we specify additional symbolic notation:

Definition A1 (Modified Structure).

$$I[M] \stackrel{\text{def}}{=} (I - \overline{M}) \cup M.$$

Definition A2 (Reduct, Stable Model in Terms of Reduct). Let F be a formula over $\mathcal{C} \cup \mathcal{F}$ and assume $M \subseteq (\mathsf{NEG} \cap \mathcal{C})$.

(i) $\langle I, \beta\rangle \models \mathsf{reduct}_{M,J}(F)$ iff$_{\text{def}}$ $\langle I[J \cap (\mathcal{F} \cup M)], \beta\rangle \models F$.

(ii) $\langle I, \beta\rangle \models \mathsf{min\text{-}reduct}_{M}(F)$ iff$_{\text{def}}$ $\langle I, \beta\rangle \models \mathsf{circ}_{\mathcal{C}\cap\mathsf{POS}}(\mathsf{reduct}_{M,I}(F)) \wedge \mathcal{F}{=}\mathcal{C}$.

(iii) $\mathsf{ans\text{-}stable\text{-}via\text{-}reduct}_{M}(F) \stackrel{\text{def}}{=} \mathsf{forget}_{\mathcal{F}}(\mathsf{min\text{-}reduct}_{M}(F))$.

For F over $\mathcal{C} \cup \mathcal{F}$, the outset of Def. A2 is the same as of the definition of ans-stable: A logic program is a formula over the two predicate groups $\mathcal{C}$ and $\mathcal{F}$, where $\mathcal{F}$ identifies occurrences of predicates that are subject to negation as failure. Accordingly, interpretations range over these two predicate groups, but only their portions that concern group $\mathcal{C}$ are used to identify the finally resulting stable models.

The definition of ans-stable-via-reduct proceeds in three steps Def. A2.i–A2.iii. Definition A2.i models the notion of *reduct* of a logic program F with respect to a given interpretation J, which has been specified originally by a symbolic subscript notation [GL88]. The traditional notion of *reduct* involves deletion of clauses and literals from F that are subject to negation as failure, in correspondence to the given interpretation J. Our characterization in Definition A2.i does not make this transformation explicit, but just expresses an easily verifiable semantic property of it. By avoiding the syntactic transformation, we can also avoid to take into account the traditional view of first-order programs as *infinite sets* of propositional clauses. So to speak, we extend our syntax not by some infinite construct, but by the reduct operator, defined semantically in Def. A2.i.

There is an extra parameter M in Definition A2.i, whose value can be any set of negative literals with predicates from $\mathcal{C}$, for instance $\emptyset$ or $(\mathsf{NEG} \cap \mathcal{C})$.

Actually, as can be seen from Theorem A1, the value of M does not affect the meaning of ans-stable-via-reduct. The M parameter just opens up the possibility to include into determining the reduct also the propagation of certain values from J into literals of F that are *not subject to negation as failure*, and thus model corresponding notions of reduct from the literature. The original definition [GL88] corresponds to $M = \emptyset$.

The following proposition shows two properties of reduct, referenced in the proof of Theorem A1. Proposition A1.i expresses a semantic counterpart of the fact that the syntactically defined reduct in the traditional sense contains no literals that are subject to negation as failure: The reduct is equivalent to its projection onto $\mathcal{C}$.

Proposition A1 (Properties of Reduct). *If F is over $\mathcal{C} \cup \mathcal{F}$, and $M \subseteq (\mathsf{NEG} \cap \mathcal{C})$, then*

(i) $\mathsf{reduct}_{M,I}(F) \equiv \mathsf{project}_{\mathcal{C}}(\mathsf{reduct}_{M,I}(F))$;

(ii) $\langle I, \beta\rangle \models \mathsf{reduct}_{M,I}(F)$ *iff* $\langle I, \beta\rangle \models F$.

Definition A2.ii renders the fixed point property (I on both sides of the definition) and the minimality property of stable models, which is expressed as circumscribing the predicates from $\mathcal{C}$. Predicates from $\mathcal{F}$ are varied there. Justified by Prop. A1.i, it would be equivalently possible to circumscribe there with fixed $\mathcal{F}$, as explicated in the proof of Theorem A1 (equivalence of steps (7) and (6), p. 17). The condition $\mathcal{F} = \mathcal{C}$ selects only interpretations which assign the same meaning to predicates from $\mathcal{C}$ and their correspondents from $\mathcal{F}$, reflecting that in the traditional view predicates propagated into negated literals at reduct construction are not distinguished. In the third step Def. A2.iii, predicate group $\mathcal{F}$ which is irrelevant for the final answer sets is removed by forgetting.

Theorem A1 (Correctness of Ans-Stable). *If F is over $\mathcal{C} \cup \mathcal{F}$, and $M \subseteq (\mathsf{NEG} \cap \mathcal{C})$, then*

$$\mathsf{ans\text{-}stable}(F) \equiv \mathsf{ans\text{-}stable\text{-}via\text{-}reduct}_M(F).$$

Before we come to the proof of Theorem A1, we introduce the concepts bipols and unipols that are used in the proof, and state a lemma about circumscription as proposition:

Proposition A2 (Equivalent Circumscription Scopes). *If $S_p \subseteq S_c$ and $\mathsf{uniscope}(S_p) = \mathsf{uniscope}(S_c)$, then*

$$\mathsf{circ}_{S_p}(\mathsf{project}_{S_p \cup \overline{S_p}}(F)) \equiv \mathsf{circ}_{S_c}(\mathsf{project}_{S_p \cup \overline{S_p}}(F)).$$

Proof (Proof of Theorem A1 – Correctness of Ans-Stable*).* Define the shorthands: $F_1 \stackrel{\text{def}}{=} \mathsf{circ}_{(\mathcal{C} \cap \mathsf{POS}) \cup \mathcal{F}}(F)$ and $F_2 \stackrel{\text{def}}{=} \mathsf{circ}_{\mathcal{C} \cap \mathsf{POS}}(\mathsf{reduct}_{M,I}(F))$. Let $\langle J, \beta\rangle$ be an interpretation. By expanding ans-stable, rename and forget, it holds that $\langle J, \beta\rangle \models \mathsf{ans\text{-}stable}(F)$ if and only if

