Logic Programming Revisited from a Classical Standpoint

Eric A. Martin

University of New South Wales, Australia
emartin@cse.unsw.edu.au

Technical Report
UNSW-CSE-TR-0821
September 2008

THE UNIVERSITY OF
NEW SOUTH WALES

School of Computer Science and Engineering
The University of New South Wales
Sydney 2052, Australia

Abstract

Logic programming has developed as a rich field, built over a logical substratum whose main constituent is a nonclassical form of negation, sometimes coexisting with classical negation. The field has seen the advent of a number of alternative semantics, with Kripke-Kleene semantics, the well founded semantics, the stable model semantics, and answer-set programming standing out as the most successful of all. We show that using classical negation only, all aforementioned semantics are particular cases of a unique semantics applied to a general notion of logic program possibly transformed following a simple procedure. The notions and results presented in this paper give a classical perspective on the field of logic programming and broaden its scope, as that simple procedure suggests a number of possible transformations of a logic program, that can be classified into families, some members of some of those families matching a particular paradigm in the field. The paper demonstrates that logic programming can be developed in such a way that negation does not present itself as an intrinsically complex operator, hard to interpret properly and that needs a complicated formal apparatus to be fully apprehended, but still in a way that accommodates the semantics that have put nonclassical negation at the center of their investigations.

1 Background

1.1 Nonclassical versus classical

Since its inception, the field of logic programming has embraced an increasingly complex diaspora of rules, the syntax of the rules becoming more and more general as the field developed. Taking conjunction and disjunction as only boolean operators, the bodies of the rules have been built from atomic formulas only, from atomic formulas that are possibly negated with a nonclassical form of negation, or from atomic formulas that are possibly negated with classical negation and then possibly negated with a nonclassical form of negation. The heads of the rules have been assumed to be atomic formulas, or either atomic formulas that are possibly negated with classical negation. Disjunction has made its way into the heads of the rules, and a large number of extra constraints imposed on the rules that make up a logic program have been proposed. It has even been advocated to use more than two kinds of negation [1, 2]. All these developments took place as part of the advances in the field of nonmonotonic reasoning [15].

Starting with the simplest case of sets of rules whose heads are atomic formulas and whose bodies result from the application of conjunction and disjunction to atomic formulas only, a recurring question has been: what is the intended meaning of a set of rules? that translates into: what are the intended interpretations of a set of rules? Some approaches seek a unique intended interpretation, while other approaches accommodate many. In a first-order setting, the intended interpretations have been selected from the class of all structures or from the more restricted class of all Herbrand structures, that give every individual a unique name. But more fundamentally, the crux of the investigations in the field has been to characterize what cannot or should not be derived from a set of rules [3]. Modal-theoretic interpretations have made particularly clear that the primary form of negation in logic programming is nonprovability, whether or not classical negation can be used to assert falsity. Still, logic programming can be looked at from a totally different angle, that we qualify as "classical" as it recognizes no other form of negation but classical negation, and to a lesser extent as it does not require that intended interpretations be restricted to the class of Herbrand interpretations. This classical standpoint relies on two key principles.

The first principle is that a set of positive rules, that is, rules whose heads are atomic formulas, can be thought of as a set of rules that are both positive and negative, that is, rules whose heads are atomic formulas or negations of atomic formulas, where the negative rules are left *implicit* because they are fully determined by the positive rules thanks to a duality principle. This idea is far from novel; it is nothing more than a variation on the notion of Clark's completion of a logic program [5]. The usual justification for Clark's completion is that it yields the intended meaning of the original set of rules; it represents what a person who writes or reads the set of rules actually has in mind or understands. When we consider that the negative rules are left implicit, we adopt a similar stance. Clark's completion does not transform a set of positive rules into a set of positive and negative rules, but rather into a set of logical equivalences

augmented with an axiomatization of a notion of pseudo-equality. Our formalization is somehow a streamlined version of Clark's completion. With positive rules only, one can only infer some negative information by failing to generate some positive information—the process known as negation as finite failure that certainly compels us to adopt the view that negation in logic programming is essentially nonclassical. But given both positive and negative rules, one can generate both positive and negative information, and conceive of negation as finite failure as an ingenious proof technique to generate negative information from the positive rules only, as an alternative to generating negative information using both the positive and the negative rules. This paper will demonstrate that this view is perfectly tenable; classical negation is all one needs, and negation as finite failure can be understood as part of a more general proof procedure that generates nothing but logical consequences. Not surprisingly, this will result in a semantics which, in case the class of intended interpretations is the class of Herbrand interpretations, is fundamentally equivalent to Kripke-Kleene semantics [8]. We will not make any restriction on the class of intended interpretations, and present our semantics in the most general setting.

The second principle will allow us to stick to our semantics as "the base semantics," while still accounting for the well founded semantics [9], the stable model semantics [10], and answer-set programming [11]. This second principle is based on the idea that any of those semantics "force" some assumptions to be made in some parts of some rules, resulting in a new logic program whose base semantics is precisely the desired semantics of the original program. To force some assumptions to be made in some parts of some rules, we use a particular kind of transformation of a logical formula, that we now introduce. Consider two formulas, φ, of the form

$$\exists x\big(p(x) \vee q(x)\big) \wedge \exists x\big(p(x) \vee r(x)\big),$$

and ψ, of the form

$$\exists x\big(p(x) \vee q(x)\big) \wedge \exists x\big(p(x) \vee p(a) \vee p(b) \vee r(x)\big).$$

Then we can read ψ as "φ, where the second occurrence of $p(x)$ is assumed to be true in case x is either a or b." Or to put it another way, if in φ, we hypothesize that $p(a)$ and $p(b)$ are true in the context given by the second occurrence of $p(x)$ in φ, then we get (a logical representation equivalent to) ψ. Though the formalization to come will differ syntactically from what this example illustrates, it will formalize the notion of "transforming a formula into another by making some contextual hypotheses in the former," similarly to the way φ can be transformed into ψ by making the hypotheses $p(a)$ and $p(b)$ in the context given by the second occurrence of $p(x)$ in φ. Having this notion of "contextual hypothesis" and associated formula transformation in hand, our "classical" approach to logic programming will replace the question of "what should be acknowledged to fail to be derived from a logic program?'—the question at the heart of the well known semantics in the "nonclassical" approaches to logic programming—by the question of "what contextual hypotheses should be made in the bodies of the rules of a logic program?" This will allow us to revisit the main semantics that have been proposed and view them as particular members of families of semantics, and more particularly, as those members

that are "maximally biased" towards negative information. For an illustration, consider a vocabulary with a constant $\overline{0}$, a unary function symbol s and a unary predicate symbol p, and the logic program $\mathcal{P}$ consisting of the following rule.

$$p(X) \leftarrow p(s(s(X)))$$

Given a natural number n, write $\overline{n}$ for the term obtained from $\overline{0}$ by n applications of s. Applied to $\mathcal{P}$, the well founded semantics makes all of $p(\overline{n})$, $n \in \mathbb{N}$, false in its intended model of $\mathcal{P}$, based on the principle that when a logic program presents an infinite descending chain of atoms, all members of that chain should be set to false. It turns out that this is a particular case of a more general principle, that will be formalized in the body of the paper, consistent with a large number of models of $\mathcal{P}$, including in particular

- structures in which $p(\overline{n})$ is false for all n's;
- structures in which $p(\overline{n})$ is true for all n's;
- structures in which $p(\overline{n})$ is false for all even n's, but true for all odd n's;
- structures in which $p(\overline{n})$ is true for all even n's, but false for all odd n's.

So this more general principle isolates a number of Herbrand models one of which is maximally biased towards negative information, that happens to be the intended model advocated by the well founded semantics.

1.2 A mechanistic view on rules

The rules that make up a logic program are expressions of the form

$$\mathit{head} \leftarrow \mathit{body}$$

that are read in many possible ways. One can view $\leftarrow$ as a link between cause and effect and conceive of *body* as a statement that if *activated*, allows the rule to *fire* and *head* to be *generated*; when formally defined, this amount to a kind of *operational* semantics. Or one can view $\leftarrow$ as a link between antecedent and consequent and conceive of *body* as a statement that if *true*, allows the rule to be *logically applicable* and *head* to be established as *true*; when formally defined, this amounts to a *denotational* semantics. A legitimate aim is to propose both an operational and a denotational semantics, and make sure that they match. In this paper, we propose an operational semantics as it is the shortest path to casting Kripke-Kleene semantics, the well founded semantics, the stable model semantics, and answer-set programming into our framework. We also have a denotational semantics but will not present it in this paper.

Let us specify a bit more the syntactic structure of rules and the process by which they fire. Recall that a formula is in *negation normal form* if negation is applied to atomic formulas only; so formulas in negation normal form are built from *literals* (atomic formulas and their negations) using disjunction, conjunction, existential quantification, and universal quantification. Assume that every rule $\mathit{head} \leftarrow \mathit{body}$ of a logic program is such that *head* is a literal and *body* is

a formula in negation normal form. Firing rules causes literals—the heads of the rules that fire—to be *generated*. Literals can be combined into formulas in negation normal form some of which can, thanks to the generated literals, be *inferred*. We impose that inferring formulas in negation normal form be a *constructive* process; so $p \vee \neg p$ can be inferred provided that p or $\neg p$ has been generated, and $\exists x\, p(x)$ can be inferred provided that $p(t)$ has been generated for at least one closed term t.

Having literals as heads of the rules of a logic program is natural in relation to answer-set programming. As sketched in the previous section, it is also natural in relation to the Kripke-Kleene semantics, the well founded semantics and the stable model semantics, all settings that more or less directly, borrow from Clark's completion of a logic program. We will not use Clark's completion; rather, we will rely on notions of *duality* and *symmetry* that we now introduce. Given a formula φ in negation normal form, define the dual of φ as the formula $\sim\varphi$ obtained from φ by changing disjunction into conjunction, conjunction into disjunction, existential quantification into universal quantification, universal quantification into existential quantification, by negating nonnegated atomic formulas, and by deleting all negation signs (in front of atomic formulas). For instance, if φ is

$$(p(X) \vee \neg q(X)) \wedge (\neg p(X) \vee r(X))$$

then the dual $\sim\varphi$ of φ is

$$(\neg p(X) \wedge q(X)) \vee (p(X) \wedge \neg r(X)).$$

Now say that a logic program $\mathcal{P}$ is symmetric if the bodies of all rules are formulas in negation normal form and if for all $n \in \mathbb{N}$ and n-ary predicate symbol $\wp$, $\mathcal{P}$ contains exactly two rules of respective form $\wp(v_1, \ldots, v_n) \leftarrow \varphi^+_\wp$ and $\neg\wp(v_1, \ldots, v_n) \leftarrow \varphi^-_\wp$ that are dual of each other in the sense that $\varphi^-_\wp$ and $\varphi^+_\wp$ are dual of each other. With these notions in hand, we will be able to view all three semantics as applied to symmetric logic programs. The working hypothesis is that all three semantics deal with symmetric logic programs even though traditionally, many rules can have a head built from a given predicate symbol and only the positive rules are explicitly given, with the negative rules being implicit; this is legitimate as negation normal form is not restrictive, a straightforward syntactic transformation allows one to merge all rules whose heads are built from a given predicate symbol, and every negative rule is perfectly determined by its dual positive rule.

It seems natural to allow rules to fire finitely often only, as this immediately suggests obvious implementations. But we can think theoretically and assume that rules are allowed to fire transfinitely many times—and all fixed point semantics happily go for it [7, 6]. So after all rules have fired any finite number of times, they could fire for the ω-th time, and then for the $(\omega + 1)$-st time, and then for the $(\omega + 2)$-nd time... and then for the $(\omega \times 2)$-nd time, etc. For instance, if every literal of the form $p(\overline{n})$, $n \in \mathbb{N}$, has been generated at some stage before stage ω, and if all individuals in the domains of all possible interpretations are denoted by a term of the form $\overline{n}$, then $\forall x\, p(x)$ can be inferred at stage ω, a point from which any rule whose body is $\forall x\, p(x)$ can fire. Formalizing the process by which rules fire transfinitely often determines the set of literals $[\,\mathcal{P}\,]$ generated by a set $\mathcal{P}$ of (positive and negative) rules. It is an operational semantics, and

it captures our base semantics as applied to the positive part of a symmetric logic program.

1.3 Making contextual hypotheses

We do not propose any other operational semantics than what has just been described as $[\,\mathcal{P}\,]$. Let us make what we said earlier about the relationships to the well founded semantics, the stable model semantics and answer-set programming, a bit more precise. Consider a set E of literals. Also consider a function Ω, defined on the set of bodies of the rules in $\mathcal{P}$, that returns, for the body φ of each rule in $\mathcal{P}$, a selected set of occurrences of literals in φ. We could represent this mapping graphically using check marks, writing for instance

$$\big(p(X) \vee \underset{\checkmark}{q(X)}\big) \wedge \big(\underset{\checkmark}{p(X)} \vee r(X)\big)$$

to indicate that the selected occurrences of literals in the formula φ defined as

$$\big(p(X) \vee q(X)\big) \wedge \big(p(X) \vee r(X)\big)$$

are the unique occurrence of $q(X)$ and the second occurrence of $p(X)$. Now with E and Ω in hand, we define from $\mathcal{P}$ a new set of rules, denoted $\mathcal{P} +_\Omega E$, that formalizes the request that intuitively reads: "in the bodies of the rules of $\mathcal{P}$, use E as a set of hypotheses in the contexts indicated by Ω." For instance, if $\mathcal{P}$ contains the rule R defined as

$$p(X) \leftarrow \big(p(X) \vee q(X)\big) \wedge \big(p(X) \vee r(X)\big),$$

if E is defined as $\{p(\overline{2n}) \mid n \in \mathbb{N}\} \cup \{b(\overline{n}) \mid n \in \mathbb{N}\}$, and if Ω selects the unique occurrence of $q(X)$ and the second occurrence of $p(X)$ in the body of R then $\mathcal{P} +_\Omega E$ will contain a rule that is logically equivalent to

$$p(X) \leftarrow \big(p(X) \vee q(X)\big) \wedge \big(p(X) \vee \bigvee_{n\in\mathbb{N}} X \doteq \overline{2n} \vee r(X)\big)$$

where $\doteq$ denotes syntactic identity. We will see that in case $\mathcal{P}$ is symmetric, we can choose Ω and E in such a way that $[\,\mathcal{P} +_\Omega E\,]$ captures the well-founded semantics as applied to the positive rules of $\mathcal{P}$; moreover, this choice of Ω and E is a particular case of choices made according to a simple principle, that happens to be maximally biased towards negative information. Still in case where $\mathcal{P}$ is symmetric, we can also choose Ω and E in ways such that $[\,\mathcal{P} +_\Omega E\,]$ are the stable models of the positive rules of $\mathcal{P}$; similarly, these choices of Ω and E are particular cases of choices made according to a simple principle, that happen to be maximally biased towards negative information. Importantly, these correspondences are between a framework where negation is classical and frameworks where negation is meant not to be classical. Answer-set programming seems to offer a greater challenge as its syntax accommodates two kinds of negation: $\neg$, meant to be classical, and *not*, meant to be nonclassical. But the correspondence turns out to be easy to establish if one conceives of *not* as a syntactic variant to Ω. More precisely, conceive of *not literal* as a request to select $\sim$*literal*, where $\sim$*literal* is $\neg$*literal* if *literal* is an atom, and *atom* if it

is of the form $\neg atom$. Given a set of rules $\mathcal{P}$ in the bodies of which both $\neg$ and *not* might occur, consider the set of rules $\mathcal{P}'$ obtained from $\mathcal{P}$ by replacing all occurrences of *not literal* with $\sim$*literal* (so only classical negation occurs in $\mathcal{P}'$). Then set Ω to select precisely the occurrences of literals in the bodies of the rules of $\mathcal{P}'$ that have replaced an occurrence of an expression of the form *not literal* in the bodies of the rules of $\mathcal{P}$. For instance, if $\mathcal{P}$ contains the rule

$$p(X) \leftarrow \big(not\, p(X) \vee q(X)\big) \wedge \big(\neg p(X) \vee not\, \neg r(X)\big),$$

then $\mathcal{P}'$ will contain the rule

$$p(X) \leftarrow \big(\neg p(X) \vee q(X)\big) \wedge \big(\neg p(X) \vee r(X)\big),$$

and Ω will select in the body of that rule the first occurrence of $\neg p(X)$ and the occurrence of $r(X)$. We will see that we can naturally choose E in ways such that $[\,\mathcal{P}' +_\Omega E\,]$ are the answer-sets for $\mathcal{P}$ (one answer-set for each choice of E).

2 Logical background

$\mathbb{N}$ denotes the set of natural numbers and Ord the class of ordinals.

2.1 Syntax

Definition 1. A *vocabulary* is a set of (possibly nullary) function symbols and (possibly nullary) predicate symbols, none of which is the distinguished binary predicate symbol $\doteq$ (identity).

We will discuss later the distinction between $\doteq$ and equality ($=$), which note can be one of the predicate symbols in a vocabulary. Accepting nullary predicate symbols in vocabularies will allow us to formalize all notions in a setting that can be either purely propositional, or purely first-order, or hybrid. There are a few cases of degenerated vocabularies, for instance, vocabularies that contain function symbols but all of whose predicate symbols are nullary, or vocabularies that contain at least one nonnullary predicate symbol but no constant (these vocabularies can be perfectly acceptable in the usual treatment of first-order logic, but they are degenerated cases in this setting). In any event, no notion nor result will be invalidated by a degenerated underlying vocabulary, hence degenerated vocabularies are not precisely identified nor ruled out.