There exists an I such that $\langle I, \beta\rangle \models F_1 \wedge \mathcal{F}{=}\mathcal{C}$ and $I \cap \mathcal{C} = J \cap \mathcal{C}$. (xiv)

Let M be a subset of $\mathsf{NEG} \cap \mathcal{C}$. By expanding ans-stable-via-reduct, min-reduct, and forget it holds that $\langle J, \beta\rangle \models \mathsf{ans\text{-}stable\text{-}via\text{-}reduct}_M(F)$ if and only if

$$\text{There exists an } I \text{ such that } \langle I, \beta\rangle \models F_2 \wedge \mathcal{F}{=}\mathcal{C} \text{ and } I \cap \mathcal{C} = J \cap \mathcal{C}. \qquad \text{(xv)}$$

Since the only difference between (xiv) and (xv) is F_1 versus F_2, the equivalence of $\mathsf{ans\text{-}stable}(F)$ to $\mathsf{ans\text{-}stable\text{-}via\text{-}reduct}_M(F)$ follows if

$$\text{For all } F \text{ over } \mathcal{C} \cup \mathcal{F} \text{ and } \langle I, \beta\rangle : \ \langle I, \beta\rangle \models F_1 \text{ if and only if } \langle I, \beta\rangle \models F_2. \quad \text{(xvi)}$$

Statement (xvi) is derived as shown below in tabular form. Let F be a formula over $\mathcal{C} \cup \mathcal{F}$, let $\langle I, \beta\rangle$ be an interpretation and let M be a subset of $\mathsf{NEG} \cap \mathcal{C}$.

(1) $\langle I, \beta\rangle \models \mathsf{circ}_{(\mathcal{C}\cap\mathsf{POS})\cup\mathcal{F}}(F)$
(2) iff $\langle I, \beta\rangle \models F$ and there does not exist a J such that:
$\langle J, \beta\rangle \models F$ and $J \cap ((\mathcal{C} \cap \mathsf{POS}) \cup \mathcal{F}) \subset I \cap ((\mathcal{C} \cap \mathsf{POS}) \cup \mathcal{F})$
(3) iff $\langle I, \beta\rangle \models F$ and there does not exist a J such that:
$\langle J[I \cap (\mathcal{F} \cup M)], \beta\rangle \models F$ and $J \cap ((\mathcal{C} \cap \mathsf{POS}) \cup \mathcal{F}) \subset I \cap ((\mathcal{C} \cap \mathsf{POS}) \cup \mathcal{F})$
(4) iff $\langle I, \beta\rangle \models F \wedge \neg\mathsf{raise}_{(\mathcal{C}\cap\mathsf{POS})\cup\mathcal{F}}(\mathsf{reduct}_{M,I}(F))$
(5) iff $\langle I, \beta\rangle \models \mathsf{reduct}_{M,I}(F) \wedge \neg\mathsf{raise}_{(\mathcal{C}\cap\mathsf{POS})\cup\mathcal{F}}(\mathsf{reduct}_{M,I}(F))$
(6) iff $\langle I, \beta\rangle \models \mathsf{circ}_{(\mathcal{C}\cap\mathsf{POS})\cup\mathcal{F}}(\mathsf{reduct}_{M,I}(F))$
(7) iff $\langle I, \beta\rangle \models \mathsf{circ}_{\mathcal{C}\cap\mathsf{POS}}(\mathsf{reduct}_{M,I}(F))$.

Step (2) is obtained from (1) by expanding circ and raise. Equivalence of (3) to (2) can be shown as follows: $(J \cap ((\mathcal{C} \cap \mathsf{POS}) \cup \mathcal{F})) \subset (I \cap ((\mathcal{C} \cap \mathsf{POS}) \cup \mathcal{F}))$ implies $(J \cap ((\mathcal{C} \cap \mathsf{POS}) \cup \mathcal{F})) \subseteq I$, which is equivalent to $(I \cap (\mathsf{NEG} \cup \mathcal{F})) \subseteq J$ (since in general $(J \cap S) \subseteq I$ if and only if $(I \cap \overline{S}) \subseteq J$). Because $M \subseteq \mathsf{NEG}$, this implies $(I \cap (M \cup \mathcal{F})) \subseteq J$, and thus also $J = J[I \cap (M \cup \mathcal{F})]$. Step (4) is obtained from (3) by contracting reduct and raise. Equivalence of (5) to (4) follows from Prop. A1.ii. Step (6) is obtained from (5) by contracting circ. Equivalence of (7) to (6) follows from Prop. A2 whose parameters S_p, S_c, G (to avoid confusion with F in our proof, the proposition parameter is renamed to G here) are instantiated as follows: $S_p = \mathcal{C} \cap \mathsf{POS}$, $S_c = (\mathcal{C} \cap \mathsf{POS}) \cup \mathcal{F}$, and $G = \mathsf{reduct}_{M,I}(F)$. Thus $S_p \cup \overline{S_p} = \mathcal{C}$. From Prop. A1.i follows that $\mathsf{reduct}_{M,I}(F)$ is equivalent to $\mathsf{project}_{\mathcal{C}}(\mathsf{reduct}_{M,I}(F))$, which then matches $\mathsf{project}_{S_p\cup\overline{S_p}}(G)$ in Prop. A2. □

B Correctness of the Reconstruction of Completion

In this appendix we formally show the correspondence of the reconstruction of Clark's completion in terms of circumscription and projection (Def. 7.ii) to the traditional definition as a syntactic formula transformation [Cla78]. As already indicated in Sect. 5, we proceed similarly to [LL06] but with some differences which will be discussed at the end of this appendix.

The technical material in this appendix takes the definitions of *normal completion input sentence* (Def. 9) and *syntactic completion addendum* (Def. 10) from Sect. 6 as a starting point. The notion of *normal completion input sentence* is first extended such that also logic programs with integrity constraints can be

taken into account (Def. B1). On this basis, completion is characterized in two ways, syntactically, directly rendering Clark's completion, and semantically, in terms of projection (Defs. B2 and B3, resp.). Equivalence of both characterizations is then stated as Theorem B1. Theorem B2 states the equivalence of the semantic characterization to a third characterization of completion in terms of circumscription. Theorem B3 then combines the equivalences of the three characterizations to the statement that $\mathsf{ans\text{-}completion}(F)$ (Def. 7.ii) is equivalent to the syntactic completion of F.

In contrast to Sect. 6, we consider here not just normal logic programs, but the more general class of normal logic programs *with constraints*, that is, with rules of the form (vi) where $k = l \leq 1$. They correspond to a clausal sentence with rule clauses that are over $\mathcal{F}$ except possibly for a single positive literal over $\mathcal{C}$. The following definition of *completion input sentence* is the straightforward generalization of *normal completion input sentence* (Def. 9) by permitting also constraints, clauses with $k = l = 0$:

Definition B1 (Completion Input Sentence and Formula). A sentence F is called a *completion input sentence* if it is over $\mathcal{C} \cup R$, with R being a set of predicates not in $\mathcal{C}$, and is of the form $\forall \overline{x}\, F'(\overline{x})$, such that

$$F'(\overline{x}) = (\bigwedge_{p \in \mathsf{PREDS}} (+p^{\mathcal{C}}(\overline{x}_p) \vee G_p(\overline{x}_p))) \wedge H,$$

where
(1.) $\overline{x}$ is $x_1, \ldots, x_k$, with k being the maximal arity of all members of PREDS,
(2.) $G_p(\overline{x}_p)$ are formulas whose free variables are in $\overline{x}_p$,
(3.) H has no free variables, and
(4.) $G_p(\overline{x}_p)$ and H do not contain predicates from $\mathcal{C}$.

The formula $F'(\overline{x})$ is called the *completion input formula* of F.

In the traditional terminology of logic programming, the subformulas of a completion input sentence correspond to heads and bodies as indicated in the sequel to Def. 9. In addition, the subformula H corresponds to the conjunction of all integrity constraints of F. The following definition of *syntactic completion* straightforwardly renders the syntactically defined original definition of Clark's completion [Cla78]:

Definition B2 (Syntactic Completion). Let F be a completion input sentence with syntactic constituents as specified in Def. B1. The *syntactic completion addendum* of F is exactly as defined for *normal* completion input sentences F in Def. 10 (the constituent H which corresponds to the integrity constraints of F is irrelevant for the syntactic completion addendum).

The *syntactic completion* of a completion input sentence F is the formula

$$\mathsf{rename}_{\mathcal{F} \mapsto \mathcal{C}}(F \wedge F^*),$$

where F^* is the syntactic completion addendum of F.

Axioms for syntactic equality included in the original formulation of Clark's completion [Cla78] have no correspondents in Def. B2, since we consider syntactic equality ($\doteq$) as an operator that is "built-in" into the logic (Tab. 1). The following definition specifies a notion of completion semantically in terms of projection. Its equivalence to syntactic completion is then stated as Theorem B1.

Definition B3 (Semantic Completion). Let F be a completion input sentence with completion input formula $F'(\overline{x})$. The following sentence is called the *semantic completion addendum* of F:

$$\bigwedge_{p \in \mathsf{PREDS}} (\forall \overline{x}\ {-p^{\mathcal{C}}(\overline{x}_p)} \vee \neg \mathsf{forget}_{p^{\mathcal{C}}}(F'(\overline{x}) \wedge {-p^{\mathcal{C}}(\overline{x}_p)})).$$

The *semantic completion* of a completion input sentence F is

$$\mathsf{rename}_{\mathcal{F} \mapsto \mathcal{C}}(F \wedge F^*),$$

where F^* is the semantic completion addendum of F.

Theorem B1 (Equivalence of Semantic and Syntactic Completion). *If F is a completion input sentence then*

(i) *The conjunction of F and its semantic completion addendum is equivalent to the conjunction of F and its syntactic completion addendum.*

(ii) *The semantic completion of F is equivalent to the syntactic completion of F.*

Proof. Theorem B1.ii follows immediately from Theorem B1.i and the definitions of *semantic completion* and *syntactic completion*. Theorem B1.i can be shown as follows: Let F be a completion input sentence with syntactic constituents as specified in Def. B1. Let p be a member of PREDS. The theorem follows from the following equivalence, which we are going to show:

$$F \wedge \forall \overline{x}\ ({-p^{\mathcal{C}}(\overline{x})} \vee \neg G_p(\overline{x})) \equiv F \wedge \forall \overline{x}\ ({-p^{\mathcal{C}}(\overline{x})} \vee \neg \mathsf{forget}_{p^{\mathcal{C}}}(F'(\overline{x}) \wedge {-p^{\mathcal{C}}(\overline{x})})). \quad \text{(xvii)}$$

Let $H'(\overline{x}) \stackrel{\text{def}}{=} (\bigwedge_{q \in \mathsf{PREDS}-\{p\}} (+q^{\mathcal{C}}(\overline{x}_p) \vee G_q(\overline{x}_q))) \wedge H$. The following properties of H' follow immediately from the definition of F':

(1) $F'(\overline{x}) \equiv (+p^{\mathcal{C}}(\overline{x}) \vee G_p(\overline{x})) \wedge H'(\overline{x})$.
(2) $p^{\mathcal{C}}$ does not occur in $H'(\overline{x})$.
(3) $F \models \forall \overline{x}\ H'(\overline{x})$.

Since $p^{\mathcal{C}}$ does also not occur in $G_p(\overline{x})$, from (1) and (2), along with properties of projection, e.g. [Wer08, Theorem 4], it follows that

(4) $\mathsf{forget}_{p^{\mathcal{C}}}(F'(\overline{x}) \wedge {-p^{\mathcal{C}}(\overline{x})})$
(5) $\equiv \mathsf{forget}_{p^{\mathcal{C}}}((+p^{\mathcal{C}}(\overline{x}) \vee G_p(\overline{x})) \wedge H'(\overline{x}) \wedge {-p^{\mathcal{C}}(\overline{x})})$
(6) $\equiv \mathsf{forget}_{p^{\mathcal{C}}}(G_p(\overline{x}) \wedge H'(\overline{x}) \wedge {-p^{\mathcal{C}}(\overline{x})})$
(7) $\equiv G_p(\overline{x}) \wedge H'(\overline{x})$.