Notation 2. When a vocabulary contains the constant $\overline{0}$ and the unary function symbol s, we use $\overline{n}$ to refer to the term obtained from $\overline{0}$ by n applications of s.

Notation 3. We denote by $\mathcal{V}$ a vocabulary.

Notation 4. We fix a countably infinite set of (first-order) variables together with a repetition-free enumeration $(v_i)_{i\in\mathbb{N}}$ of this set.

By *term* we mean term over $\mathcal{V}$, built from the function symbols in $\mathcal{V}$ and the members of $(v_i)_{i\in\mathbb{N}}$. We say that a term is *closed* if it contains no variable.

Definition 5. The set $\mathcal{L}_{\omega_1\omega}(\mathcal{V})$ of (*infinitary*) *statements* (*over* $\mathcal{V}$) is inductively defined as the smallest set that satisfies the following conditions.

- All *literals*—*atoms* and *negated atoms*—(*over* $\mathcal{V}$), namely, all expressions of the form $\wp(t_1, \ldots, t_n)$ or $\neg\wp(t_1, \ldots, t_n)$ where $n \in \mathbb{N}$, $\wp$ is an n-ary predicate symbol in $\mathcal{V}$, and $t_1, \ldots, t_n$ are terms over $\mathcal{V}$, belong to $\mathcal{L}_{\omega_1\omega}(\mathcal{V})$.
- All *identities* and *distinctions* (*over* $\mathcal{V}$), namely, all expressions of the form $t \doteq t'$ or $t \neq t'$where t and t' are terms over $\mathcal{V}$, belong to $\mathcal{L}_{\omega_1\omega}(\mathcal{V})$.[1]
- All *disjunctive statements* (*over* $\mathcal{V}$), namely, all expressions of the form $\bigvee X$ with X a countable set of statements over $\mathcal{V}$, belong to $\mathcal{L}_{\omega_1\omega}(\mathcal{V})$.
- All *conjunctive statements* (*over* $\mathcal{V}$), namely, all expressions of the form $\bigwedge X$ with X a countable set of statements over $\mathcal{V}$, belong to $\mathcal{L}_{\omega_1\omega}(\mathcal{V})$.
- All *existential statements* (*over* $\mathcal{V}$), namely, all expressions of the form $\exists x\, \varphi$ where x is a variable and φ is a statement over $\mathcal{V}$ that has x as a free variable, belong to $\mathcal{L}_{\omega_1\omega}(\mathcal{V})$.
- All *universal statements* (*over* $\mathcal{V}$), namely, all expressions of the form $\forall x\, \varphi$ where x is a variable and φ is a statement over $\mathcal{V}$ that has x as a free variable, belong to $\mathcal{L}_{\omega_1\omega}(\mathcal{V})$.

A few observations about the definition of $\mathcal{L}_{\omega_1\omega}(\mathcal{V})$ are in order. First, negation is assumed to be applicable to atoms only, which amounts to imposing a negation normal form, at no loss of generality. This is technically convenient, and often used in logic programming. Second, the application of quantifiers is restricted to statements that have the quantified variable as a free variable, again at no loss of generality. This is to embed the propositional framework neatly in a first-order setting: if $\mathcal{V}$ consists of nullary predicate symbols only then $\mathcal{L}_{\omega_1\omega}(\mathcal{V})$ is just the infinitary propositional language built on $\mathcal{V}$. Third, if we wanted to sometimes restrict some concepts to a set of finite statements, then we would still be happy with disjunction and conjunction being unary operators on sets of statements, that would then be required to be finite. This treatment of disjunction and conjunction, which contrasts to the traditional view of binary operators on pairs of formulas, does not only make $\mathcal{L}_{\omega_1\omega}(\mathcal{V})$ a more natural extension of the set of finite first-order formulas over $\mathcal{V}$. It also simplifies the formal developments. In particular, there is no need to introduce two extra symbols `true` and `false`, as is usually done in logic programming, since $\bigwedge\emptyset$ is valid and can play the role of `true`, and $\bigvee\emptyset$ is invalid and can play the role of `false`.

Let a statement φ be given. We let $\mathrm{fv}(\varphi)$ denote the set of free variables of φ. If $\mathrm{fv}(\varphi) = \varnothing$ then φ is said to be *closed*. Let e be a statement or a term. Given $n \in \mathbb{N}$, pairwise distinct variables $x_1, \ldots, x_n$ and closed terms $t_1, \ldots, t_n$, we write $e[t_1/x_1, \ldots, t_n/x_n]$ for the result of substituting simultaneously in e all free occurrences of $x_1, \ldots, x_n$ by $t_1, \ldots, t_n$, respectively. Let e and e' be two statements or terms. We say that e' is a closed instance of e iff there exists $n \in \mathbb{N}$, pairwise distinct variables $x_1, \ldots, x_n$ and closed terms $t_1, \ldots,$

[1] Despite the notation $t \neq t'$, we we still consider that $t \neq t'$ is of the form $\neg\psi$.

t_n such that e' is $e[t_1/x_1, \ldots, t_n/x_n]$; if e' is known to be closed then we say "instance of e" rather than "closed instance of e." Given $n \in \mathbb{N}$ and terms t_1, t'_1, $\ldots$, t_n, t'_n, we say that $(t'_1, \ldots, t'_n)$ is a closed instance of $(t_1, \ldots, t_n)$ iff for all members i of $\{1, \ldots, n\}$, t'_i is a closed instance of t_i; when t'_1, $\ldots$, t'_n are known to be closed then we say "instance of $(t_1, \ldots, t_n)$" rather than "closed instance of $(t_1, \ldots, t_n)$."

Though negation can be applied only to atoms and identities, we need to be able to semantically negate a statement in a syntactically friendly manner which is achieved in the following usual way: given a statement φ, $\sim\varphi$ denotes

- $\neg\varphi$ if φ is an atom;
- ψ if φ is of the form $\neg\psi$;
- $t \neq t'$ if φ is of the form $t \doteq t'$;
- $t \doteq t'$ if φ is of the form $t \neq t'$;
- $\bigwedge\{\sim\psi \mid \psi \in X\}$ if φ is of the form $\bigvee X$;
- $\bigvee\{\sim\psi \mid \psi \in X\}$ if φ is of the form $\bigwedge X$;
- $\forall x \sim\psi$ if φ is of the form $\exists x\, \psi$;
- $\exists x \sim\psi$ if φ is of the form $\forall x\, \psi$.

Given a set X of statements, we let $\sim X$ denote $\{\sim\varphi \mid \varphi \in X\}$. A set X of literals is said to be *saturated* just in case every closed atom is an instance of a member of at least one of the sets X and $\sim X$. A set of literals is said to be *complete* just in case it is saturated and consistent.

Given $n \in \mathbb{N}$ and statements φ_1, $\ldots$, φ_n, we use $\varphi_1 \vee \cdots \vee \varphi_n$ and $\varphi_1 \wedge \cdots \wedge \varphi_n$ as abbreviations for $\bigvee\{\varphi_i \mid 1 \leq i \leq n\}$ and $\bigwedge\{\varphi_i \mid 1 \leq i \leq n\}$, respectively. Also, given two statements φ_1 and φ_2, $\varphi_1 \rightarrow \varphi_2$ is an abbreviation for $\sim\varphi_1 \vee \varphi_2$ and $\varphi_1 \leftrightarrow \varphi_2$ is an abbreviation for $(\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)$. Note that $\rightarrow$ is a logical symbol whereas $\leftarrow$ is not: $\leftarrow$ separates the body of a rule from its head, and does not receive a logical meaning in an operational semantics.

The substatements of a statement φ of the form $\bigvee X$ or $\bigwedge X$ are φ and the substatements of the members of X. The substatements of a statement φ of the form $\exists x\, \psi$ or $\forall x\, \psi$ are φ and the substatements of ψ. The substatements of a statement of the form $\neg\psi$ are $\neg\psi$ and ψ. An atom or identity is its unique substatement.

Let a statement φ be given. Let T be the parse tree of φ where the nodes are labeled with one of $\bigvee$, $\bigwedge$, $\exists x$ for some variable x, or $\forall x$ for some variable x, so that the leaves of T are all (intuitive) occurrences of literals, identities and distinctions in φ. Then a (formal) occurrence of a literal in φ can be defined as the set of all formulas that appear on the branch of T whose leaf is that (intuitive) occurrence of literal.

Definition 6. Given a statement φ and a literal ψ, an *occurrence of ψ in φ* is defined as a $\subseteq$-minimal set O of statements such that:

- $\varphi \in O$ and ψ is the only literal in O;
- for all members of O of the form $\bigvee X$ or $\bigwedge X$, O contains a member of X;
- for all members of O of the form $\exists x\, \xi$ or $\forall x\, \xi$, O contains ξ.

Example 7. Suppose that $\mathcal{V}$ contains 3 nullary predicate symbols p, q and r. Let φ denote $\bigwedge\{\neg p, \bigvee\{q, r, \neg p\}\}$. The occurrences of literals in φ are:

- $\{\varphi, \neg p\}$—an occurrence of $\neg p$ in φ;
- $\{\varphi, \bigvee\{q, r, \neg p\}, q\}$—an occurrence of q in φ;
- $\{\varphi, \bigvee\{q, r, \neg p\}, r\}$—an occurrence of r in φ;
- $\{\varphi, \bigvee\{q, r, \neg p\}, \neg p\}$—an occurrence of $\neg p$ in φ.

2.2 Semantics

Definition 8. Let a consistent set S of closed literals be given. For all closed statements φ, we inductively define the notion S *forces* φ, denoted $S \Vdash \varphi$, as follows.

- For all closed terms t_1 and t_2, $S \Vdash t_1 \doteq t_2$ in case t_1 and t_2 are identical, and $S \Vdash t_1 \neq t_2$ in case t_1 and t_2 are distinct.
- For all closed literals φ, $S \Vdash \varphi$ iff $\varphi \in S$.
- For all countable sets X of closed statements, $S \Vdash \bigvee X$ iff S forces some member of X, and $S \Vdash \bigwedge X$ iff S forces all members of X.
- For all statements φ and variables x with $\mathrm{fv}(\varphi) = \{x\}$, $S \Vdash \exists x\, \varphi$ iff $S \Vdash \varphi[t/x]$ for some closed term t, and $S \Vdash \forall x\, \varphi$ iff $S \Vdash \varphi[t/x]$ for all closed terms t.

Given a set S of literals and a set T of statements, we say that S *forces* T, denoted $S \Vdash T$, just in case either S is inconsistent or the set of all closed instances of all members of S forces all closed instances of all members of T.

Definition 9. A *standard structure* (*over* $\mathcal{V}$) is a set of closed atoms.

Note the following particular cases.

- If $\mathcal{V}$ contains at least one constant and no nullary predicate symbol then a standard structure over $\mathcal{V}$ is basically a Herbrand interpretation.
- If $\mathcal{V}$ contains nullary predicate symbols only then a standard structure is basically a propositional interpretation.

Definition 10. Let a standard structure $\mathfrak{M}$ be given. Let X be the complete set of closed literals such that for all closed atoms φ, $\varphi \in X$ iff $\varphi \in \mathfrak{M}$. For all statements φ, we say that φ *is true in* $\mathfrak{M}$, or that $\mathfrak{M}$ *is a model of* φ, iff $X \Vdash \varphi$.

Notation 11. Let a standard structure $\mathfrak{M}$ be given. Given a statement φ, we write $\mathfrak{M} \vDash \varphi$ if $\mathfrak{M}$ is a model of φ, and $\mathfrak{M} \nvDash \varphi$ otherwise. Given a set T of statements, we write $\mathfrak{M} \vDash T$ if $\mathfrak{M}$ is a model of all members of T, and $\mathfrak{M} \nvDash T$ otherwise.

Notation 12. We denote by $\mathcal{W}$ the set of all standard structures (over $\mathcal{V}$).

Given a set T of statements and a statement φ, we write $T \vDash_{\mathcal{W}} \varphi$ if every standard model of T is a model of φ; if $T \vDash_{\mathcal{W}} \varphi$ then we say that T *logically implies* φ *in* $\mathcal{W}$ or that φ *is a logical consequence of* T *in* $\mathcal{W}$. The same notation and terminology applies to sets of statements instead of statements. Two statements φ and ψ are said to be *logically equivalent in* $\mathcal{W}$ iff they have the same models in $\mathcal{W}$.

3 Denotational semantics of formal logic programs

3.1 Formal logic programs

The notions introduced in the previous section might suggest that we are considering a notion of logical consequence, namely $\vDash_{\mathcal{W}}$, that, because of its focus on standard structures, is stronger than the classical notion of logical consequence. Now conceive of $\mathcal{V}$ as the vocabulary used to *describe* a structure, to express what a structure is "made of," but not necessarily the vocabulary used to *talk about* a structure, to express properties of a structure. We assume that the vocabulary used to talk about a structure is no more expressive, and is possibly less expressive, than the vocabulary used to describe a structure.

Notation 13. We denote by $\mathcal{V}^\star$ a countable subset of $\mathcal{V}$.

We think of $\mathcal{V}^\star$ as the vocabulary used to talk about a structure: theories, axioms, theorems must consist of statements over $\mathcal{V}^\star$. Suppose that infinitely many closed terms are not terms over $\mathcal{V}^\star$, either because $\mathcal{V}$ contains infinitely many constants not in $\mathcal{V}^\star$, or because $\mathcal{V}$ contains at least one constant and contains at least one function symbol of arity one or more that is not in $\mathcal{V}^\star$. Then for all sets T of statements over $\mathcal{V}^\star$ and for all statements φ over $\mathcal{V}^\star$, $T \vDash_{\mathcal{W}} \varphi$ iff $T \vDash \varphi$. In other words, if countably many closed terms are "unspeakable of" then $\vDash_{\mathcal{W}}$, with sets of statements that can be "spoken out" on the left hand side and with statements that can be "spoken out" on the right hand side, is equivalent to the classical notion of logical consequence [14]. This means that by choosing $\mathcal{V}$ to be countable and by setting $\mathcal{V}^\star$ to $\mathcal{V}$, one opts for a semantics based on Herbrand structures, but by setting $\mathcal{V}^\star$ to a strict subset of $\mathcal{V}$ that makes countably many closed terms "unspeakable of," then one opts for a semantics equivalent to the classical notion of logical consequence defined on the basis of all structures.

Most of the work done in logic programming is developed on the basis of the class of Herbrand structures. But there are exceptions, for instance, the semantics of definite logic programs and queries can be based on either Herbrand structures

or all structures: given a definite logic program T and a definite query Q, Prolog returns a computed answer substitution θ iff the universal closures of $Q\theta$ are true in all Herbrand models of T, or equivalently, are true in the minimal Herbrand model of T, or equivalently, are true in all models of T [12]. So it is sometimes desirable not to be tied to a semantics based on Herbrand structures. Moreover, we will see that such a restriction is not conceptually necessary: we will never have to assume that $\mathcal{V}^\star$ is equal to $\mathcal{V}$ in any definition of concept—but we will sometimes have to assume that $\mathcal{V}^\star$ is equal to $\mathcal{V}$ in the statements of some propositions. So we are going to define a notion of logic program as a set of rules built from $\mathcal{V}^\star$, not from $\mathcal{V}$.

Notation 14. We denote by $\mathrm{Prd}(\mathcal{V}^\star)$ the set of predicate symbols in $\mathcal{V}^\star$. For all $n \in \mathbb{N}$, we denote by $\mathrm{Prd}(\mathcal{V}^\star, n)$ the set of members of $\mathrm{Prd}(\mathcal{V}^\star)$ of arity n.

We want to consider sets of rules whose heads are literals and whose bodies are arbitrary. Since statements can be infinitary and can contain occurrences of $\doteq$, we just have to provide, for every $n \in \mathbb{N}$ and $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$, two rules: one whose head is $\wp(v_1, \ldots, v_n)$ (which is nothing but $\wp$ if $n = 0$), and one whose head is $\neg\wp(v_1, \ldots, v_n)$ (which is nothing but $\neg\wp$ if $n = 0$). For instance,

$$\{p(\overline{2n}) \leftarrow q(\overline{2n+1}) \mid n \in \mathbb{N}\}$$

can be represented as

$$p(v_1) \leftarrow \bigvee\{q(s(v_1)) \wedge v_1 \doteq \overline{2n} \mid n \in \mathbb{N}\}.$$

Definition 15. We define a *formal logic program* (*over* $\mathcal{V}^\star$) as a $\mathrm{Prd}(\mathcal{V}^\star)$-family of pairs of statements over $\mathcal{V}^\star$, say $\big((\varphi^+_\wp, \varphi^-_\wp)\big)_{\wp \in \mathrm{Prd}(\mathcal{V}^\star)}$, such that for all $n \in \mathbb{N}$ and $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$, $\mathrm{fv}(\varphi^+_\wp) \cup \mathrm{fv}(\varphi^-_\wp) \subseteq \{v_1, \ldots, v_n\}$.

The condition on variables is at no loss of generality and is imposed so as to simplify subsequent notation, and is also often used in the literature; it just states that a variable that occurs free in the body of a rule occurs in the head of that rule. Note that if the set of predicate symbols in $\mathcal{V}^\star$ is finite then finite sets of rules over $\mathcal{V}^\star$ are naturally translated into finite formal logic programs.