Along with (3), the equivalence of (7) to (4) entails (xvii). □

The following Theorem B2 essentially states equivalence of completion defined in terms of projection (Def. B3) to completion defined in terms of circumscription (ans-completion defined in Def. 7.ii). The renaming of $\mathcal{F}$ to $\mathcal{C}$ involved in completion is omitted in this theorem, such that it can also be applied in different contexts, for example to justify the characterization of semantics based on the Fitting operator as discussed Sect. 6.

Theorem B2 (Completion Lemma). *If F is a completion input sentence over $\mathcal{C} \cup R$, with R being a set of predicates not in $\mathcal{C}$, then the conjunction of F and its semantic completion addendum is equivalent to*

$$\mathsf{circ}_{(\mathcal{C} \cap \mathsf{POS}) \cup R}(F).$$

To prove Theorem B2 we need the two auxiliary propositions B1 and B2 stated below, preceding the proof of the theorem. They refer to the *biscope* and *uniscope* of a scope, two disjoint subsets into which a scope can be partitioned: The biscope contains those members of the scope whose complement is also a member of the scope (thus they are "*bi*-polar" members). The uniscope contains the remaining members of the scope, that is, those whose complement is not also a member of the scope (thus they are "*uni*-polar" members). The following definition provides formal notation for this:

Definition B4 (Biscope and Uniscope).

(i) $\mathsf{biscope}(S) \stackrel{\text{def}}{=} S \cap \overline{S}$.

(ii) $\mathsf{uniscope}(S) \stackrel{\text{def}}{=} S - \overline{S}$.

Proposition B1 (Raising in Terms of Biscopes and Uniscopes).

$$\langle I, \beta \rangle \models \mathsf{raise}_S(F)$$

if and only if there exists a J such that

1. $\langle J, \beta \rangle \models F$,
2. $J \cap \mathsf{biscope}(S) = I \cap \mathsf{biscope}(S)$, *and*
3. $J \cap \mathsf{uniscope}(S) \subset I \cap \mathsf{uniscope}(S)$.

Proposition B2 (Switching Forgotten Literals in Interpretations).

$$\text{If } \langle I, \beta \rangle \models \mathsf{forget}_S(F) \text{ and } M \subseteq \mathsf{uniscope}(S), \text{ then } \langle I[\overline{M}], \beta \rangle \models \mathsf{forget}_S(F).$$

Proof (Proof of Theorem B2 – Completion Lemma). Let F be a completion input sentence as specified in the precondition of the theorem, with syntactic constituents as specified in Def. B1. Let F^* be the completion addendum of F. Considering the definition of circ, the theorem follows from the following two entailments, which we are going to show:

$$\mathsf{raise}_{(\mathcal{C} \cap \mathsf{POS}) \cup R}(F) \models \neg F^*. \tag{xviii}$$

$$F \wedge \neg F^* \models \mathsf{raise}_{(\mathcal{C} \cap \mathsf{POS}) \cup R}(F). \tag{xix}$$

From the definition of *completion addendum* and expanding forget and project it follows that an interpretation $\langle I, \beta\rangle$ is a model of $\neg F^*$ if and only if there exists a ground atom $p^{\mathcal{C}}(\bar{t}_p)$ (where $\bar{t}_p$ is a sequence of terms with the arity of p as length), a sequence of ground terms $\bar{t}$ with $\bar{t}_p$ as prefix and the same length as $\overline{x}$, and a K such that (A1)-(A5) hold:

(A1) $\langle I, \beta\rangle \models +p^{\mathcal{C}}(\bar{t}_p)$.
(A2) $\langle K, \beta\rangle \models F'(\bar{t})$.
(A3) $\langle K, \beta\rangle \models -p^{\mathcal{C}}(\bar{t}_p)$.
(A4) $K \cap R = I \cap R$.
(A5) $K \cap (\mathcal{C} - p^{\mathcal{C}}) = I \cap (\mathcal{C} - p^{\mathcal{C}})$.

Proof of (xviii): Consider the table below. Let $\langle I, \beta\rangle$ be a model of

$$\mathsf{raise}_{(\mathcal{C}\cap\mathsf{POS})\cup R}(F). \tag{xx}$$

We show that $\langle I, \beta\rangle$ is then also a model of $\neg F^*$:

(1)	$\langle I, \beta\rangle \models \mathsf{raise}_{(\mathcal{C}\cap\mathsf{POS})\cup R}(F)$.	assumption
(2)	There exists a J such that	} by (1) and Prop. B1
(3)	$\langle J, \beta\rangle \models \forall \overline{x}\ F'(\overline{x})$,	
(4)	$J \cap R = I \cap R$,	
(5)	$J \cap \mathcal{C} \cap \mathsf{POS} \subset I \cap \mathcal{C} \cap \mathsf{POS}$.	
(6)	There exists a ground atom $p^{\mathcal{C}}(\bar{t}_p)$ such that	} by (5)
(7)	$\langle I, \beta\rangle \models +p^{\mathcal{C}}(\bar{t}_p)$,	
(8)	$\langle J, \beta\rangle \models -p^{\mathcal{C}}(\bar{t}_p)$.	

Now let

$$K \stackrel{\text{def}}{=} J[((\mathcal{C} \cap \mathsf{POS}) - p^{\mathcal{C}}) \cap I], \tag{xxi}$$

and let $\bar{t}$ be a sequence of ground terms with prefix $\bar{t}_p$ and the same length as $\overline{x}$. We show that $p^{\mathcal{C}}(\bar{t}_p), \bar{t}, K$ satisfy (A1)–(A5), from which it follows that $\langle I, \beta\rangle \models \neg F^*$. Condition (A1) is already stated as (7). The other conditions follow from the definition of K and additional preconditions: Condition (A2) from (3) and Prop. B2, since in $F'(\overline{x})$ only literals which are positive and not in the scope of $\neg$ have a predicate from $\mathcal{C}$, hence $F'(\overline{x}) \equiv \mathsf{forget}_{\mathcal{C}\cap\mathsf{NEG}} F'(\overline{x})$. Condition (A3) follows from (8); condition (A4) from (4); and condition (A5) from (5).