Example 16. Suppose that $\mathcal{V}^\star$ consists of a constant $\overline{0}$, a unary function symbol s, 5 nullary predicate symbols $q_1, \ldots, q_5$, and 4 unary predicate symbols $p_1, \ldots,$

p_4. An example of formal logic program is given by the following statements.

$$\varphi^+_{p_1} \equiv v_1 \doteq \overline{0} \vee \exists v_0\big(v_1 \doteq s(s(v_0)) \wedge p_1(v_0)\big)$$
$$\varphi^-_{p_1} \equiv v_1 \not\doteq \overline{0} \wedge \forall v_0\big(v_1 \not\doteq s(s(v_0)) \vee \neg p_1(v_0)\big)$$

$$\varphi^+_{p_2} \equiv v_1 \doteq \overline{0} \vee \exists v_0\big(v_1 \doteq s(s(v_0)) \wedge p_2(v_0)\big)$$
$$\varphi^-_{p_2} \equiv v_1 \doteq s(\overline{0}) \vee \exists v_0\big(v_1 \doteq s(s(v_0)) \wedge \neg p_2(v_0)\big)$$

$$\varphi^+_{p_3} \equiv v_1 \doteq \overline{0} \vee \exists v_0\big(v_1 \doteq s(v_0) \wedge \neg p_3(v_0)\big)$$
$$\varphi^-_{p_3} \equiv \exists v_1\big(v_1 \doteq s(v_0) \wedge p_3(v_0)\big)$$

$$\varphi^+_{p_4} \equiv p_4(s(s(v_1))) \qquad \varphi^+_{q_1} \equiv \bigwedge \emptyset \qquad \varphi^+_{q_2} \equiv q_3$$
$$\varphi^-_{p_4} \equiv \neg p_4(s(s(v_1))) \qquad \varphi^-_{q_1} \equiv \bigvee \emptyset \qquad \varphi^-_{q_2} \equiv \neg q_3$$

$$\varphi^+_{q_3} \equiv q_2 \qquad \varphi^+_{q_4} \equiv q_4 \qquad \varphi^+_{q_5} \equiv \neg q_5$$
$$\varphi^-_{q_3} \equiv \neg q_2 \qquad \varphi^-_{q_4} \equiv \bigvee \emptyset \qquad \varphi^-_{q_5} \equiv \bigvee \emptyset$$

As $\mathcal{V}^\star$ contains both nullary and nonnullary predicate symbols, Example 16 describes a "hybrid" formal logic program, though of a simple kind as it has a purely first-order part and a purely propositional part. Let us take advantage of this example to illustrate how Definition 15 is put to use to represent rules. Recall that $\bigwedge \emptyset$ is valid and $\bigvee \emptyset$ is invalid. For the propositional rules,

- $\varphi^+_{q_1}$ and $\varphi^-_{q_1}$ represent the fact q_1,
- $\varphi^+_{q_2}$ and $\varphi^-_{q_2}$ represent the rules $q_2 \leftarrow q_3$ and $\neg q_2 \leftarrow \neg q_3$,
- $\varphi^+_{q_3}$ and $\varphi^-_{q_3}$ represent the rules $q_3 \leftarrow q_2$ and $\neg q_3 \leftarrow \neg q_2$,
- $\varphi^+_{q_4}$ and $\varphi^-_{q_4}$ represent the rule $q_4 \leftarrow q_4$, and
- $\varphi^+_{q_5}$ and $\varphi^-_{q_5}$ represent the rule $q_5 \leftarrow \neg q_5$.

Let us now comment on the first-order rules. The statements $\varphi^+_{p_i}$, $i \in \{1, 2\}$, represent the rule

$$p_i(v_1) \leftarrow v_1 \doteq \overline{0} \vee \exists v_0\big(v_1 \doteq s(s(v_0)) \wedge p_i(v_0)\big)$$

which could be rewritten as the following two rules.

$$p_i(\overline{0})$$
$$p_i(s(s(v_1))) \leftarrow p_i(v_1)$$

So $\varphi^+_{p_i}$, with i equal to either 1 or 2, allows one to generate all literals of the form $p_i(\overline{2n})$, $n \in \mathbb{N}$. Obviously $\varphi^-_{p_1}$ and $\varphi^-_{p_2}$ are logically equivalent in $\mathcal{W}$, and $\varphi^-_{p_i}$, with i equal to either 1 or 2, allows one to generate all literals of the form $\neg p_i(\overline{2n+1})$, $n \in \mathbb{N}$. More precisely, the rule $\neg p_1(v_1) \leftarrow \varphi^-_{p_1}$, namely

$$\neg p_1(v_1) \leftarrow v_1 \not\doteq \overline{0} \wedge \forall v_0\big(v_1 \not\doteq s(s(v_0)) \vee \neg p_1(v_0)\big)$$

could be naturally implemented from $\{p_1(\overline{0}), p_1(s(s(v_1))) \leftarrow p_1(v_1)\}$ using negation as finite failure, and its syntax is naturally related to Cark's completion of the set $\{p_1(\overline{0}),\ p_1(s(s(v_1))) \leftarrow p_1(v_1)\}$. The rule $\neg p_2(v_1) \leftarrow \varphi^-_{p_2}$, namely

$$\neg p_2(v_1) \leftarrow v_1 \doteq s(\overline{0}) \vee \exists v_0\big(v_1 \doteq s(s(v_0)) \wedge \neg p_2(v_0)\big)$$

is the dual of the rule $p_2(v_1) \leftarrow \varphi^+_{p_2}$, and could be rewritten

$$\begin{aligned}&\neg p_2(s(\overline{0}))\\ &\neg p_2(s(s(v_1))) \leftarrow \neg p_2(v_1)\end{aligned}$$

to generate $\{\neg p_2(\overline{2n+1}) \mid n \in \mathbb{N}\}$ similarly to the way $\{p_2(\overline{2n}) \mid n \in \mathbb{N}\}$ would be generated using $p_2(v_1) \leftarrow \varphi^+_{p_2}$. The statements $\varphi^+_{p_3}$ and $\varphi^-_{p_3}$ offer a third way of generating the set of even numbers and its complement, with both the positive rule $p_3(v_1) \leftarrow \varphi^+_{p_3}$ and the negative rule $\neg p_3(v_1) \leftarrow \varphi^-_{p_3}$ being used alternatively, starting from the positive rule. Finally, the statements $\varphi^+_{p_4}$ and $\varphi^-_{p_4}$ represent the rules

$$\begin{aligned}p_4(v_1) &\leftarrow p_4(s(s(v_1)))\\ \neg p_4(v_1) &\leftarrow \neg p_4(s(s(v_1)))\end{aligned}$$

and would not generate any literal.

Equality can be one of the predicate symbols in $\mathcal{V}^\star$, and its intended interpretation captured by any logic program $\big((\varphi^+_\wp, \varphi^-_\wp)\big)_{\wp \in \mathrm{Prd}(\mathcal{V}^\star)}$ such that $\varphi^+_=$ is of the form $\bigvee X$ where X is a superset of

$$\begin{aligned}&\Big\{v_1 \doteq v_2,\ v_2 = v_1,\ \exists v_0(v_1 = v_0 \wedge v_0 = v_2)\Big\} \cup\\ &\qquad\Big\{\exists v_3 \ldots \exists v_{3+2n-1}\big(v_3 = v_{3+n} \wedge \cdots \wedge v_{3+n-1} = v_{3+2n-1} \wedge\\ &\qquad v_1 \doteq f(v_3, \ldots, v_{3+n-1}) \wedge v_2 \doteq f(v_{3+n}, \ldots, v_{3+2n-1})\big) \ \Big|\\ &\qquad\qquad n \in \mathbb{N},\ f \text{ is an } n\text{-ary function symbol in } \mathcal{V}^\star\Big\},\end{aligned}$$

$\varphi^-_=$ is of the form $\bigvee X$ where X contains

$$\exists v_0\big((v_0 = v_1 \wedge v_0 \neq v_2) \vee (v_0 \neq v_1 \wedge v_0 = v_2)\big)$$

and for all $n \in \mathbb{N}$ and $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$, $\varphi^+_\wp$ is of the form $\bigvee X$ where X contains

$$\exists v_{n+1} \ldots \exists v_{2n}\big(v_1 = v_{n+1} \wedge \cdots \wedge v_n = v_{2n} \wedge \wp(v_{n+1}, \ldots, v_{2n})\big)$$

and $\varphi^-_\wp$ is of the form $\bigvee X$ where X contains

$$\exists v_{n+1} \ldots \exists v_{2n}\big(v_1 = v_{n+1} \wedge \cdots \wedge v_n = v_{2n} \wedge \neg\wp(v_{n+1}, \ldots, v_{2n})\big).$$

So we distinguish between identity and equality. Identity is treated as a logical symbol, equality as a nonlogical symbol. Identity is a key notion in logic programming as it is at the heart of the unification algorithm, and the usual approach is to treat identity and equality as equivalent, with the restriction to

the class of Herbrand interpretations as a justification for the identification of both notions. Our approach consists in logically defining identity from $\mathcal{V}$, the vocabulary used to describe a structure, and in axiomatizing equality from $\mathcal{V}^\star$, the vocabulary used to talk about a structure. With this approach, equality and Herbrand structures are not in a state of mutual dependency: if infinitely many closed terms are "unspeakable of" then equality as axiomatized above behaves equivalently to the way it behaves w.r.t. the classical notion of logical consequence.

Definition 17. Given a formal logic program $\mathcal{P} = \big((\varphi^+_\wp, \varphi^-_\wp)\big)_{\wp\in \mathrm{Prd}(\mathcal{V}^\star)}$, the *classical logical form of* $\mathcal{P}$ is defined as

$$\big\{\varphi^+_\wp \to \wp(v_1, \ldots, v_n),\, \varphi^-_\wp \to \neg\wp(v_1, \ldots, v_n) \;\big|\; n \in \mathbb{N},\, \wp \in \mathrm{Prd}(\mathcal{V}^\star, n)\big\}.$$

Of course, $\vDash_{\mathcal{W}}$ applied to the classical logical form of a formal logic program $\mathcal{P}$ does not adequately capture the logical meaning of $\mathcal{P}$. An appropriate logical reading of a formal logic program, which amounts to an appropriate denotational semantics, requires more than reading the arrow that links the left hand side and right hand side of a rule as a logical implication: it requires the explicit use of a modal operator of necessity to capture the notion of derivability, or provability, in the style of epistemic logic [16, 13]. We will complete this task in another paper in a more general setting, with a generalization of formal logic programs to sets of rules with arbitrary bodies and heads.

Notation 18. Given a formal logic program $\mathcal{P}$, we let $\mathrm{Clf}(\mathcal{P})$ denote the classical logical form of $\mathcal{P}$.

We have indicated that the general logic programs that are the object of Kripke-Kleene semantics, the well founded semantics and the stable model semantics will be seen as a particular case of formal logic programs where the negative rules are fully determined by the positive rules and can be left implicit; they are in one-to-one correspondence with the formal logic programs defined next.

Definition 19. Let a formal logic program $\mathcal{P} = \big((\varphi^+_\wp, \varphi^-_\wp)\big)_{\wp\in \mathrm{Prd}(\mathcal{V}^\star)}$ be given. We say that $\mathcal{P}$ is *symmetric* iff for all $\wp \in \mathrm{Prd}(\mathcal{V}^\star)$, $\varphi^-_\wp = {\sim}\varphi^+_\wp$.

The next definition introduces a notion that is a key property of symmetric formal logic programs.

Definition 20. We say that a formal logic program $\big((\varphi^+_\wp, \varphi^-_\wp)\big)_{\wp\in \mathrm{Prd}(\mathcal{V}^\star)}$ is *locally consistent* iff for all $\wp \in \mathrm{Prd}(\mathcal{V}^\star)$, no closed instance of $\varphi^+_\wp \wedge \varphi^-_\wp$ has a model in $\mathcal{W}$.

Property 21. *A symmetric formal logic program is locally consistent.*

3.2 Generated literals

The mechanistic view on (the rules of) a formal logic program $\mathcal{P}$ presented in Section 1.2 allows one to talk about the literals over $\mathcal{V}^\star$ generated by $\mathcal{P}$; these literals make up a set that we denote by $[\,\mathcal{P}\,]$, and that can be described as follows.

Notation 22. Let a formal logic program $\mathcal{P} = \big((\varphi^+_\wp, \varphi^-_\wp)\big)_{\wp \in \mathrm{Prd}(\mathcal{V}^\star)}$ be given. We denote by $[\,\mathcal{P}\,]$ the (unique) $\subseteq$-minimal set of literals such that for all $n \in \mathbb{N}$, $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$ and terms $t_1, \ldots, t_n$ over $\mathcal{V}^\star$, the following holds.

- $\wp(t_1, \ldots, t_n) \in [\,\mathcal{P}\,]$ iff for all closed terms $t'_1, \ldots, t'_n$, if $(t'_1, \ldots, t'_n)$ is an instance of $(t_1, \ldots, t_n)$ then $[\,\mathcal{P}\,] \Vdash \varphi^+_\wp[t'_1/v_1, \ldots, t'_n/v_n]$, and
- $\neg\wp(t_1, \ldots, t_n) \in [\,\mathcal{P}\,]$ iff for all closed terms $t'_1, \ldots, t'_n$, if $(t'_1, \ldots, t'_n)$ is an instance of $(t_1, \ldots, t_n)$ then $[\,\mathcal{P}\,] \Vdash \varphi^-_\wp[t'_1/v_1, \ldots, t'_n/v_n]$.

Let a formal logic program $\mathcal{P} = \big((\varphi^+_\wp, \varphi^-_\wp)\big)_{\wp \in \mathrm{Prd}(\mathcal{V}^\star)}$, $n \in \mathbb{N}$, $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$ and terms $t_1, \ldots, t_n$ over $\mathcal{V}^\star$ be given. In case none of the variables that occur in one of $t_1, \ldots, t_n$ is captured by quantifiers in $\varphi^+_\wp$ when simultaneously substituting $v_1, \ldots, v_n$ in $\varphi^+_\wp$ by $t_1, \ldots, t_n$, respectively, then it is safe to write $\varphi^+_\wp[t_1/v_1, \ldots, t_n/v_n]$ and the first clause in Notation 22 can be simplified as: $\wp(t_1, \ldots, t_n) \in [\,\mathcal{P}\,]$ iff $[\,\mathcal{P}\,] \Vdash \varphi^+_\wp[t_1/v_1, \ldots, t_n/v_n]$ (and the second clause can be similarly simplified under the corresponding assumption about $\varphi^-_\wp$). We avoid the issues of captured variables in substitutions by restricting the definition and notation of substitution of variables by terms to the definition and notation of substitution of variables by closed terms.

Example 23. If $\mathcal{P}$ is the formal logic program of Example 16 then

$$[\,\mathcal{P}\,] = \big\{p_i(\overline{2n}),\, \neg p_i(\overline{2n+1}) \mid i \in \{1, 2, 3\},\, n \in \mathbb{N}\big\} \cup \{q_1\}.$$

It is easy to verify that the set of literals over $\mathcal{V}^\star$ generated by a formal logic program is closed under forcing.

Property 24. *For all formal logic programs $\mathcal{P}$ and literals ψ over $\mathcal{V}^\star$,*

$$[\,\mathcal{P}\,] \Vdash \psi \text{ iff } \psi \in [\,\mathcal{P}\,].$$

The notion of local consistency introduced in Definition 20 will play a pivotal role in the statements of some propositions, but the more general notion of consistency tout court defined next is the real counterpart to the classical concept of a consistent theory.

Definition 25. A formal logic program $\mathcal{P}$ is said to be *consistent* just in case $[\,\mathcal{P}\,]$ is consistent.

Property 26. *Every locally consistent formal logic program is consistent.*

Let a formal logic program $\mathcal{P} = \big((\varphi^+_\wp, \varphi^-_\wp)\big)_{\wp \in \mathrm{Prd}(\mathcal{V}^\star)}$ be given. When $\mathcal{V}^\star = \mathcal{V}$, the definition of $[\,\mathcal{P}\,]$ can involve closed literals only—a consequence of Property 24 and the next property. In the general case, $[\,\mathcal{P}\,]$ is a set of possibly nonclosed literals, and some rules might fire because their bodies are activated thanks to (a universal closure of) such literals. For instance, assume that $\mathcal{V}^\star$ contains a unary predicate symbol p and a nullary predicate symbol q, $\varphi^+_p = \bigwedge \varnothing$, and $\varphi^+_q = \forall v_0\, p(v_0)$. Then $[\,\mathcal{P}\,]$ contains $p(v_1)$, hence it contains q. Also, $[\,\mathcal{P}\,]$ contains $p(t)$ for all terms t over $\mathcal{V}^\star$, hence in particular for all closed terms t

over $\mathcal{V}^\star$. Still, if $\mathcal{V}$ contains at least one constant not in $\mathcal{V}^\star$, then the set of all closed members of $[\,\mathcal{P}\,]$ of the form $p(t)$ (with t a closed term over $\mathcal{V}^\star$) does not force $\forall v_0\, p(v_0)$, which shows that the next property would not hold if the assumption $\mathcal{V}^\star = \mathcal{V}$ was dropped.