Proof of (xix): Let $\langle I, \beta\rangle$ be a model of F and $\neg F^*$. Let $p^{\mathcal{C}}(\bar{t}_p), \bar{t}, K$ be objects that satisfy (A1)–(A5). Let

$$J \stackrel{\text{def}}{=} K[(p^{\mathcal{C}} - \{+p^{\mathcal{C}}(\bar{t}_p)\}) \cap I]. \tag{xxii}$$

We show that J meets conditions (1.)–(3.) of Prop. B1, from which it follows that $\langle I, \beta\rangle \models \mathsf{raise}_{(\mathcal{C}\cap\mathsf{POS})\cup R}(F)$. Conditions (2.) and (3.) of Prop. B1. follow straightforwardly from the definition of J and conditions (A1)–(A5) as indicated in the following table:

(9)	$J \cap R = I \cap R$.	by (A4)
(10)	$J \cap \mathcal{C} \cap \mathsf{POS} \subset I \cap \mathcal{C} \cap \mathsf{POS}$.	by (A1), (A3), (A5)

It remains to show condition (3.) of Prop. B1, that is, $\langle J, \beta\rangle \models F$, or equivalently $\langle J, \beta\rangle \models \forall \overline{x}\ F'(\overline{x})$. We show this by proving $\langle J, \beta\rangle \models \forall \overline{x}\ F'(\overline{u})$, where $\overline{u}$ is an arbitrary sequence of ground terms with the same length as $\overline{x}$. The following are two auxiliary statements:

(11) $\langle I, \beta\rangle \models \forall \overline{x}\ F'(\overline{x})$. by the assumption $\langle I, \beta\rangle \models F$
(12) $J = I[\{-p^{\mathcal{C}}(\overline{t}_p)\}]$. by (A4), (A5) and the definition of J

We first consider the case where $\overline{t}_p$ is not a prefix of $\overline{u}$. The sole occurrence of $p^{\mathcal{C}}$ in $F'(\overline{u})$ is then instantiated by a sequence of ground terms that is different from $\overline{t}_p$. Thus the value of $p^{\mathcal{C}}(\overline{t}_p)$ in J is irrelevant to whether $\langle J, \beta\rangle$ models $F'(\overline{u})$. From (11) and (12) then follows $\langle J, \beta\rangle \models F'(\overline{u})$. We now consider the other case, where $\overline{t}_p$ is a prefix of $\overline{u}$. The interpretation $\langle J, \beta\rangle$ is a model of $F'(\overline{u})$ if it is a model of H and, for all predicates $q \in \mathsf{PREDS}$, it is a model of $+q^{\mathcal{C}}(\overline{u}_q) \vee G_q(\overline{u}_q)$, where $\overline{u}_q$ is the prefix of $\overline{u}$ with the arity of q as length. Of these disjunctions, only the one in which q is p, that is, the one which contains the sole occurrence of $p^{\mathcal{C}}$ in $F'(\overline{u})$, is affected by the value of $p^{\mathcal{C}}(\overline{t}_p)$ in J. We show that $\langle J, \beta\rangle$ is a model of this disjunct, and thus also a model of $F'(\overline{u})$, which completes the proof:

(13) $\langle K, \beta\rangle \models G_p(\overline{t}_p)$. by (A2), (A3) and the definition of F'
(14) $\langle J, \beta\rangle \models G_p(\overline{t}_p)$. by (13) and the definition of J, since $p^{\mathcal{C}}$ is not in G_p
(15) $\langle J, \beta\rangle \models +p^{\mathcal{C}}(\overline{t}_p) \vee G_p(\overline{t}_p)$. by (14)
(16) $\langle J, \beta\rangle \models F'(\overline{u})$. by (11), (12), (15) and the definition of F' □

As a straightforward consequence of Theorems B2 and B1, Theorem B3 now states the "correctness" of ans-completion:

Theorem B3 (Correctness of Ans-Completion). *If F is a completion input sentence, then* ans-completion(F) *is equivalent to the semantic completion of F, and also to the syntactic completion of F.*

Proof. Easy to see from the respective definitions and Theorems B1 and B2. □

The correspondence of Clark's completion to circumscription shown in this section basically follows the way described in [LL06], but with some differences: Obviously, we use forgetting/projection instead of *weakest sufficient condition*, which is no essential difference since both are straightforwardly definable in terms of each other. We consider first-order logic, where [LL06] is confined to propositional logic. In the formulas given as input to completion and to circumscription we distinguish head literals by placing them in predicate group $\mathcal{C}$. This allows us to use general circumscription instead of pointwise circumscription (both are equivalent if the circumscribed predicates occur only positively in the circumscribed formula). Furthermore, it allows us to state *equivalence* between Clark's completion and circumscription in Theorem B3, whereas the related Proposition 4 in [LL06] just makes the weaker statement that the semantically defined completion of F_1 is equivalent to Clark's completion of F_2 for *some* F_2 *that is equivalent to* F_1.

C Semantic Extraction of the Completion Addendum

As indicated in Sect. 6, *literal* projection can be applied to extract to the syntactic completion addendum from the conjunction of a sentence and its syntactic or semantic completion addendum. This extraction by projection is useful in cases where the syntactic completion addendum is not explicitly given, but just a sentence known to be equivalent to the conjunction of some original formula with its completion addendum, as, for example, certain circumscriptions according to Theorem B2. The following theorem gives a precise account of this extraction.

Theorem C1 (Semantic Extraction of the Completion Addendum). *Let F be a normal completion input sentence and F^* its syntactic or semantic completion addendum. Then*

$$\mathsf{forget}_{\mathcal{C}\cap\mathsf{POS}}(F \wedge F^*)$$

is equivalent to the syntactic completion addendum of F.