Property 27. *Suppose that $\mathcal{V}^\star = \mathcal{V}$. Let a formal logic program $\mathcal{P}$ be given. Write $\mathcal{P}$ as $\big((\varphi^+_\wp,\, \varphi^-_\wp)\big)_{\wp\in\mathrm{Prd}(\mathcal{V}^\star)}$. Then the set of closed members of $[\,\mathcal{P}\,]$ is the unique $\subseteq$-minimal set of literals such that for all $n \in \mathbb{N}$, $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$ and closed terms $t_1, \ldots, t_n$,*

- $\wp(t_1, \ldots, t_n) \in [\,\mathcal{P}\,]$ *iff* $[\,\mathcal{P}\,] \Vdash \varphi^+_\wp[t_1/v_1, \ldots, t_n/v_n]$, *and*
- $\neg\wp(t_1, \ldots, t_n) \in [\,\mathcal{P}\,]$ *iff* $[\,\mathcal{P}\,] \Vdash \varphi^-_\wp[t_1/v_1, \ldots, t_n/v_n]$.

Even though the classical logical form of a formal logic program $\mathcal{P}$, formalized in Definition 17, does not capture in a satisfactory way the logical meaning of $\mathcal{P}$, it is still well behaved, in the sense of the two properties that follow.

Property 28. *For all formal logic programs $\mathcal{P}$,* $\mathrm{Clf}(\mathcal{P}) \vDash_{\mathcal{W}} [\,\mathcal{P}\,]$.

Corollary 29. *For all formal logic programs $\mathcal{P}$, if $[\,\mathcal{P}\,]$ is complete then the set of closed instances of atoms in $[\,\mathcal{P}\,]$ is a model of* $\mathrm{Clf}(\mathcal{P})$.

Let a formal logic program $\mathcal{P}$ and $\psi \in [\,\mathcal{P}\,]$ be given. What are the subsets X of $[\,\mathcal{P}\,]$ that allow ψ to be generated by successively firing rules, starting with rules whose body can be unconditionally activated (such as $\bigwedge \varnothing$), till enough literals have been generated and put into X so that there exists a rule in $\mathcal{P}$ of the form $\chi \leftarrow \xi$ and a substitution θ such that ψ is $\chi\theta$ and for all closed substitutions θ', $\xi(\theta\theta')$ can be activated thanks to the literals in X? Every such subset of $[\,\mathcal{P}\,]$ is one of the members of the set $\mathcal{P}[\psi]$ defined next, but $\mathcal{P}[\psi]$ is more inclusive as it contains the sets of literals that can be collected along a top-down attempt to derive ψ, even if that attempt is unsuccessful due to infinite descending chains or cycles. For instance, assume that $\mathcal{V}^\star$ contains $\overline{0}$, the unary function symbol s, a unary predicate predicate p and a nullary predicate symbol q, $\varphi^+_p = p(s(v_1))$, and $\varphi^+_q = q$. Then the notation that follows defines $\mathcal{P}[p(v_1)]$ as a set of sets of literals that contains $\{p(s^n(v_1)) \mid n > 0\}$, and $\mathcal{P}[q]$ as a set of sets of literals that contains $\{q\}$.

Notation 30. Let a formal logic program $\mathcal{P}$ be given. Let S be the set of literals over $\mathcal{V}^\star$. Let ρ be the (unique) function from S into the set of subsets of S such that for all $n \in \mathbb{N}$, $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$ and terms $t_1, \ldots, t_n$ over $\mathcal{V}^\star$, the following holds.

- $\rho\big(\wp(t_1, \ldots, t_n)\big)$ consists of all sets of the form

  $$\{\rho(\psi) \cup \{\psi\} \mid \psi \in Y\}$$

  where Y is a subset of S such that for all closed terms $t'_1, \ldots, t'_n$, if $(t'_1, \ldots, t'_n)$ is an instance of $(t_1, \ldots, t_n)$ then Y forces $\varphi^+_\wp[t'_1/v_1, \ldots, t'_n/v_n]$.
- $\rho\big(\neg\wp(t_1, \ldots, t_n)\big)$ consists of all sets of the form

  $$\{\rho(\psi) \cup \{\psi\} \mid \psi \in Y\}$$

where Y is a subset of S such that for all closed terms $t'_1, \ldots, t'_n$, if $(t'_1, \ldots, t'_n)$ is an instance of $(t_1, \ldots, t_n)$ then Y forces $\varphi^-_\wp[t'_1/v_1, \ldots, t'_n/v_n]$.

Then for all literals ψ over $\mathcal{V}^\star$, we let $\mathcal{P}[\psi]$ denote $\rho(\psi)$.

Let us rephrase the definition of $[\,\mathcal{P}\,]$ and list the set of literals that can be generated from $\mathcal{P}$ at the αth round of activations of closed instances of the bodies of $\mathcal{P}$'s rules.

Notation 31. Let a formal logic program $\mathcal{P} = \big((\varphi^+_\wp, \varphi^-_\wp)\big)_{\wp \in \mathrm{Prd}(\mathcal{V}^\star)}$ be given. Inductively define a sequence $([\,\mathcal{P}\,]_\alpha)_{\alpha \in \mathrm{Ord}}$ of sets of literals over $\mathcal{V}^\star$ as follows. Let an ordinal α be given and assume that for all $\beta < \alpha$, $[\,\mathcal{P}\,]_\beta$ has been defined. Then denote by $[\,\mathcal{P}\,]_\alpha$ the set of all literals over $\mathcal{V}^\star$ such that for all $n \in \mathbb{N}$, $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$ and terms $t_1, \ldots, t_n$ over $\mathcal{V}^\star$, the following holds.

- $\wp(t_1, \ldots, t_n) \in [\,\mathcal{P}\,]_\alpha$ iff for all closed terms $t'_1, \ldots, t'_n$, if $(t'_1, \ldots, t'_n)$ is an instance of $(t_1, \ldots, t_n)$ then $\bigcup_{\beta<\alpha}[\,\mathcal{P}\,]_\beta \Vdash \varphi^+_\wp[t'_1/v_1, \ldots, t'_n/v_n]$.
- $\neg\wp(t_1, \ldots, t_n) \in [\,\mathcal{P}\,]_\alpha$ iff for all closed terms $t'_1, \ldots, t'_n$, if $(t'_1, \ldots, t'_n)$ is an instance of $(t_1, \ldots, t_n)$ then $\bigcup_{\beta<\alpha}[\,\mathcal{P}\,]_\beta \Vdash \varphi^-_\wp[t'_1/v_1, \ldots, t'_n/v_n]$.

With Notation 31 in hand, we can then characterize the members of $\mathcal{P}[\psi]$ that collect all the literals in a successful generation of ψ from $\mathcal{P}$.

Property 32. *Let a formal logic program $\mathcal{P}$ be given. Then $[\,\mathcal{P}\,]$ is equal to $\bigcup_{\alpha\in\mathrm{Ord}}[\,\mathcal{P}\,]_\alpha$. Moreover, for all ordinals α and literals ψ over $\mathcal{V}^\star$, $\psi \in [\,\mathcal{P}\,]_\alpha$ iff there exists a subset X of $\bigcup_{\beta<\alpha}[\,\mathcal{P}\,]_\beta$ with $X \in \mathcal{P}[\psi]$.*

As can be expected, the members of $\mathcal{P}[\psi]$ that attest of unsuccessful attempts at deriving ψ from $\mathcal{P}$ due to infinite descending chains or cycles will play a special role in particular transformations of $\mathcal{P}$ related to the well founded semantics.

3.3 Characterization of Kripke-Kleene semantics

Definition 33. A *partial interpretation* (*over* $\mathcal{V}$) is a consistent set of closed literals.

Kripke-Kleene semantics is usually presented in a 3-valued logical setting. The relationship between Definition 33 and a 3-valued logical setting is the following. Let M be a partial interpretation, and let a closed atom φ be given. Then the truth value of φ in M can be set to `true` is $\varphi \in M$, to `false` if $\neg\varphi \in M$, and to a third value or to "undefined" otherwise. Definition 34 then generalizes the notion of a partial model of a general logic program—that as we have pointed out, can be seen as a symmetric formal logic program whose negative rules have not been explicitly written.

Definition 34. Let a logic program $\mathcal{P} = \big((\varphi^+_\wp, \varphi^-_\wp)\big)_{\wp \in \mathrm{Prd}(\mathcal{V}^\star)}$ be given. A *partial model of* $\mathcal{P}$ is a partial interpretation M such that for all $n \in \mathbb{N}$, members $\wp$ of $\mathrm{Prd}(\mathcal{V}^\star, n)$ and closed terms $t_1, \ldots, t_n$, M contains $\wp(t_1, \ldots, t_n)$ or $\neg\wp(t_1, \ldots, t_n)$ iff M forces $\varphi^+_\wp[t_1/v_1, \ldots, t_n/v_n]$ or $\varphi^-_\wp[t_1/v_1, \ldots, t_n/v_n]$, respectively.

Given a formal logic program $\mathcal{P}$, a $\subseteq$-*minimal partial model of* $\mathcal{P}$ is referred to more simply as a minimal partial model of $\mathcal{P}$. Proposition 35 expresses that the generalization of Kripke-Kleene semantics given in Definition 34 is equivalent to the denotational semantics of consistent formal logic programs, provided that $\mathcal{V}^\star$is equal to $\mathcal{V}$, which is the underlying assumption of all frameworks where that semantics is considered. Note that Proposition 35 still applies to more general frameworks as it deals with formal logic programs that might not be symmetric.

Proposition 35. *Assume that* $\mathcal{V}^\star = \mathcal{V}$. *Let a consistent formal logic program* $\mathcal{P}$ *be given. Then* $\mathcal{P}$ *has a unique minimal partial model, which is nothing but the set of closed instances of members of* $[\,\mathcal{P}\,]$.

Proof. Write $\mathcal{P}$ as $\big((\varphi^+_\wp,\, \varphi^-_\wp)\big)_{\wp\in\mathrm{Prd}(\mathcal{V}^\star)}$. Let X denote the set of partial models of $\mathcal{P}$. It is immediately verified that:

- the set of closed instances of members of $\bigcup_{\alpha\in\mathrm{Ord}}[\,\mathcal{P}\,]_\alpha$ is included in $\bigcap X$;
- the set of closed instances of members of $\bigcup_{\alpha\in\mathrm{Ord}}[\,\mathcal{P}\,]_\alpha$ belongs to X.

So $\bigcap X$, being equal to the set of closed instances of members of $\bigcup_{\alpha\in\mathrm{Ord}}[\,\mathcal{P}\,]_\alpha$, is a partial model of $\mathcal{P}$. We conclude with Property 32. □

4 Extensors, and relationships to particular semantics

4.1 Extensors

We now formalize the operation, discussed in Section 1.3, of transforming a formal logic program $\mathcal{P}$ into another formal logic program $\mathcal{P} +_\Omega E$, where Ω selects some occurrences of literals in the bodies of $\mathcal{P}$'s rules and E is a set of literals, the intended meaning of $\mathcal{P} +_\Omega E$ being: "in $\mathcal{P}$, assume E in the contexts indicated by Ω." Definition 36 defines the kind of formal object denoted by Ω. Notation 37 specifies two particular cases the first of which will play a special role in relation to the stable model semantics and the well founded semantics.

Definition 36. Let a formal logic program $\mathcal{P} = \big((\varphi^+_\wp,\, \varphi^-_\wp)\big)_{\wp\in\mathrm{Prd}(\mathcal{V}^\star)}$ be given. A *literal marker for* $\mathcal{P}$ is a sequence of the form $\big((O^+_\wp,\, O^-_\wp)\big)_{\wp\in\mathrm{Prd}(\mathcal{V}^\star)}$ where for all members $\wp$ of $\mathrm{Prd}(\mathcal{V}^\star)$, $O^+_\wp$ and $O^-_\wp$ are sets of occurrences of literals in $\varphi^+_\wp$ and $\varphi^-_\wp$, respectively.

Notation 37. Let a formal logic program $\mathcal{P} = \big((\varphi^+_\wp,\, \varphi^-_\wp)\big)_{\wp\in\mathrm{Prd}(\mathcal{V}^\star)}$ and a literal marker $\Omega = \big((O^+_\wp,\, O^-_\wp)\big)_{\wp\in\mathrm{Prd}(\mathcal{V}^\star)}$ for $\mathcal{P}$ be given.

- If for all $\wp \in \mathrm{Prd}(\mathcal{V}^\star)$, $O^+_\wp$ and $O^-_\wp$ are the sets of all occurrences of negated atoms in $\varphi^+_\wp$ and $\varphi^-_\wp$, respectively, then we denote Ω by $\langle - \rangle_\mathcal{P}$.
- If for all $\wp \in \mathrm{Prd}(\mathcal{V}^\star)$, $O^+_\wp$ and $O^-_\wp$ are the sets of all occurrences of literals in $\varphi^+_\wp$ and $\varphi^-_\wp$, respectively, then we denote Ω by $\langle \pm \rangle_\mathcal{P}$.

In Section1.3, we gave the following introductory example. Assume that $\mathcal{V}^\star$ contains the constant $\overline{0}$, the unary function symbol s and three unary predicate symbols p, q and r. Let $\mathcal{P}$ be a formal logic program, say $\big((\varphi^+_\wp, \varphi^-_\wp)\big)_{\wp\in\mathrm{Prd}(\mathcal{V}^\star)}$, such that φ^+_p is equal to

$$\big(p(v_1) \vee q(v_1)\big) \wedge \big(p(v_1) \vee r(v_1)\big).$$

Let $\Omega = \big((O^+_\wp, O^-_\wp)\big)_{\wp\in\mathrm{Prd}(\mathcal{V}^\star)}$ be the literal marker for $\mathcal{P}$ such that O^+_p is equal to

$$\big\{\{\varphi^+_\wp,\, p(v_1) \vee r(v_1),\, p(v_1)\}\big\}.$$

Let E be defined as $\big\{p(\overline{2n}) \mid n \in \mathbb{N}\big\}$. Then $\mathcal{P} +_\Omega E$ is a formal logic program, say $\big((\psi^+_\wp, \psi^-_\wp)\big)_{\wp\in\mathrm{Prd}(\mathcal{V}^\star)}$, such that ψ^+_p will be defined in such a way that it is logically equivalent in $\mathcal{W}$ to

$$\Big(p(v_1) \vee q(v_1)\Big) \wedge \Big(p(v_1) \vee \bigvee_{n\in\mathbb{N}} v_1 \doteq \overline{2n} \vee r(v_1)\Big).$$

If we modify the example and assume that E is rather set to $\{p(s(s(v_0)))\}$, then ψ^+_p will be defined in such a way that it is logically equivalent in $\mathcal{W}$ to

$$\Big(p(v_1) \vee q(v_1)\Big) \wedge \Big(p(v_1) \vee \exists v_0\big(v_1 \doteq s(s(v_0))\big) \vee r(v_1)\Big).$$

The eventual definition of $\mathcal{P} +_\Omega E$ for arbitrary choices of $\mathcal{P}$, Ω and E, will be a straightforward generalization of those examples. One should keep in mind that E is meant to be a set of literals over $\mathcal{V}^\star$ (as opposed to a set of literals over $\mathcal{V}$), with $\mathcal{V}$ and $\mathcal{V}^\star$ being possibly different, which means that again, we cannot assume in full generality that E can be restricted to consist of closed literals only.

The notation that follows should be thought of as recording the set of all possible substitutions thanks to which a given statement φ can be shown to subsume some member of a set E of statements.

Notation 38. Given a statement φ, $n \in \mathbb{N}$, pairwise distinct variables $x_1, \ldots, x_n$ with $\mathrm{fv}(\varphi) = \{x_1, \ldots, x_n\}$, and a set E of statements, we let $\mathrm{Unif}(\varphi, E)$ denote the set of all statements of the form[1]

$$\exists y_1 \ldots \exists y_m (x_1 \doteq t_1 \wedge \ldots \wedge x_n \doteq t_n)$$

where $t_1, \ldots, t_n$ are terms over $\mathcal{V}^\star$, m is a member of $\mathbb{N}$, $y_1, \ldots, y_m$ are pairwise distinct variables, all distinct from $x_1, \ldots, x_n$, $\{y_1, \ldots, y_m\}$ is the set of variables that occur in at least one of $t_1, \ldots, t_n$ and for all closed terms $t'_1, \ldots, t'_n$, if $(t'_1, \ldots, t'_n)$ is an instance of $(t_1, \ldots, t_n)$ then $\varphi[t'_1/x_1, \ldots, t'_n/x_n]$ is an instance of a member of E.

The notation that follows describes the operations of strengthening or weakening some occurrences of literals in a given statement: given a statement φ, a set O of occurrences of literals in φ and a set E of literals,

[1] In case $n = 0$, $\mathrm{Unif}(\varphi, E)$ is either $\varnothing$ or $\{\bigwedge \varnothing\}$.

- $\odot_E^O\varphi$ denotes the statement obtained from φ by assuming that any occurrence of literal in φ that belongs to O is false unless it subsumes some member of E;
- $\circledcirc_E^O\varphi$ denotes the statement obtained from φ by assuming that any occurrence of literal in φ that belongs to O is true if it subsumes some member of E.

The first operation prepares the technical definition of a formal logic program obtained from $\mathcal{P}$ and E, and denoted $\mathcal{P} \mid_\Omega E$, that will be useful to easily formalize in our setting answer-set programming, the stable model semantics, and the well founded semantics; for the latter two, Ω will actually be set to $\langle - \rangle_\mathcal{P}$, which prompts for a special notation, that of Notation 40. The second operation prepares the definition of $\mathcal{P} +_\Omega E$. Both $\mathcal{P} \mid_\Omega E$ and $\mathcal{P} +_\Omega E$ are formally defined in Notation 42.