Proof. Let F and F^* be as specified in the preconditions of the theorem. Consider the following table:

$$\begin{array}{lll}
(1) & & \mathsf{forget}_{\mathcal{C}\cap\mathsf{POS}}(F \wedge F^*) \\
(2) & \equiv & \mathsf{forget}_{\mathcal{C}\cap\mathsf{POS}}(\forall\overline{x}((\bigwedge_{p\in\mathsf{PREDS}}(+p^{\mathcal{C}}(\overline{x}_p)\vee G_p(\overline{x}_p))) \wedge (\bigwedge_{p\in\mathsf{PREDS}}(-p^{\mathcal{C}}(\overline{x})\vee\neg G_p(\overline{x})))))\\
(3) & \equiv & \mathsf{forget}_{\mathcal{C}\cap\mathsf{POS}}(\bigwedge_{p\in\mathsf{PREDS}}(\forall\overline{x}_p\ ((+p^{\mathcal{C}}(\overline{x}_p) \wedge \neg G_p(\overline{x}_p)) \vee (-p^{\mathcal{C}}(\overline{x}_p) \wedge G_p(\overline{x}_p)))))\\
(4) & \equiv & \bigwedge_{p\in\mathsf{PREDS}}(\forall\overline{x}_p\ ((\top \wedge \neg G_p(\overline{x}_p)) \vee (-p^{\mathcal{C}}(\overline{x}_p) \wedge G_p(\overline{x}_p))))\\
(5) & \equiv & \bigwedge_{p\in\mathsf{PREDS}}(\forall\overline{x}_p\ (-p^{\mathcal{C}}(\overline{x}_p) \vee \neg G_p(\overline{x}_p)))\\
(6) & \equiv & \forall\overline{x}\ (\bigwedge_{p\in\mathsf{PREDS}}(-p^{\mathcal{C}}(\overline{x}) \vee \neg G_p(\overline{x}))).
\end{array}$$

From Defs. B1 and B2 it follows that the argument formula of forget in (1), that is, $F \wedge F^*$, is equivalent to the argument formula in (2), where the syntactic constituents correspond to the mentioned definitions. Equivalence of (4) to (3) is justified by properties of projection, e.g. [Wer08, Theorem 4]. The other equivalences are obtained by straightforwardly replacing subformulas with logically equivalent formulas. Sentence (6) is the syntactic completion addendum of F. □

D Replicating Structures

As introduced in Sect. 2, a structure is a set of ground literals that contains for all ground atoms A exactly one of $+A$ or $-A$. In Sect. 4 we have introduced *predicate groups*, partitions of the predicates where each "original predicate" has a correspondent in each partition. In this section, we introduce a convenient representation for structures over predicates that are grouped in this way: A structure is then represented as a tuple with one place for each predicate group. The members of the tuple are versions of those literals of the structure whose predicate is in the respective group, but with the predicate replaced by a symbol without group indicator, like the "original predicate". Since now just the position within the tuple indicates the predicate group, members of the tuple can

be moved from one place in the tuple to another to express systematic renaming operations on the structure in a convenient way. We call such a tuple representation of a structure a *replicating structure*. The following definitions and remarks make this precise:

Definition D1 (Ungrouped Atom, Literal and Structure). An *ungrouped atom* (*ungrouped literal*, resp.) is like an atom (literal, resp.) except that in place of the predicate it has just a predicate position, that is, a member of PREDS (see Sect. 4). If A (L, resp.) is an ungrouped atom $p(t_1,\ldots,t_n)$, (ungrouped literal $+p(t_1,\ldots,t_n)$ or $-p(t_1,\ldots,t_n)$, resp.) and P is a predicate group, then A^P (L^P, resp.) denotes the atom $p^P(t_1,\ldots,t_n)$ (literal $+p^P(t_1,\ldots,t_n)$ or $-p^P(t_1,\ldots,t_n)$, resp.). An *ungrouped structure* is a set of ungrouped ground literals that contains for all ungrouped ground atoms A exactly one of $+A$ or $-A$.

Definition D2 (Replicating Structure). A *replicating structure* is a tuple with length ≥ 1 of ungrouped structures.

We overload the symbols A, L, I, J, K, to denote also *ungrouped* atoms, literals and structures, respectively, and in case of I, J, K also *replicating* structures, if unambiguous from the context. In addition, we use POS, (NEG, resp.) also to denote the set of all positive (negative, resp.) *ungrouped* ground literals, depending on the context.

If we write a *replicating* structure $\langle I_1,\ldots,I_n\rangle$ in place of a structure (such as on the left side of the $\models$ symbol), we understand it as representation of the structure that is obtained with respect to a given tuple $\langle P_1,\ldots,P_n\rangle$ of predicate groups as:

$$\bigcup_{i\in\{1,\ldots,n\}} \{L^{P_i} \mid L \in I_i\}. \tag{xxiii}$$

As an example for the application of replicating structures, we use them in the proof of Prop. D2 below. This proof refers to properties of replicating structures which are compiled in the following Prop. D1. Of course, these properties are specific instances of more general properties which might be defined in terms of primitive operations on replicating structures such as copying an ungrouped structure from one place in the tuple into another, but we leave elaboration of these general properties to future work.

Proposition D1 (Selected Properties of Replicating Structures). *Let replicating structures represent structures with respect to $\langle\mathcal{C},\mathcal{F},\mathcal{O}\rangle$. If F is over $\mathcal{C}\cup\mathcal{F}\cup\mathcal{O}$, then*

(i) $\langle\langle I_1,I_2,I_3\rangle,\beta\rangle \models \mathsf{rename}_{\mathcal{F}\mapsto\mathcal{C}}(F)$ *iff* $\langle\langle I_1,I_1,I_3\rangle,\beta\rangle \models F$.

(ii) $\langle\langle I_1,I_2,I_3\rangle,\beta\rangle \models \mathsf{rename}_{\mathcal{O}\mapsto\mathcal{C}}(F)$ *iff* $\langle\langle I_1,I_2,I_1\rangle,\beta\rangle \models F$.

(iii) $\langle\langle I_1,I_2,I_3\rangle,\beta\rangle \models \mathsf{rename}_{\mathcal{O}\mapsto\mathcal{F}}(F)$ *iff* $\langle\langle I_1,I_2,I_2\rangle,\beta\rangle \models F$.