Notation 39. Let E be a set of literals. We inductively define for all statements φ and sets O of occurrences of literals in φ two statements $\odot_E^O\varphi$ and $\circledcirc_E^O\varphi$. Let $\varphi \in \mathcal{L}_{\omega_1\omega}(\mathcal{V})$ and a set O of occurrences of literals in φ be given.

- Suppose that φ is of the form $\bigvee X$ or $\bigwedge X$. For all $\psi \in X$, let O_ψ be the (unique) set o of occurrences of literals in ψ with $o \cup \{\varphi\} \in O$.
  - If φ is the statement $\bigvee X$ then $\odot_E^O\varphi$ is $\bigvee\{\odot_E^{O_\psi}\psi \mid \psi \in X\}$ and $\circledcirc_E^O\varphi$ is $\bigvee\{\circledcirc_E^{O_\psi}\psi \mid \psi \in X\}$.
  - If φ is the statement $\bigwedge X$ then $\odot_E^O\varphi$ is $\bigwedge\{\odot_E^{O_\psi}\psi \mid \psi \in X\}$ and $\circledcirc_E^O\varphi$ is $\bigwedge\{\circledcirc_E^{O_\psi}\psi \mid \psi \in X\}$.
- Suppose that φ is of the form $\exists x\, \psi$ or $\forall x\, \psi$. Let O_ψ be the (unique) set o of occurrences of literals in ψ with $o \cup \{\varphi\} \in O$.
  - If φ is the statement $\exists x\, \psi$ then $\odot_E^O\varphi$ is $\exists x\, \odot_E^{O_\psi}\psi$ and $\circledcirc_E^O\varphi$ is $\exists x\, \circledcirc_E^{O_\psi}\psi$.
  - If φ is the statement $\forall x\, \psi$ then $\odot_E^O\varphi$ is $\forall x\, \odot_E^{O_\psi}\psi$ and $\circledcirc_E^O\varphi$ is $\forall x\, \circledcirc_E^{O_\psi}\psi$.
- Suppose that φ is an identity, a distinction, or a literal.
  - If $O = \varnothing$ then both $\odot_E^O\varphi$ and $\circledcirc_E^O\varphi$ are φ.
  - If $O = \{\varphi\}$ then $\odot_E^O\varphi$ is $\bigvee \mathrm{Unif}(\varphi, E)$ and $\circledcirc_E^O\varphi$ is $\bigvee\{\varphi\} \cup \mathrm{Unif}(\varphi, E)$.

Notation 40. Given $\varphi \in \mathcal{L}_{\omega_1\omega}(\mathcal{V})$ and a set E of literals, and letting O be the set of occurrences of negated atoms in φ, we write $\odot_E^-\varphi$ for $\odot_E^O\varphi$ and $\circledcirc_E^-\varphi$ for $\circledcirc_E^O\varphi$.

Example 41. Suppose that $\mathcal{P}$ is the formal logic program of Example 16 and

$$E = \{\neg p_3(\overline{1}),\ p_3(\overline{2}),\ \neg p_3(\overline{2}),\ p_3(\overline{3}),\ p_4(\overline{2}),\ \neg p_4(\overline{1}),\ \neg q_5\}.$$

- $\odot^-_E\varphi^+_{p_3}$ and $\circledcirc^-_E\varphi^+_{p_3}$ are logically equivalent in $\mathcal{W}$ to

$$v_1 \doteq \overline{0} \vee \exists v_0\big(v_1 \doteq s(v_0) \wedge (v_0 \doteq \overline{1} \vee v_0 \doteq \overline{2})\big),$$

and

$$v_1 \doteq \overline{0} \vee \exists v_0\big(v_1 \doteq s(v_0) \wedge (\neg p_3(v_0) \vee v_0 \doteq \overline{1} \vee v_0 \doteq \overline{2})\big),$$

which are logically equivalent in $\mathcal{W}$ to

$$v_1 \doteq \overline{0} \vee v_1 \doteq \overline{2} \vee v_1 \doteq \overline{3}$$

and

$$v_1 \doteq \overline{0} \vee v_1 \doteq \overline{2} \vee v_1 \doteq \overline{3} \vee \exists v_0\big(v_1 \doteq s(v_0) \wedge \neg p_3(v_0)\big),$$

respectively.

- $\odot^-_E\varphi^+_{p_4}$ and $\circledcirc^-_E\varphi^+_{p_4}$ are both logically equivalent in $\mathcal{W}$ to $\varphi^+_{p_4}$.
- $\odot^-_E\varphi^-_{p_4}$ and $\odot^-_E\varphi^-_{q_2}$ are both logically equivalent in $\mathcal{W}$ to $\bigvee \varnothing$, while $\circledcirc^-_E\varphi^-_{p_4}$ and $\circledcirc^-_E\varphi^-_{q_2}$ are logically equivalent in $\mathcal{W}$ to $\varphi^-_{p_4}$ and $\varphi^-_{q_2}$, respectively.
- $\odot^-_E\varphi^+_{q_5}$ and $\circledcirc^-_E\varphi^+_{q_5}$ are both logically equivalent in $\mathcal{W}$ to $\bigwedge \varnothing$.

Notation 42. Let a formal logic program $\mathcal{P}$ and a literal marker Ω for $\mathcal{P}$ be given. Write $\mathcal{P} = \big((\varphi^+_\wp, \varphi^-_\wp)\big)_{\wp\in\mathrm{Prd}(\mathcal{V}^\star)}$ and $\Omega = \big((O^+_\wp, O^-_\wp)\big)_{\wp\in\mathrm{Prd}(\mathcal{V}^\star)}$. Let a set E of literals be given.

- We let $\mathcal{P}\,|_\Omega\, E$ denote $\big((\odot^{O^+_\wp}_E\varphi^+_\wp, \odot^{O^-_\wp}_E\varphi^-_\wp)\big)_{\wp\in\mathrm{Prd}(\mathcal{V}^\star)}$.
- We let $\mathcal{P} +_\Omega E$ denote $\big((\circledcirc^{O^+_\wp}_E\varphi^+_\wp, \circledcirc^{O^-_\wp}_E\varphi^-_\wp)\big)_{\wp\in\mathrm{Prd}(\mathcal{V}^\star)}$.

The transformation of a formal logic program $\mathcal{P}$ into a formal logic program of the form $\mathcal{P} +_\Omega E$ will be of interest only in case Ω and E are chosen in such a way that the condition in the definition that follows holds.

Definition 43. Let a formal logic program $\mathcal{P}$ be given. We call *extensor for* $(\mathcal{P}, \Omega)$ any set E of literals over $\mathcal{V}^\star$ such that $[\,\mathcal{P} +_\Omega E\,]$ is consistent.

The next property justifies the terminology.

Property 44. *For all formal logic programs $\mathcal{P}$, literal markers Ω for $\mathcal{P}$ and extensors E for $(\mathcal{P}, \Omega)$, $[\,\mathcal{P}\,] \subseteq [\,\mathcal{P} +_\Omega E\,]$.*

The next property will play a key role when we establish the relationship between this framework and the three semantics besides Kripke-Kleene semantics under consideration in this paper.

Property 45. *Let a formal logic program $\mathcal{P}$ and a literal marker Ω for $\mathcal{P}$ be given. If either all closed instances of members of $[\,\mathcal{P} +_\Omega E\,]$ or all closed instances of members of $[\,\mathcal{P}\,|_\Omega E\,]$ are instances of members of E then $[\,\mathcal{P} +_\Omega E\,]$ is equal to $[\,\mathcal{P}\,|_\Omega E\,]$.*

Before we can end this section, we need one more technical notation. In relation to the well founded semantics, we will see that one needs to consider not a single extensor, but a family of extensors: a formal logic program $\mathcal{P}$ will be extended to a formal logic program of the form $\mathcal{P} +_\Omega E_0$, that will itself be extended to a formal logic program of the form $\mathcal{P} +_\Omega E_0 \cup E_1$ determined by $\mathcal{P} +_\Omega E_0$ and an occurrence marker Ω_1 for $\mathcal{P} +_\Omega E_0$, that will itself be extended to a formal logic program of the form $\mathcal{P} +_\Omega E_0 \cup E_1 \cup E_2$ determined by $\mathcal{P} +_\Omega E_0 \cup E_1$ and an occurrence marker Ω_2 for $\mathcal{P} +_\Omega E_0 \cup E_1$ ...Now Ω_1, Ω_2, etc., will not be arbitrary: they will all select occurrences of literals in $\mathcal{P} +_\Omega E_0$, $\mathcal{P} +_\Omega E_0 \cup E_1$, etc., determined by Ω, even though these occurrences of literals are taken from different formal logic programs as $\mathcal{P}$ is being successively transformed. For instance, in Example 41, the occurrence of $\neg p_3(v_0)$ in $\varphi^+_{p_3}$ can still to be "tracked down" in $\circledcirc^-_E \varphi^+_{p_3}$, though they are formally two different sets of statements. The couple of notation that follows will allow us to formally express Ω_1, Ω_2, Ω_3, etc., from Ω and E_0, E_1, E_2, etc., and write $\Omega + E_0$ for Ω_1, $\Omega + E_0 \cup E_1$ for Ω_2, $\Omega + E_0 \cup E_1 \cup E_2$ for Ω_3, etc.

Notation 46. For all statements φ, sets O of occurrences of literals in φ and nonsingleton members o of O, let $\rho(O, o)$ be the set of occurrences o' of literals in the statement $o \setminus \{\varphi\}$ is an occurrence of literal in with $o' \cup \{\varphi\} \in O$. Given a statement φ, a set O of occurrences of literals in φ, a set E of literals and a member o of O, set

$$\circledcirc^O_E o = \begin{cases} \{\circledcirc^O_E \varphi\} \cup \circledcirc^{\rho(O,\,o)}_E o \setminus \{\varphi\} & \text{if } \varphi \text{ is not a literal,} \\ \{\circledcirc^O_E \varphi,\, \varphi\} & \text{otherwise.} \end{cases}$$

Notation 47. Let a formal logic program $\mathcal{P}$, a literal marker Ω for $\mathcal{P}$ and a set E of literals be given. Write $\Omega = \big((O^+_\wp,\, O^-_\wp)\big)_{\wp \in \mathrm{Prd}(\mathcal{V}^\star)}$. We let $\Omega + E$ denote

$$\Big(\big(\{\circledcirc^{O^+_\wp}_E o \mid o \in O^+_\wp\},\, \{\circledcirc^{O^-_\wp}_E o \mid o \in O^-_\wp\}\big)\Big)_{\wp \in \mathrm{Prd}(\mathcal{V}^\star)}.$$

4.2 Special extensors

The task of casting the well founded semantics, the stable model semantics and answer-set programming into our framework boils down to defining appropriate literal markers and extensors. At a fundamental level, the question "what are legitimate contextual assumptions?" replaces the question "how does negation behave?" We now define the key properties that literal markers and extensors can enjoy and allow one to complete that task.

Definition 48. Let a formal logic program $\mathcal{P}$, a literal marker Ω for $\mathcal{P}$, and an extensor E for $(\mathcal{P}, \Omega)$ be given.

- We say that E is *imperative* iff for all closed literals φ,

  $$\varphi \text{ is not an instance of a member of } E \text{ iff } [\,\mathcal{P} +_\Omega E\,] \Vdash \sim\varphi.$$

- We say that E is *implicative* iff $E \subseteq [\,\mathcal{P} +_\Omega E\,]$.

- We say that E is *supporting* iff for all $\psi \in E$, some member of $(\mathcal{P} +_\Omega E)[\psi]$ is included in $[\,\mathcal{P}\,]$.
- Given an ordinal α, we say that E is *α-foundational* iff there exists a sequence $(E_\beta)_{\beta<\alpha}$ of sets of literals such that $E = \bigcup_{\beta<\alpha} E_\beta$ and for all $\beta < \alpha$, E_β is a supporting extensor for $(\mathcal{P} +_\Omega \bigcup_{\gamma<\beta} E_\gamma,\ \Omega + \bigcup_{\gamma<\beta} E_\gamma)$.
- We say that E is *foundational* iff there exists a sequence $(E_\alpha)_{\alpha\in\mathrm{Ord}}$ of sets of literals such that $E = \bigcup_{\alpha\in\mathrm{Ord}} E_\alpha$ and for all ordinals α, $\bigcup_{\beta<\alpha} E_\beta$ is an α-foundational extensor for $(\mathcal{P},\ \Omega)$.

Already observe the following properties.

Property 49. *For all formal logic programs $\mathcal{P}$ and literal markers Ω for $\mathcal{P}$, all imperative extensors for $(\mathcal{P},\ \Omega)$ are saturated.*

Property 50. *For all formal logic programs $\mathcal{P}$ and literal markers Ω for $\mathcal{P}$, all supporting extensors for $(\mathcal{P},\ \Omega)$ are implicative.*

It will be shown that the well founded semantics is related to foundational extensors, and answer-set programming to imperative extensors. As for the stable model semantics, it will be shown to be related to both imperative and implicative extensors, by virtue of the following property.

Property 51. *For all formal logic programs $\mathcal{P}$, literal markers Ω for $\mathcal{P}$ and complete sets E of literals over $\mathcal{V}^\star$, E is an implicative extensor for $(\mathcal{P},\ \Omega)$ iff E is an imperative extensor for $(\mathcal{P},\ \Omega)$.*

In case $\mathcal{V}^\star$ is equal to $\mathcal{V}$, it is easy to see that the class of imperative extensors has a "neutral element":

Property 52. *Suppose that $\mathcal{V}^\star = \mathcal{V}$. Let a consistent formal logic program $\mathcal{P}$ be given. Let E be the set of all closed literals φ with $\sim\varphi \notin [\,\mathcal{P}\,]$. Then for all literal markers Ω for $\mathcal{P}$, E is an imperative extensor for $(\mathcal{P},\ \Omega)$ and $[\,\mathcal{P} +_\Omega E\,] = [\,\mathcal{P}\,]$.*

It is fair to say that to cast answer-set programming, the stable model semantics and the well founded semantics into our framework, it would be sufficient to work under the assumption that $\mathcal{V}^\star = \mathcal{V}$: either these semantics are developed in a propositional setting, or they restrict the class of interpretations to Herbrand structures. There is no need to impose such restrictions, but a natural question is how much more general the notions become when the equality $\mathcal{V}^\star = \mathcal{V}$ is not imposed. In relation to answer-set programming and the stable model semantics, the answer is: not much more. Indeed, the following proposition establishes that when $\mathcal{V}^\star$ and $\mathcal{V}$ are distinct, the notion of imperative extensor is often degenerate.

Proposition 53. *Suppose that $\mathcal{V} \setminus \mathcal{V}^\star$ contains a function symbol of arity 1 at least. Let a formal logic program $\mathcal{P}$, a literal marker Ω for $\mathcal{P}$, and an imperative extensor E for $(\mathcal{P},\ \Omega)$ be given. Then for all $n \in \mathbb{N}$ and $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$, the set of members of $[\,\mathcal{P} +_\Omega E\,]$ of the form $\wp(t_1, \ldots, t_n)$ or $\neg\wp(t_1, \ldots, t_n)$ is either empty or equal to the set of all atoms over $\mathcal{V}^\star$ of the form $\wp(t_1, \ldots, t_n)$ or equal to the set of all negated atoms over $\mathcal{V}^\star$ of the form $\neg\wp(t_1, \ldots, t_n)$.*

Proof. There is nothing to prove if $\mathcal{V}$ contains no constant, so suppose otherwise. Let $n \in \mathbb{N}$ and $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$ be given. It is easy to verify that there exists 3 sets X, Y and Z of n-tuples of terms over $\mathcal{V}^\star$ such that for all closed terms t_1, $\ldots$, t_n, one of the following conditions holds.

- Both $\wp(t_1, \ldots, t_n)$ and $\neg\wp(t_1, \ldots, t_n)$ are instances of members of E and $(t_1, \ldots, t_n)$ is an instance of some member of X, but not of any member of $Y \cup Z$.
- $\wp(t_1, \ldots, t_n)$ is an instance of a member of E, $\neg\wp(t_1, \ldots, t_n)$ is not an instance of any member of E, and $(t_1, \ldots, t_n)$ is an instance of some member of Y, but not of any member of $X \cup Z$.
- $\neg\wp(t_1, \ldots, t_n)$ is an instance of a member of E, $\wp(t_1, \ldots, t_n)$ is not an instance of any member of E, and $(t_1, \ldots, t_n)$ is an instance of some member of Z, but not of any member of $X \cup Y$.

Let a nonnullary function symbol f in $\mathcal{V} \setminus \mathcal{V}^\star$ be given. Then there exists an n-tuple $(\iota_1, \ldots, \iota_n)$ of pairwise distinct closed terms that all start with f. Obviously, for all terms t_1, $\ldots$, t_n over $\mathcal{V}^\star$, if $(\iota_1, \ldots, \iota_n)$ is an instance of $(t_1, \ldots, t_n)$ then t_1, $\ldots$, t_n are pairwise distinct variables. So

- either all n-tuples of closed terms are instances of some member of X, in which case $[\,\mathcal{P} +_\Omega E\,]$ contains no literal over $\mathcal{V}^\star$ of the form $\wp(t_1, \ldots, t_n)$ or $\neg\wp(t_1, \ldots, t_n)$,
- or all n-tuples of closed terms are instances of some member of Y, in which case $[\,\mathcal{P} +_\Omega E\,]$ contains all literals over $\mathcal{V}^\star$ of the form $\wp(t_1, \ldots, t_n)$,
- or all n-tuples of closed terms are instances of some member of Z, in which case $[\,\mathcal{P} +_\Omega E\,]$ contains all literals over $\mathcal{V}^\star$ of the form $\neg\wp(t_1, \ldots, t_n)$,

completing the proof of the proposition. □

The following example shows that if $\mathcal{V} \setminus \mathcal{V}^\star$ does not contain a function symbol of arity 1 at least, then the notion of imperative extensor does not have to be degenerate.

Example 54. Suppose that $\mathcal{V}$ consists of $\overline{0}$, s and a binary predicate symbol p, and assume that $\mathcal{V}^\star = \{s, p\}$. Set $\mathcal{P} = \big((v_0 \doteq v_1,\ \bigvee \varnothing)\big)$. Let E be the set of literals defined as

$$\{p(v_0, v_0)\} \cup \{p(s^n(v_0), v_0),\ p(v_0, s^n(v_0)),\ \neg p(s^n(v_0), v_0),\ \neg p(v_0, s^n(v_0)) \mid n > 0\}.$$

Set $\Omega = \big((\varnothing, \varnothing)\big)$. Then E is an imperative extensor for $(\mathcal{P}, \Omega)$ and $[\,\mathcal{P} +_\Omega E\,]$, which is obviously equal to $[\,\mathcal{P}\,]$, is the set of all atoms over $\mathcal{V}^\star$ of the form $p(t, t)$.

To summarize the previous considerations, we have not assumed in Definition 48 that $\mathcal{V}^\star$ and $\mathcal{V}$ are equal simply because none of the results we want to establish need that assumption to be made. But the notion of imperative extensor (which is the key notion in relation to answer-set programming and the stable model semantics) is defined in such a way that cases where that notion takes interesting values are cases where $\mathcal{V}^\star = \mathcal{V}$ or where $\mathcal{V}^\star$ and $\mathcal{V}$ are very special.

4.3 A few technical results

The technical results that follow will be used in the sequel.

Lemma 55. *Let a formal logic program $\mathcal{P}$ and a literal marker Ω for $\mathcal{P}$ be given. For all sets E and F of literals, if $E \subseteq F$ then $[\,\mathcal{P} +_\Omega E\,] \subseteq [\,\mathcal{P} +_\Omega F\,]$.*

Proof. Let E and F be two sets of literals with $E \subseteq F$. It is immediately verified by induction that for all ordinals α, $[\,\mathcal{P} +_\Omega E\,]_\alpha \subseteq [\,\mathcal{P} +_\Omega F\,]_\alpha$. We conclude with Property 32. □

Lemma 56. *Let a formal logic program $\mathcal{P}$ and a literal marker Ω for $\mathcal{P}$ be given. For all sets E and F of literals, $\big[\mathcal{P} +_\Omega [\,\mathcal{P} +_\Omega E\,] \cup F\,\big] \subseteq [\,\mathcal{P} +_\Omega E \cup F\,]$.*

Proof. Let E and F be two sets of literals. Let ordinal λ be such that $[\,\mathcal{P} +_\Omega E\,]_\lambda$ is equal to $[\,\mathcal{P} +_\Omega E\,]_{\lambda+1}$. It is easy to verify by induction that for all ordinals α, $[\,\mathcal{P} +_\Omega [\,\mathcal{P} +_\Omega E\,] \cup F\,]_\alpha \subseteq [\,\mathcal{P} +_\Omega E \cup F\,]_{\lambda+\alpha}$. We conclude with Property 32. □

Proposition 57. *Let a formal logic program $\mathcal{P}$ be locally consistent. Let a literal marker Ω for $\mathcal{P}$ be given. Let a set X of implicative extensors for $(\mathcal{P}, \Omega)$ be such that $\bigcup X$ is consistent. Then $\bigcup X$ is an extensor for $(\mathcal{P}, \Omega)$.*

Proof. Write $\mathcal{P} = \big((\varphi^+_\wp, \varphi^-_\wp)\big)_{\wp \in \mathrm{Prd}(\mathcal{V}^\star)}$. Set $E = \bigcup X$. We show by induction that for all ordinals α, $E \cup [\,\mathcal{P} +_\Omega E\,]_\alpha$ is consistent. Let an ordinal α be given and assume that for all $\beta < \alpha$, $E \cup [\,\mathcal{P} +_\Omega E\,]_\beta$ is consistent. Since $\mathcal{P}$ is locally consistent and E is consistent (used in the case where $\alpha = 0$), there exists no $n \in \mathbb{N}$, $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$ and closed terms $t_1, \ldots, t_n$ such that $E \cup \bigcup_{\beta<\alpha} [\,\mathcal{P} +_\Omega E\,]_\beta$ forces $\varphi^+_\wp[t_1/v_1, \ldots, t_n/v_n]$ and $\varphi^-_\wp[t_1/v_1, \ldots, t_n/v_n]$. Hence $E \cup [\,\mathcal{P} +_\Omega E\,]_\alpha$ cannot be inconsistent unless the set of closed instances of members of $[\,\mathcal{P} +_\Omega E\,]_\alpha$ intersects the set of closed instances of members of $\sim E$. Assume that the set of closed instances of members of $[\,\mathcal{P} +_\Omega E\,]_\alpha$ indeed intersects the set of closed instances of members of $\sim E$. Let ordinal λ be least such that there exists a closed literal φ with $\bigcup_{F \in X} [\,\mathcal{P} +_\Omega F\,]_\lambda \Vdash \varphi$ and $[\,\mathcal{P} +_\Omega E\,]_\alpha \Vdash \sim\varphi$. Let $F \in X$ and a closed literal φ be such that $[\,\mathcal{P} +_\Omega F\,]_\lambda \Vdash \varphi$ and $[\,\mathcal{P} +_\Omega E\,]_\alpha \Vdash \sim\varphi$. Set

$$Y = E \cup \bigcup_{\beta<\lambda} [\,\mathcal{P} +_\Omega F\,]_\beta \cup \bigcup_{\beta<\alpha} [\,\mathcal{P} +_\Omega E\,]_\beta.$$

We derive from the hypothesis on α and the choice of λ that Y is consistent. Let $n \in \mathbb{N}$, $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$ and terms $t_1, \ldots, t_n$ be such that φ is $\wp(t_1, \ldots, t_n)$ or $\neg\wp(t_1, \ldots, t_n)$. By the choice of φ, Y forces both $\varphi^+_\wp[t_1/v_1, \ldots, t_n/v_n]$ and $\varphi^-_\wp[t_1/v_1, \ldots, t_n/v_n]$, which is impossible since $\mathcal{P}$ is locally consistent. We conclude that $E \cup [\,\mathcal{P} +_\Omega E\,]_\alpha$ is consistent. □

Corollary 58. *Let a formal logic program $\mathcal{P}$ be locally consistent. Let a literal marker Ω for $\mathcal{P}$ be given. Let X be a set of supporting extensors for $(\mathcal{P}, \Omega)$ such that $\bigcup X$ is consistent. Then $\bigcup X$ is a supporting extensor for $(\mathcal{P}, \Omega)$.*

Proof. This is an immediate consequence of Property 50 and Proposition 57. □

To end this section, let us give a simple application of some of the previous observations. Complete sets of literals can obviously be identified with standard structures, hence it is natural to ask whether a complete set of the form $[\,\mathcal{P}+_\Omega E\,]$ is a model of the classical logical form of $\mathcal{P}$. It is easy to answer that question positively for implicative extensors.

Proposition 59. *Let a formal logic program $\mathcal{P}$, a literal marker Ω for $\mathcal{P}$, and an implicative extensor E for $(\mathcal{P}, \Omega)$ be such that $[\,\mathcal{P} +_\Omega E\,]$ is complete. Then the set of closed instances of atoms in $[\,\mathcal{P} +_\Omega E\,]$ is a model of* $\mathrm{Clf}(\mathcal{P})$.

Proof. Obviously, for all statements φ and sets O of occurrences of literals in φ, φ logically implies $\circledcirc_E^O\varphi$ in $\mathcal{W}$. It follows that $\mathrm{Clf}(\mathcal{P} +_\Omega [\,\mathcal{P} +_\Omega E\,])$ logically implies $\mathrm{Clf}(\mathcal{P})$ in $\mathcal{W}$. Since $E \subseteq [\,\mathcal{P} +_\Omega E\,]$, we derive from Lemma 55 that $[\,\mathcal{P} +_\Omega E\,]$ is a subset of $\big[\mathcal{P} +_\Omega [\,\mathcal{P} +_\Omega E\,]\big]$. Moreover, Lemma 56 implies that $\big[\mathcal{P} +_\Omega [\,\mathcal{P} +_\Omega E\,]\big]$ is a subset of $[\,\mathcal{P} +_\Omega E\,]$. Hence the complete set $[\,\mathcal{P} +_\Omega E\,]$ is equal to $\big[\mathcal{P} +_\Omega [\,\mathcal{P} +_\Omega E\,]\big]$, and we derive from Corollary 29 that $[\,\mathcal{P} +_\Omega E\,]$ logically implies $\mathrm{Clf}(\mathcal{P}+_\Omega [\,\mathcal{P}+_\Omega E\,])$ in $\mathcal{W}$. We conclude that $[\,\mathcal{P}+_\Omega E\,]$ logically implies $\mathrm{Clf}(\mathcal{P})$ in $\mathcal{W}$. □

4.4 Relationship to answer-set programming

In this section we consider the enrichment of $\mathcal{L}_{\omega_1\omega}(\mathcal{V})$ with a second negation operator, written *not*, that can be applied to any literal, and to literals only. We do not develop the formalism beyond this minimalist syntactic consideration as we use *not* to remind the reader of the usual definition of answer-sets, but we will not use it in an alternative definition of answer-sets that will immediately be seen to be equivalent to the usual definition. For this purpose, let us first introduce some preliminary notation. Let a statement φ and a set O of occurrences of literals in φ be given. We define a member $\varphi[O]$ of the enrichment of $\mathcal{L}_{\omega_1\omega}(\mathcal{V})$ with *not*, thanks to the inductive construction that follows.

- Suppose that φ is of the form $\bigvee X$ or $\bigwedge X$. For all $\psi \in X$, let O_ψ be the (unique) set o of occurrences of literals in ψ with $o \cup \{\varphi\} \in O$.
  - If φ is the statement $\bigvee X$ then $\varphi[O]$ is $\bigvee\{\psi[O_\psi] \mid \psi \in X\}$.
  - If φ is the statement $\bigwedge X$ then $\varphi[O]$ is $\bigwedge\{\psi[O_\psi] \mid \psi \in X\}$.
- Suppose that φ is of the form $\exists x\,\psi$ or $\forall x\,\psi$. Let O_ψ be the (unique) set o of occurrences of literals in ψ with $o \cup \{\varphi\} \in O$.
  - If φ is the statement $\exists x\,\psi$ then $\varphi[O]$ is $\exists x\,\psi[O_\psi]$.
  - If φ is the statement $\forall x\,\psi$ then $\varphi[O]$ is $\forall x\,\psi[O_\psi]$.
- Suppose that φ is an identity, a distinction, or a literal.
  - If $O = \varnothing$ then $\varphi[O]$ is φ.
  - If $O = \{\varphi\}$ and φ is an atom then $\varphi[O]$ is $not\,\neg\varphi$.
  - If $O = \{\varphi\}$ and φ is of the form $\neg\psi$ then $\varphi[O]$ is $not\,\psi$.

Now let a formal logic program $\mathcal{P}$ and a literal marker $\Omega = \big((O^+_\wp, O^-_\wp)\big)_{\wp \in \mathrm{Prd}(\mathcal{V}^\star)}$ for $\mathcal{P}$ be given. Set $\mathcal{P}[\Omega] = \big((\varphi^+_\wp[O^+_\wp], \varphi^-_\wp[O^-_\wp])\big)_{\wp \in \mathrm{Prd}(\mathcal{V}^\star)}$. Then $\mathcal{P}[\Omega]$ is what is known in the literature as a general logic program, a kind of logic program with two kinds of negation, $\neg$ and *not*. Conversely, let a general logic program G that, without loss of generality, is written in such a way that for every $n \in \mathbb{N}$ and $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$, G has one rule whose head is $\wp(v_1, \ldots, v_n)$, one rule whose head is $\neg\wp(v_1, \ldots, v_n)$, and no other rule whose head is of the form $\wp(t_1, \ldots, t_n)$ or $\neg\wp(t_1, \ldots, t_n)$. Then there exists a unique formal logic program $\mathcal{P}$ and a unique literal marker Ω for $\mathcal{P}$ such that $G = \mathcal{P}[\Omega]$.

For instance, assume that $\mathcal{V}$ consists of 4 nullary predicate symbols p_1, p_2, p_3 and p_4. Suppose that $\mathcal{P}$ is given by the following statements.

$$\begin{array}{llll} \varphi^+_{p_1} \text{ is } p_2 \wedge p_3 & \varphi^+_{p_2} \text{ is } p_4 & \varphi^+_{p_3} \text{ is } p_3 & \varphi^+_{p_4} \text{ is } \neg p_3 \\ \varphi^-_{p_1} \text{ is } p_2 \vee \neg p_4 & \varphi^-_{p_2} \text{ is } \neg p_3 & \varphi^-_{p_3} \text{ is } \neg p_3 \wedge p_2 & \varphi^-_{p_4} \text{ is } \bigvee \varnothing \end{array}$$

Suppose that Ω is given by the following sets.

$$\begin{array}{ll} O^+_{p_1} \text{ is } \varnothing & O^-_{p_1} \text{ is } \big\{\{p_2 \vee \neg p_4,\, p_2\},\, \{p_2 \vee \neg p_4,\, \neg p_4\}\big\} \\ O^+_{p_2} \text{ is } \varnothing & O^-_{p_2} \text{ is } \varnothing \\ O^+_{p_3} \text{ is } \varnothing & O^-_{p_3} \text{ is } \big\{\{\neg p_3 \wedge p_2,\, p_2\}\big\} \\ O^+_{p_4} \text{ is } \big\{\{\neg p_3\}\big\} & O^-_{p_4} \text{ is } \varnothing \end{array}$$

Then $\mathcal{P}[\Omega]$ is the general logic program

$$\begin{array}{llll} p_1 \leftarrow p_2 \wedge p_3 & p_2 \leftarrow p_4 & p_3 \leftarrow p_3 & p_4 \leftarrow not\, p_3 \\ \neg p_1 \leftarrow not\, \neg p_2 \vee not\, p_4 & \neg p_2 \leftarrow \neg p_3 & \neg p_3 \leftarrow \neg p_3 \wedge not\, \neg p_2 & \end{array}$$

Moreover, it is easy to see that the extensors for $(\mathcal{P}, \Omega)$ are:

- all subsets E of $\{p_1, \neg p_1, \neg p_2, p_3, p_4\}$, in which case $[\,\mathcal{P} +_\Omega E\,] = \varnothing$;
- all subsets E of $\{p_1, \neg p_1, p_2, \neg p_2, p_3, p_4, \neg p_4\}$ which contain at least one of p_2 and $\neg p_4$, in which case $[\,\mathcal{P} +_\Omega E\,] = \{\neg p_1\}$;
- all subsets E of $\{p_1, \neg p_1, p_2, \neg p_2, p_3, \neg p_3, p_4, \neg p_4\}$ which $\neg p_3$ belongs to, in which case $[\,\mathcal{P} +_\Omega E\,]$ is equal to $\{\neg p_1, p_2, p_4\}$.

Out of these, only $\{\neg p_1, p_2, p_3, \neg p_3, p_4\}$ is imperative. Moreover, there is a unique answer-set for $\mathcal{P}[\Omega]$, namely $\{\neg p_1, p_2, p_4\}$.

Having realized that the class of general logic programs is in one-to-one correspondence with the class of pairs $(\mathcal{P}, \Omega)$ where $\mathcal{P}$ is a formal logic program and Ω a literal marker for $\mathcal{P}$ (the correspondence in question putting a pair of the form $(\mathcal{P}, \Omega)$ in relation to $\mathcal{P}[\Omega]$), it is easy to see that if one assumes that $\mathcal{V}^\star$ is equal to $\mathcal{V}$, then Definition 60 amounts to the notion of an answer-set—recall the discussion at the end of Section 4.2 about not assuming that $\mathcal{V}^\star$ and $\mathcal{V}$ are equal.

Definition 60. Let a formal logic program $\mathcal{P}$ and a literal marker Ω for $\mathcal{P}$ be given. An *answer-set for* $(\,\mathcal{P}, \Omega\,)$ is a partial interpretation M for which there exists a saturated set E of literals over $\mathcal{V}^\star$ with the following property.

- For all closed literals φ, $\varphi \in M$ iff $\sim\varphi$ is not an instance of a member of E.
- M is the set of closed instances of members of $[\,\mathcal{P}\,|_\Omega\,E\,]$.

The next proposition shows that the concept of imperative extensor fully characterizes the notion of answer-set.