(iv) $\langle\langle I_1,I_2,I_3\rangle,\beta\rangle \models \mathsf{raise}_{(\mathcal{C}\cap\mathsf{POS})\cup\mathcal{F}}(F)$ *iff there exist* J_1, J_3 *such that* $J_1\cap\mathsf{POS}\subset I_1\cap\mathsf{POS}$ and $\langle\langle J_1,I_2,J_3\rangle,\beta\rangle \models F$.

(v) $\langle\langle I_1,I_2,I_3\rangle,\beta\rangle \models \mathsf{raise}_{(\mathcal{C}\cap\mathsf{POS})}(F)$ *iff there exist* J_1, J_2, J_3 *such that* $J_1\cap\mathsf{POS}\subset I_1\cap\mathsf{POS}$ and $\langle\langle J_1,J_2,J_3\rangle,\beta\rangle \models F$.

(vi) *If* $F \equiv \mathsf{forget}_{\mathcal{C}\cap\mathsf{NEG}}(F)$ *and* $I_1 \cap \mathsf{POS} \subseteq J_1 \cap \mathsf{POS}$*, then*
If $\langle\langle I_1, I_2, I_3\rangle, \beta\rangle \models F$ then $\langle\langle J_1, I_2, I_3\rangle, \beta\rangle \models F$.

(vii) *If* $F \equiv \mathsf{forget}_{\mathcal{F}\cap\mathsf{NEG}}(F)$ *and* $I_2 \cap \mathsf{POS} \subseteq J_2 \cap \mathsf{POS}$*, then*
If $\langle\langle I_1, I_2, I_3\rangle, \beta\rangle \models F$ then $\langle\langle I_1, J_2, I_3\rangle, \beta\rangle \models F$.

(viii) *If* $F \equiv \mathsf{forget}_{\mathcal{O}\cap\mathsf{POS}}(F)$ *and* $J_3 \cap \mathsf{POS} \subseteq I_3 \cap \mathsf{POS}$*, then*
If $\langle\langle I_1, I_2, I_3\rangle, \beta\rangle \models F$ then $\langle\langle I_1, I_2, J_3\rangle, \beta\rangle \models F$.

Proof. Prop. D1.vi–D1.viii follow from Prop. B2. The remaining propositions follow easily from the definitions of the involved operators. □

Propositions D2.i and D2.ii below show that a stable model of a formula representing a logic program is also a model of the completion, and, moreover, if the formula is restricted such that negation as failure is disallowed, then a stable model is a minimal model of the completion. At least for normal logic programs, these relationships between stable model semantics and completion are well known and proven elegantly with the fixed-point characterization of completion as supported models in [MS92]. Thus, aside of the applicability to less restricted classes of logic programs, the point in providing these propositions is to give examples that indicate how the framework devised in this report – characterizations of semantics for logic programming in terms of circumscription and projection and operations on the left side of the $\models$ symbol by means of replicating structures – are applied "in practice" to prove theorems.

Proposition D2 (A Stable Model is a Model of Clark's Completion). *If* F *is over* $\mathcal{C} \cup \mathcal{F} \cup \mathcal{O}$ *and* $F \equiv \mathsf{forget}_{\mathcal{O}\cap\mathsf{POS}}(F)$*, then*

(i) $\mathsf{ans\text{-}stable}(F) \models \mathsf{ans\text{-}completion}(F)$.

(ii) *If, in addition,* $F \equiv \mathsf{forget}_{\mathcal{F}\cap\mathsf{NEG}}(F)$*, then*

$$\mathsf{ans\text{-}stable}(F) \models \mathsf{circ}_{\mathcal{C}\cap\mathsf{POS}}(\mathsf{ans\text{-}completion}(F)).$$

Proof. (Prop. D2.i) Consider the table below. Let F be a formula satisfying the precondition of the proposition, step (1). Let replicating structures represent structures with respect to $\langle\mathcal{C}, \mathcal{F}, \mathcal{O}\rangle$. Let $\langle\langle I_1, I_2, I_3\rangle, \beta\rangle$ be an interpretation that satisfies the left side of the proposition, step (2). We derive that this interpretation also satisfies the right side, step (16). In the annotations at the rightmost column of the table we abbreviate *expanding the definition of* with *exp.* and *contracting the definition of* with *con.*.

(1)	$F \equiv \mathsf{forget}_{\mathcal{O}\cap\mathsf{POS}}(F)$.	assumption
(2)	$\langle\langle I_1, I_2, I_3\rangle, \beta\rangle \models \mathsf{ans\text{-}stable}(F)$.	assumption
(3)	$\langle\langle I_1, I_2, I_3\rangle, \beta\rangle \models \mathsf{rename}_{\mathcal{F}\mapsto\mathcal{C}}(\mathsf{circ}_{(\mathcal{C}\cap\mathsf{POS})\cup\mathcal{F}}(\mathsf{rename}_{\mathcal{O}\mapsto\mathcal{C}}(F)))$.	by (2), exp. ans-stable
(4)	$\langle\langle I_1, I_1, I_3\rangle, \beta\rangle \models \mathsf{circ}_{(\mathcal{C}\cap\mathsf{POS})\cup\mathcal{F}}(\mathsf{rename}_{\mathcal{O}\mapsto\mathcal{C}}(F))$.	by (3), Prop. D1.i
(5)	$\langle\langle I_1, I_1, I_3\rangle, \beta\rangle \models \mathsf{rename}_{\mathcal{O}\mapsto\mathcal{C}}(F)$.	by (4), exp. circ
(6)	$\langle\langle I_1, I_1, I_1\rangle, \beta\rangle \models F$.	by (5), Prop. D1.ii
(7)	$\langle\langle I_1, I_1, I_3\rangle, \beta\rangle \models \mathsf{rename}_{\mathcal{O}\mapsto\mathcal{F}}(F)$.	by (6), Prop. D1.iii
(8)	$\langle\langle I_1, I_1, I_3\rangle, \beta\rangle \models \neg\mathsf{raise}_{(\mathcal{C}\cap\mathsf{POS})\cup\mathcal{F}}(\mathsf{rename}_{\mathcal{O}\mapsto\mathcal{C}}(F))$.	by (4), exp. circ