Proposition 61. *Let a formal logic program $\mathcal{P}$, a literal marker Ω for $\mathcal{P}$, and a set E of literals over $\mathcal{V}^\star$ be given. Let F be the set of closed instances of members of E, and let M be the set of all closed literals φ with $\sim\varphi \notin F$. Then E is an imperative extensor for $(\,\mathcal{P},\,\Omega\,)$ iff M is an answer-set for $(\,\mathcal{P},\,\Omega\,)$.*

Proof. Assume that E is an imperative extensor for $(\,\mathcal{P},\,\Omega\,)$. Since $[\,\mathcal{P}\,{+_\Omega}\,E\,]$ is consistent, we infer from Definition 48 that the closed instances of the member of $[\,\mathcal{P}\,{+_\Omega}\,E\,]$ are instances of members of E. Hence by Property 45, $[\,\mathcal{P}\,{+_\Omega}\,E\,]$ is equal to $[\,\mathcal{P}|_\Omega\,E\,]$. Moreover, it is immediately verified that $[\,\mathcal{P}|_\Omega\,E\,] = [\,\mathcal{P}|_\Omega\,F\,]$. We infer that M is an answer-set for $(\,\mathcal{P},\,\Omega\,)$.

Conversely, assume that M is an answer-set for $(\mathcal{P},\,\Omega)$ (hence M is consistent). By Definition 60, the closed instances of the members of $[\,\mathcal{P}\,|_\Omega\,F\,]$ belong to F. Hence by Property 45, $[\,\mathcal{P}{+_\Omega}F\,] = [\,\mathcal{P}|_\Omega F\,]$. Moreover, it is immediately verified that $[\,\mathcal{P}\,{+_\Omega}\,E\,] = [\,\mathcal{P}\,{+_\Omega}\,F\,]$. We infer that M is the set of closed instances of the members of $[\,\mathcal{P}\,{+_\Omega}\,E\,]$ and E is an imperative extensor for $(\,\mathcal{P},\,\Omega\,)$. □

In answer-set programming, *not* φ intuitively means that φ is not provable. The way to go from the usual presentation of answer-set programming to our setting is to replace global hypotheses of the form "assume that φ is not provable" by local hypotheses of the form "assume here that $\sim\varphi$ holds." What is then derived has to be consistent with any individual local hypothesis. Moreover, whenever an assumption can be made, that assumption *should* be made, unless it contradicts what can then be derived (from the formal logic program together with the assumptions that are made). In the usual treatment of answer-set programming, the justification to the approach is that if φ cannot be derived then φ is indeed not provable. In our setting, we just impose a particular constraint on extensors, that can be paraphrased as: *make as many assumptions as possible, in all allowed contexts, except for assumptions that individually contradict what can then be derived.*

4.5 Relationship to the stable model semantics

The stable model semantics takes the class of sets of positive rules as object of study; but as mentioned repeatedly, this class is in one-to-one correspondence with the class of symmetric formal logic programs, hence it is legitimate to study the stable model semantics on the basis of the latter. If one assumes that $\mathcal{V}^\star$ is equal to $\mathcal{V}$, then Definition 62 captures the notion of stable model—again, recall the discussion at the end of Section 4.2 about not assuming that $\mathcal{V}^\star$ and $\mathcal{V}$ are equal. Note how Notation 39 is being used in Definition 62 to basically describe the Lloyd-Topor transformation.

Definition 62. Let a formal logic program $\mathcal{P} = \big((\varphi_\wp^+, \varphi_\wp^-)\big)_{\wp \in \mathrm{Prd}(\mathcal{V}^\star)}$ be given. A partial interpretation M is said to be *stable for* $\mathcal{P}$ iff there exists a complete set E of literals over $\mathcal{V}^\star$ such that M is the set of closed instances of members of E and for all closed atoms φ, $\varphi \in M$ iff

$$\big\{\odot_E^- \varphi_\wp^+ \to \wp(v_1, \ldots, v_n) \;\big|\; n \in \mathbb{N},\, \wp \in \mathrm{Prd}(\mathcal{V}^\star, n)\big\} \vDash_{\mathcal{W}} \varphi.$$

Note that the condition on E only depends on the positive rules of $\mathcal{P}$. In Definition 62, $\mathcal{P}$ is not assumed to be symmetric; but it is essential to assume that $\mathcal{P}$ is symmetric to obtain the result stated in the proposition that follows. Together with Property 51, this proposition shows that both concepts of imperative and implicative extensors relative to the literal markers that collect all occurrences of all negated atoms fully characterize the notion of stable model.

Proposition 63. *For all symmetric formal logic programs $\mathcal{P}$ and complete sets E of literals over $\mathcal{V}^\star$, the set of closed instances of members of E is stable for $\mathcal{P}$ iff E is an implicative extensor for $(\mathcal{P}, \langle - \rangle_{\mathcal{P}})$.*

Proof. Let a symmetric formal logic program $\mathcal{P} = \big((\varphi_\wp^+, \varphi_\wp^-)\big)_{\wp \in \mathrm{Prd}(\mathcal{V}^\star)}$ and a complete set E of literals over $\mathcal{V}^\star$ be given. Let E^+ be the set of atoms in E, and let E^- be the set of negated atoms in E. Suppose that the set of closed instances of members of E is stable for $\mathcal{P}$. Then $\mathrm{Clf}(\mathcal{P} \,|_{\langle - \rangle_{\mathcal{P}}}\, E)$ logically implies E^+ in $\mathcal{W}$. Since negation does not occur in any implication in $\mathrm{Clf}(\mathcal{P} \,|_{\langle - \rangle_{\mathcal{P}}}\, E)$, it follows that E^+ is a subset of $[\,\mathcal{P} \,|_{\langle - \rangle_{\mathcal{P}}}\, E\,]$. Let $n \in \mathbb{N}$, $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$, and closed terms $t_1, \ldots, t_n$ be given. Since E is complete and $\mathcal{P}$ is symmetric, E forces one and only one of $\varphi_\wp^+[t_1/v_1, \ldots, t_n/v_n]$ and $\varphi_\wp^-[t_1/v_1, \ldots, t_n/v_n]$. Suppose that $\neg\wp(t_1, \ldots, t_n)$ is an instance of a member of E. If $E \Vdash \varphi_\wp^+[t_1/v_1, \ldots, t_n/v_n]$ then $E^+ \Vdash \odot_E^- \varphi_\wp^+[t_1/v_1, \ldots, t_n/v_n]$, hence there exists terms $t_1', \ldots, t_n'$ over $\mathcal{V}^\star$ such that $(t_1, \ldots, t_n)$ is an instance of $(t_1', \ldots, t_n')$ and $\wp(t_1', \ldots, t_n') \in E$, contradicting the assumption that E is consistent. We infer that $E^+ \Vdash \odot_E^- \varphi_\wp^-[t_1/v_1, \ldots, t_n/v_n]$, hence there exists terms $t_1', \ldots, t_n'$ over $\mathcal{V}^\star$ such that $(t_1, \ldots, t_n)$ is an instance of $(t_1', \ldots, t_n')$ and $\neg\wp(t_1', \ldots, t_n')$ belongs to $[\,\mathcal{P} \,|_{\langle - \rangle_{\mathcal{P}}}\, E\,]$. Suppose that $\wp(t_1, \ldots, t_n)$ is an instance of a member of E. Then $E \Vdash \varphi_\wp^+[t_1/v_1, \ldots, t_n/v_n]$, hence E does not force $\varphi_\wp^-[t_1/v_1, \ldots, t_n/v_n]$, hence E does not force $\odot_E^- \varphi_\wp^-[t_1'/v_1, \ldots, t_n'/v_n]$ for any terms $t_1', \ldots, t_n'$ over $\mathcal{V}^\star$ such that $(t_1, \ldots, t_n)$ is an instance of $(t_1', \ldots, t_n')$. It is then easy to conclude that for all closed literals φ, $[\,\mathcal{P} \,|_\Omega\, E\,] \Vdash \varphi$ iff φ is an instance of a member of E. Together with Property 45, this completes the verification that E is an implicative extensor for $(\mathcal{P}, \langle - \rangle_{\mathcal{P}})$.

Conversely, assume that E is an implicative extensor for $(\mathcal{P}, \langle - \rangle_{\mathcal{P}})$. By Property 45 again, $[\,\mathcal{P} +_\Omega E\,] = [\,\mathcal{P} \,|_\Omega\, E\,]$. Set

$$X = \big\{\odot_E^- \varphi_\wp^+ \to \wp(v_1, \ldots, v_n) \;\big|\; n \in \mathbb{N},\, \wp \in \mathrm{Prd}(\mathcal{V}^\star, n)\big\}.$$

Clearly, $\mathrm{Clf}(\mathcal{P} \,|_{\langle - \rangle_{\mathcal{P}}}\, E)$, being logically equivalent in $\mathcal{W}$ to the complete set E, is also logically equivalent to $E^- \cup X$ in $\mathcal{W}$. Hence $E^- \cup X \vDash_{\mathcal{W}} E^+$. Since negation does not occur in any implication in X, this implies that $X \vDash_{\mathcal{W}} E^+$, which completes the verification that the set of closed instances of members of E is stable for $\mathcal{P}$. □

4.6 Supporting and foundational extensors

The well founded semantics is related to the notion of foundational extensor, and we will need to establish some of the properties that the latter enjoys in order to establish the relationship. The notion of supporting extensor has mainly been introduced as a useful building block in the definition of foundational extensors, but it is interesting in its own right. By Property 50, supporting extensors are implicative extensors, which means that the assumptions that make up such extensors are guaranteed to be generated. But more is true. Intuitively, given a formal logic program $\mathcal{P}$ and a literal marker Ω for $\mathcal{P}$, a supporting extensor for $(\mathcal{P}, \Omega)$ is sufficiently rich in literals to "generate itself" using $\mathcal{P}$ and Ω, and not contradict any literal generated by $\mathcal{P} +_\Omega E$. So for all members φ of a supporting extensor E for $(\mathcal{P}, \Omega)$, there exists a "constructive proof" of φ, from the rules formalized as $\mathcal{P}$, such that the only literals that occur in the proof either are in $[\,\mathcal{P}\,]$ or are members of E that occur in contexts where Ω accepts that they be assumed. The next example will help grasp the idea in the simple case where Ω accepts that any literal be assumed in any context, and where no member of $[\,\mathcal{P}\,]$ is actually needed in the "constructive proofs."

Example 64. Suppose that $\mathcal{P}$ is the formal logic program of Example 7. Then the supporting extensors for $(\mathcal{P}, \langle\pm\rangle_\mathcal{P})$ which are disjoint from $[\,\mathcal{P}\,]$ are the consistent unions of

- $\{p_4(\overline{2n}) \mid n \geq m\}$ where m is an arbitrary member of $\mathbb{N}$,
- $\{\neg p_4(\overline{2n}) \mid n \geq m\}$ where m is an arbitrary member of $\mathbb{N}$,
- $\{p_4(\overline{2n+1}) \mid n \geq m\}$ where m is an arbitrary member of $\mathbb{N}$,
- $\{\neg p_4(\overline{2n+1}) \mid n \geq m\}$ where m is an arbitrary member of $\mathbb{N}$,
- $\{q_2, q_3\}$,
- $\{\neg q_2, \neg q_3\}$, and
- $\{q_4\}$.

So with $\mathcal{P}$ defined in Example 7, a particular supporting extensor for $(\mathcal{P}, \langle\pm\rangle_\mathcal{P})$ is

$$\{p_4(\overline{2n}),\ \neg p_4(\overline{2n+1}) \mid n \in \mathbb{N}\}.$$

It will allow one to transform $\mathcal{P}$ into a formal logic program that provides a fourth way of generating the set of even numbers and its complement, using the predicate symbol p_4—besides the three options already available with p_1, p_2 and p_3.

Let E be a supporting extensor for $(\mathcal{P}, \Omega)$. Usually, $[\,\mathcal{P} +_\Omega E\,]$ encompasses more literals than those in $[\,\mathcal{P}\,] \cup E$. A nice property of the class of supporting extensors for $(\mathcal{P}, \Omega)$ is that it actually contains $[\,\mathcal{P} +_\Omega E\,]$.

Proposition 65. *For all formal logic programs $\mathcal{P}$, literal markers Ω for $\mathcal{P}$ and implicative extensors E for $(\mathcal{P}, \Omega)$, $[\,\mathcal{P} +_\Omega E\,]$ is a supporting extensor for $(\mathcal{P}, \Omega)$.*

Proof. By Lemmas 55 and 56, $[\,\mathcal{P} +_\Omega E\,] = [\,\mathcal{P} +_\Omega [\,\mathcal{P} +_\Omega E\,]\,]$. The proposition follows immediately. □

Subsuming the notion of foundational extensor given in Definition 48 is the notion of foundational chain, that needs to be made explicit in order to easily investigate the properties of the foundational extensors. Given a formal logic program $\mathcal{P}$ and a literal marker Ω for $(\mathcal{P}, \Omega)$, a foundational chain for $(\mathcal{P}, \Omega)$ can be described as follows.

- Start with a supporting extensor E_0 for $(\mathcal{P}, \Omega)$.
- Propose a supporting extensor E_1 for $(\mathcal{P} +_\Omega E_0, \Omega + E_0)$.
- Propose a supporting extensor E_2 for $(\mathcal{P} +_\Omega E_0 \cup E_1, \Omega + E_0 \cup E_1)$.
- Etc.

Formally, this translates into the following definition.

Definition 66. Let a formal logic program $\mathcal{P}$ and a literal marker Ω for $\mathcal{P}$ be given.

Given an ordinal α, an *α-foundational chain for* $(\mathcal{P}, \Omega)$ is a sequence $(E_\beta)_{\beta<\alpha}$ of sets of literals over $\mathcal{V}^\star$ such that for all $\beta < \alpha$, E_β is a supporting extensor for $(\mathcal{P} +_\Omega \bigcup_{\gamma<\beta} E_\gamma, \Omega + \bigcup_{\gamma<\beta} E_\gamma)$.

A *foundational chain for* $(\mathcal{P}, \Omega)$ is a sequence $(E_\alpha)_{\alpha\in\mathrm{Ord}}$ of sets of literals over $\mathcal{V}^\star$ such that for all ordinals α, $(E_\beta)_{\beta<\alpha}$ is an α-foundational chain for $(\mathcal{P}, \Omega)$.

Let a formal logic program $\mathcal{P}$ and a literal marker Ω for $\mathcal{P}$ be given. By Definitions 48 and 66,

- for all ordinals α and α-foundational chains $(E_\beta)_{\beta<\alpha}$ for $(\mathcal{P}, \Omega)$, $\bigcup_{\beta<\alpha} E_\beta$ is an α-foundational extensor for $(\mathcal{P}, \Omega)$ and for all foundational chains $(E_\alpha)_{\alpha\in\mathrm{Ord}}$ for $(\mathcal{P}, \Omega)$, $\bigcup_{\alpha\in\mathrm{Ord}} E_\alpha$ is a foundational extensor for $(\mathcal{P}, \Omega)$;
- for all ordinals α and for all α-foundational extensors E for $(\mathcal{P}, \Omega)$, there exists an α-foundational chain $(E_\beta)_{\beta<\alpha}$ for $(\mathcal{P}, \Omega)$ such that $E = \bigcup_{\beta<\alpha} E_\beta$ and for all foundational extensors E for $(\mathcal{P}, \Omega)$, there exists a foundational chain $(E_\alpha)_{\alpha\in\mathrm{Ord}}$ for $(\mathcal{P}, \Omega)$ such that $E = \bigcup_{\alpha\in\mathrm{Ord}} E_\alpha$.

The first essential property of foundational extensors and α-foundational extensors that we have to prove is that they are implicative. This is expressed in Proposition 67 and Corollary 68.

Proposition 67. *For all formal logic programs $\mathcal{P}$, literal markers Ω for $\mathcal{P}$ and ordinals α, all α-foundational extensors for $(\mathcal{P}, \Omega)$ are implicative.*

Proof. Proof is by induction. Let a formal logic program $\mathcal{P}$, a literal marker Ω for $\mathcal{P}$, an ordinal α, and an α-foundational chain $(E_\beta)_{\beta<\alpha}$ for $(\mathcal{P}, \Omega)$ be given. Assume that for all $\beta < \alpha$, $\bigcup_{\gamma<\beta} E_\gamma \subseteq [\,\mathcal{P} +_\Omega \bigcup_{\gamma<\beta} E_\gamma\,]$. If α is a limit ordinal then it follows immediately from Lemma 55 that $\bigcup_{\beta<\alpha} E_\beta \subseteq [\,\mathcal{P} +_\Omega \bigcup_{\beta<\alpha} E_\beta\,]$.

Suppose that α is of the form $\delta + 1$. By inductive hypothesis, $\bigcup_{\gamma<\delta} E_\gamma$ is included in $[\,\mathcal{P} +_\Omega \bigcup_{\gamma<\delta} E_\gamma\,]$. Moreover, it follows from Property 50 that E_δ is a subset of $\big[\mathcal{P} +_\Omega [\,\mathcal{P} +_\Omega \bigcup_{\gamma<\delta} E_\gamma\,] \cup E_\delta\,\big]$. Lemma 56 then implies that $E_\delta \subseteq [\,\mathcal{P} +_\Omega \bigcup_{\gamma\leq\delta} E_\gamma\,]$. □

Corollary 68. *For all formal logic programs $\mathcal{P}$ and literal markers Ω for $\mathcal{P}$, all foundational extensors for $(\mathcal{P}, \Omega)$ are implicative.*

Corollary 69. *For all formal logic programs $\mathcal{P}$, literal markers Ω for $\mathcal{P}$ and foundational extensors E for $(\mathcal{P}, \Omega)$, there exists an ordinal α such that E is an α-foundational extensor for $(\mathcal{P}, \Omega)$.*

The converse to Corollary 69 holds.