(9) There do not exist J_1, J_3 such that
$J_1 \cap \mathsf{POS} \subset I_1 \cap \mathsf{POS}$ and
$\langle\langle J_1, I_1, J_3\rangle, \beta\rangle \models \mathsf{rename}_{\mathcal{O}\mapsto\mathcal{C}}(F)$. by (8), Prop. D1.iv

(10) There does not exist a J_1 such that
$J_1 \cap \mathsf{POS} \subset I_1 \cap \mathsf{POS}$ and
$\langle\langle J_1, I_1, J_1\rangle, \beta\rangle \models F$. by (9), Prop. D1.ii

(11) There does not exist a J_1 such that
$J_1 \cap \mathsf{POS} \subset I_1 \cap \mathsf{POS}$ and
$\langle\langle J_1, I_1, I_1\rangle, \beta\rangle \models F$. by (10), (1), Prop. D1.viii

(12) There do not exist J_1, J_3 such that
$J_1 \cap \mathsf{POS} \subset I_1 \cap \mathsf{POS}$ and
$\langle\langle J_1, I_1, J_3\rangle, \beta\rangle \models \mathsf{rename}_{\mathcal{O}\mapsto\mathcal{F}}(F)$. by (11), Prop. D1.iii

(13) $\langle\langle I_1, I_1, I_3\rangle, \beta\rangle \models \neg\mathsf{raise}_{(\mathcal{C}\cap\mathsf{POS})\cup\mathcal{F}}(\mathsf{rename}_{\mathcal{O}\mapsto\mathcal{F}}(F))$. by (12), Prop. D1.iv

(14) $\langle\langle I_1, I_1, I_3\rangle, \beta\rangle \models \mathsf{circ}_{(\mathcal{C}\cap\mathsf{POS})\cup\mathcal{F}}(\mathsf{rename}_{\mathcal{O}\mapsto\mathcal{F}}(F))$. by (13), (7), con. circ

(15) $\langle\langle I_1, I_2, I_3\rangle, \beta\rangle \models \mathsf{rename}_{\mathcal{F}\mapsto\mathcal{C}}(\mathsf{circ}_{(\mathcal{C}\cap\mathsf{POS})\cup\mathcal{F}}(\mathsf{rename}_{\mathcal{O}\mapsto\mathcal{F}}(F)))$. by (14), Prop. D1.i

(16) $\langle\langle I_1, I_2, I_3\rangle, \beta\rangle \models \mathsf{ans\text{-}completion}(F)$. by (5), con. ans-completion

(Prop. D2.ii) Consider the proof of Prop. D2.i. In the following table we extend this proof by the additional precondition as an assumption, step (17), and then deriving as step (24) that $\langle\langle I_1, I_2, I_3\rangle, \beta\rangle \models \neg\mathsf{raise}_{\mathcal{C}\cap\mathsf{POS}}(\mathsf{ans\text{-}completion}(F))$. The proposition then follows from steps (24) and (16) by contracting circ.

(17) $F \equiv \mathsf{forget}_{\mathcal{F}\cap\mathsf{NEG}}(F)$. assumption

(18) There does not exist a J_1 s.t.
$J_1 \cap \mathsf{POS} \subset I_1 \cap \mathsf{POS}$ and
$\langle\langle J_1, J_1, J_1\rangle, \beta\rangle \models F$. by (17), (10), Prop. D1.vii

(19) There do not exist J_1, J_3 such that
$J_1 \cap \mathsf{POS} \subset I_1 \cap \mathsf{POS}$ and
$\langle\langle J_1, J_1, J_3\rangle, \beta\rangle \models \mathsf{rename}_{\mathcal{O}\mapsto\mathcal{F}}(F)$. by (18), Prop. D1.iii

(20) There do not exist J_1, J_3 such that
$J_1 \cap \mathsf{POS} \subset I_1 \cap \mathsf{POS}$ and
$\langle\langle J_1, J_1, J_3\rangle, \beta\rangle \models \mathsf{rename}_{\mathcal{O}\mapsto\mathcal{F}}(F)$ and
$\langle\langle J_1, J_1, J_3\rangle, \beta\rangle \models \neg\mathsf{raise}_{(\mathcal{C}\cap\mathsf{POS})\cup\mathcal{F}}(\mathsf{rename}_{\mathcal{O}\mapsto\mathcal{F}}(F))$. by (19)

(21) There do not exist J_1, J_3 such that
$J_1 \cap \mathsf{POS} \subset I_1 \cap \mathsf{POS}$ and
$\langle\langle J_1, J_1, J_3\rangle, \beta\rangle \models \mathsf{circ}_{(\mathcal{C}\cap\mathsf{POS})\cup\mathcal{F}}(\mathsf{rename}_{\mathcal{O}\mapsto\mathcal{F}}(F))$. by (20), con. circ

(22) There do not exist J_1, J_2, J_3 such that
$J_1 \cap \mathsf{POS} \subset I_1 \cap \mathsf{POS}$ and
$\langle\langle J_1, J_2, J_3\rangle, \beta\rangle \models \mathsf{rename}_{\mathcal{F}\mapsto\mathcal{C}}(\mathsf{circ}_{(\mathcal{C}\cap\mathsf{POS})\cup\mathcal{F}}(\mathsf{rename}_{\mathcal{O}\mapsto\mathcal{F}}(F)))$. by (21), Prop. D1.i

(23) There do not exist J_1, J_2, I_3' such that
$J_1 \cap \mathsf{POS} \subset I_1 \cap \mathsf{POS}$ and
$\langle\langle J_1, I_2', I_3'\rangle, \beta\rangle \models \mathsf{ans\text{-}completion}(F)$. by (22), con. ans-completion

(24) $\langle\langle I_1, I_2, I_3\rangle, \beta\rangle \models \neg\mathsf{raise}_{\mathcal{C}\cap\mathsf{POS}}(\mathsf{ans\text{-}completion}(F))$. by (23), Prop. D1.v

□