Proposition 70. *For all formal logic programs $\mathcal{P}$, literal markers Ω for $\mathcal{P}$ and ordinals α, all α-foundational extensors for $(\mathcal{P}, \Omega)$ are foundational.*

Proof. Let a formal logic program $\mathcal{P}$, a literal marker Ω for $\mathcal{P}$, an ordinal α, and an α-foundational chain $(E_\beta)_{\beta<\alpha}$ for $(\mathcal{P}, \Omega)$ be given. By Propositions 65 and 67, $[\,\mathcal{P} +_\Omega \bigcup_{\beta<\alpha} E_\beta\,]$ is a supporting extensor for $(\mathcal{P}, \Omega)$. For all $\beta \geq \alpha$, set $E_\beta = \varnothing$. Obviously, $(E_\beta)_{\beta\in\mathrm{Ord}}$ is a foundational chain for $(\mathcal{P}, \Omega)$. The proposition follows immediately. □

Given a formal logic program $\mathcal{P}$, the next proposition will allow us to relate our framework to the well founded semantics for $\mathcal{P}$ either in terms of a particular foundational extensor E for $(\mathcal{P}, \langle - \rangle_\mathcal{P})$, or in terms of $[\,\mathcal{P} +_{\langle - \rangle_\mathcal{P}} E\,]$ for a particular foundational extensor E for $(\mathcal{P}, \langle - \rangle_\mathcal{P})$.

Proposition 71. *Let a formal logic program $\mathcal{P}$, a literal marker Ω for $\mathcal{P}$, and a foundational extensor E for $(\mathcal{P}, \Omega)$ be given. Then all subsets of $[\,\mathcal{P} +_\Omega E\,]$ that contain E are foundational extensors for $(\mathcal{P}, \Omega)$.*

Proof. Let a foundational chain $(E_\alpha)_{\alpha\in\mathrm{Ord}}$ for $(\mathcal{P}, \Omega)$ be given. Let X be a subset of $[\,\mathcal{P} +_\Omega E\,]$ that contains E. Let $(F_\alpha)_{\alpha\in\mathrm{Ord}}$ be the family of literals over $\mathcal{V}^\star$ such that for all ordinals α, $F_{2\alpha}$ equals $\bigcup_{\beta<\alpha} E_\beta$ and $F_{2\alpha+1}$ equals $[\,\mathcal{P} +_\Omega \bigcup_{\beta<\alpha} E_\beta\,] \cap X$. Using Lemmas 55 and 56, it is easy to verify that $(F_\alpha)_{\alpha\in\mathrm{Ord}}$ is a foundational chain for $(\mathcal{P}, \Omega)$ and $\bigcup_{\alpha\in\mathrm{Ord}} F_\alpha = [\,\mathcal{P} +_\Omega E\,] \cap E$. □

We now state a counterpart to Corollary 58 for foundational chains.

Proposition 72. *Let a formal logic program $\mathcal{P}$ be locally consistent. Let a literal marker Ω for $\mathcal{P}$ and a set I be given. Let a set of foundational chains for $(\mathcal{P}, \Omega)$ of the form $\{(E^\sigma_\alpha)_{\alpha\in\mathrm{Ord}} \mid \sigma \in I\}$ be given. Then $(\bigcup_{\sigma\in I} E^\sigma_\alpha)_{\alpha\in\mathrm{Ord}}$ is a foundational chain for $(\mathcal{P}, \Omega)$ iff $\bigcup_{\sigma\in I} \bigcup_{\alpha\in\mathrm{Ord}} E^\sigma_\alpha$ is consistent.*

Proof. Only one direction of the proposition requires a proof. The argument is by induction. For all $\alpha \in \mathrm{Ord}$, set $F_\alpha = \bigcup_{\sigma\in I} E^\sigma_\alpha$. Assume that $\bigcup_{\alpha\in\mathrm{Ord}} F_\alpha$ is consistent. Let an ordinal α be given, and assume that for all $\beta < \alpha$, $(F_\gamma)_{\gamma<\beta}$ is a β-foundational chain for $(\mathcal{P}, \Omega)$. If α is a limit ordinal then $(F_\beta)_{\beta<\alpha}$ is an α-foundational chain for $(\mathcal{P}, \Omega)$ by definition. Assume that α is of the form $\delta + 1$. It follows from Proposition 67 that for all members σ of I, the

set $[\,\mathcal{P} +_\Omega \bigcup_{\gamma<\delta} F_\gamma\,] \cup E^\sigma_\delta$ contains $\bigcup_{\gamma\leq\delta} E^\sigma_\gamma$. Hence $[\,\mathcal{P} +_\Omega \bigcup_{\gamma<\delta} F_\gamma\,] \cup F_\delta$ is a union of supporting extensors for $(\mathcal{P}, \Omega)$. So to complete the proof of the proposition, it suffices by Corollary 58 to show that $[\,\mathcal{P} +_\Omega \bigcup_{\gamma<\delta} F_\gamma\,] \cup F_\delta$ is consistent. By Lemma 55 and Proposition 67, $\bigcup_{\gamma\leq\delta} F_\gamma \subseteq [\,\mathcal{P} +_\Omega \bigcup_{\gamma\leq\delta} F_\gamma\,]$. Hence by Proposition 57, $[\,\mathcal{P} +_\Omega \bigcup_{\gamma\leq\delta} F_\gamma\,]$ is consistent. By Lemma 55 again, $[\,\mathcal{P} +_\Omega \bigcup_{\gamma\leq\delta} F_\gamma\,]$ is a superset of $[\,\mathcal{P} +_\Omega \bigcup_{\gamma<\delta} F_\gamma\,]$, as well as a superset of $[\,\mathcal{P} +_\Omega \bigcup_{\gamma\leq\delta} E^\sigma_\gamma\,]$ for all $\sigma \in I$. Moreover, Proposition 67 implies that for all $\sigma \in I$, $E^\sigma_\delta \subseteq [\,\mathcal{P} +_\Omega \bigcup_{\gamma\leq\delta} E^\sigma_\gamma\,]$. So we conclude that $[\,\mathcal{P} +_\Omega \bigcup_{\gamma<\delta} F_\gamma\,] \cup F_\delta$, being a subset of a consistent set, is consistent. □

As an application of Proposition 72, we can follow the main path in the field of logic programming, be biased towards negative information, and get the following corollary.

Proposition 73. *For all locally consistent formal logic programs $\mathcal{P}$, there exists a $\subseteq$-maximal foundational extensor for $(\mathcal{P}, \langle - \rangle_\mathcal{P})$.*

Proof. Let a locally consistent formal logic program $\mathcal{P}$ be given. By Proposition 72, there exists a $\subseteq$-maximal set E of negated atoms over $\mathcal{V}^\star$ that is a foundational extensor for $(\mathcal{P}, \langle - \rangle_\mathcal{P})$. By Proposition 71, $[\,\mathcal{P} +_{\langle - \rangle_\mathcal{P}} E\,]$ is a foundational extensor for $(\mathcal{P}, \langle - \rangle_\mathcal{P})$. Moreover, it is easy to verify that for all extensors F for $(\mathcal{P}, \langle - \rangle_\mathcal{P})$, all closed instances of negated atoms in F are instances of members of E, and that all closed instances of all members of F belong to $[\,\mathcal{P} +_{\langle - \rangle_\mathcal{P}} E\,]$. □

4.7 Relationship to the well-founded semantics

The well founded semantics takes the class of sets of positive rules as object of study; so again, it is legitimate to study the well founded semantics on the basis of the class of symmetric formal logic programs. But we will see that the hypothesis of symmetry is too strong: it is enough to focus on locally consistent formal logic programs. If one assumes that $\mathcal{V}^\star$ is equal to $\mathcal{V}$, and if one remains in the realm of symmetric formal logic programs, then Definition 62 captures the notion of well founded model. Here not assuming that $\mathcal{V}^\star$ and $\mathcal{V}$ offers a genuine generalization.

Definition 74. Let a formal logic program $\mathcal{P} = \big((\varphi^+_\wp, \varphi^-_\wp)\big)_{\wp\in\mathrm{Prd}(\mathcal{V}^\star)}$ be given. Define two sequences $(E^+_\alpha)_{\alpha\in\mathrm{Ord}}$ and $(E^-_\alpha)_{\alpha\in\mathrm{Ord}}$ of sets of literals as follows. Let $\alpha \in \mathrm{Ord}$ be given, and assume that E^+_β and E^-_β have been defined for all $\beta < \alpha$.

- E^+_α is defined as the set of closed instances of literals φ over $\mathcal{V}^\star$ with

$$\big\{\odot^-_{\bigcup_{\beta<\alpha} E^-_\beta} \varphi^+_\wp \to \wp(v_1, \ldots, v_n) \;\big|\; n \in \mathbb{N},\, \wp \in \mathrm{Prd}(\mathcal{V}^\star, n)\big\} \vDash_\mathcal{W} \varphi.$$

- E^-_α is defined as the set of closed instances of the $\subseteq$-largest set X of negated atoms over $\mathcal{V}^\star$ such that for all $\varphi \in X$, there exists a member Y of $\mathcal{P}[\varphi]$ such that all closed instances of all members of Y belong to $E^+_\alpha \cup E^-_\alpha$.

The partial interpretation $\bigcup_{\alpha\in\mathrm{Ord}}(E^+_\alpha \cup E^-_\alpha)$ is called the *well-founded model of* $\mathcal{P}$.

Recall that by Proposition 73, we can talk about "the $\subseteq$-maximal foundational extensor for $(\mathcal{P}, \langle-\rangle_\mathcal{P})$" when $\mathcal{P}$ is a locally consistent formal logic program. The next proposition shows that this extensor fully characterizes the notion of well founded model. The proposition does more than embed the well founded semantics into our framework as it encompasses all formal logic programs that are locally consistent rather than just symmetric, and as it does not assume that $\mathcal{V}^\star$ and $\mathcal{V}$ are equal.

Proposition 75. *Let $\mathcal{P}$ be a locally consistent formal logic program, and let E be the $\subseteq$-maximal foundational extensor for $(\mathcal{P}, \langle-\rangle_\mathcal{P})$. Then the set of closed instances of members of E is the well-founded model of $\mathcal{P}$.*

Proof. Let $(E_\alpha)_{\alpha\in\mathrm{Ord}}$ be a foundational chain for $(\mathcal{P}, \langle-\rangle_\mathcal{P})$ with $\bigcup_{\alpha\in\mathrm{Ord}} E_\alpha$ equal to E such that for all foundational chains $(F_\alpha)_{\alpha\in\mathrm{Ord}}$ for $(\mathcal{P}, \langle-\rangle_\mathcal{P})$, if $\bigcup_{\alpha\in\mathrm{Ord}} F_\alpha = E$ then for all ordinals α, $F_\alpha \subseteq E_\alpha$. Inductively define by induction a sequence $(E^+_\alpha)_{\alpha\in\mathrm{Ord}}$ of atoms over $\mathcal{V}^\star$ and a sequence $(E^-_\alpha)_{\alpha\in\mathrm{Ord}}$ of negated atoms over $\mathcal{V}^\star$. Let an ordinal α be given, and assume that E^+_β and E^-_β have been defined for all $\beta < \alpha$. Define E^+_α as the $\subseteq$-minimal set of atoms over $\mathcal{V}^\star$ with the property that for all $n \in \mathbb{N}$, $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$ and terms $t_1, \ldots, t_n$ over $\mathcal{V}^\star$, if

$$\bigcup_{\beta<\alpha}(E^+_\beta \cup E^-_\beta) \cup E^+_\alpha \Vdash \varphi^+_\wp[t'_1/v_1, \ldots, t'_n/v_n]$$

for all closed terms $t'_1, \ldots, t'_n$ such that $(t'_1, \ldots, t'_n)$ is an instance of $(t_1, \ldots .t_n)$ then $\wp(t_1, \ldots, t_n) \in E^+_\alpha$. Define E^-_α as the union of all sets X of negated atoms over $\mathcal{V}^\star$ that have the property that for all $n \in \mathbb{N}$, $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$ and terms $t_1, \ldots, t_n$ over $\mathcal{V}^\star$, if $\neg\wp(t_1, \ldots, t_n) \in X$ then $E^+_\alpha \cup X \Vdash \varphi^-_\wp[t'_1/v_1, \ldots, t'_n/v_n]$ for all closed terms $t'_1, \ldots, t'_n$ such that $(t'_1, \ldots, t'_n)$ is an instance of $(t_1, \ldots .t_n)$. Clearly, the set of closed instances of members of $\bigcup_{\alpha\in\mathrm{Ord}}(E^+_\alpha \cup E^-_\alpha)$ is the well-founded model of $\mathcal{P}$.

It is easy to verify by induction that for all ordinals α, $E^+_\alpha \cup E^-_\alpha$ is included in $[\,\mathcal{P} +_{\langle-\rangle_\mathcal{P}} E\,]$, with all closed instances of members of E^-_α belonging to E_α. Hence the well-founded model of $\mathcal{P}$ is included in the set of closed instances of members of $[\,\mathcal{P} +_{\langle-\rangle_\mathcal{P}} E\,]$.

Conversely, it is easy to verify by induction that for all ordinals α, there exists ordinals δ and δ' such that $\bigcup_{\beta<\alpha} E_\beta \subseteq E^+_\delta \cup E^-_\delta$ and for all $n \in \mathbb{N}$, members $\wp$ of $\mathrm{Prd}(\mathcal{V}^\star, n)$ and terms $t_1, \ldots, t_n$ over $\mathcal{V}^\star$,

- if $\wp(t_1, \ldots, t_n) \in [\,\mathcal{P} +_{\langle-\rangle_\mathcal{P}} \bigcup_{\beta<\alpha} E_\beta\,]$ then for all closed terms $t'_1, \ldots, t'_n$ such that $(t'_1, \ldots, t'_n)$ is an instance of $(t_1, \ldots, t_n)$,

$$E^+_{\delta+\delta'} \cup E^-_{\delta+\delta'} \Vdash \varphi^+_\wp[t'_1/v_1, \ldots, t'_n/v_n];$$

- if $\neg\wp(t_1, \ldots, t_n) \in E_\alpha$ then

$$E^+_{\delta+\delta'} \cup E^-_{\delta+\delta'} \Vdash \varphi^-_\wp[t_1/v_1, \ldots, t_n/v_n].$$

Hence the set of closed instances of members of $[\,\mathcal{P} +_{\langle - \rangle_{\mathcal{P}}} E\,]$ is included in the well-founded model of $\mathcal{P}$.

□

5 Conclusion

We have a presented a framework whose development has been guided by conceptual and formal simplicity. Far from restricting the field, it allows one to retrieve the usual semantics and to tighten their relationships. It has unified all those semantics under the umbrella of a unique semantics coupled with a notion of transformation of a logic program that captures the making of contextual hypotheses. Various constraints on these transformations result in families of "semantics" for the original program, with the well known ones as particular cases. One can explore the other members of those families. For instance, the well founded semantics corresponds to the particular foundational extensor that is maximally biased towards negative information. But the class of foundational extensors is rich; it encompasses an extensor that is maximally biased towards positive information, and infinitely many extensors that vary the overall quantity and the mix of positive and negative information. One can also explore totally different families besides the imperative, implicative, supporting and foundational extensors. There are endless possibilities of potentially interesting extensors.

We have carefully not defined a formal logic program as a set of logical formulas. We have chosen to model the behavior of a set of rules that can fire transfinitely often, hence to provide an operational semantics, which does not require to represent rules as logical formulas. Since the classical logical form of a formal logic program $\mathcal{P}$ does not adequately represent the behavior of $\mathcal{P}$, that classical logical form does not provide a sound basis for a declarative semantics. This is not necessarily because the logical symbols, including negation, should receive a different meaning than they do in the classical setting of first-order logic. An intuitionistic interpretation of the logical symbols would not provide an adequate semantics either, though it has been applied to particular classes of logic programs [4]. Still it is perfectly possible to provide a semantics where the logical symbols keep their classical meaning, and where the behavior of a formal logic program can be described in terms of a classical notion of logical consequence. Indeed, let $\mathcal{P}$ be a consistent formal logic program. We can then show that given a literal φ, $T \models^{\star} \varphi$ iff $\varphi \in [\,\mathcal{P}\,]$, where T is some set of modal statements that is obtained from $\mathcal{P}$ as easily as the classical logical form, and where $\models^{\star}$ is a classical notion of logical consequence applied to interpretations that generalize standard structures, in a framework that generalizes epistemic logic. So there is an easy translation from the language for the operational semantics to the language for the declarative semantics, and conversely, but such a translation is needed; both languages cannot be the same. Another paper will be devoted to this topic in a more general setting where formal logic programs will be generalized to sets of rules with arbitrary bodies and arbitrary heads.

Bibliography

[1] José Júlio Alferes, Luís Moniz Pereira, and Teodor C. Przymusinski. Strong and explicit negation in non-monotonic reasoning and logic programming. In *Logics in artificial intelligence Évora*, volume 1126 of *Lecture notes in computer science*, pages 143–163. Springer-Verlag, 1996.

[2] José Júlio Alferes, Luís Moniz Pereira, and Teodor C. Przymusinski. 'classical' negation in nonmonotonic reasoning and logic programming. *Journal of Automated Reasoning*, 20(1–2):107–142, April 1998.

[3] Krzysztof R. Apt and Roland Bol. Logic programming and negation: a survey. *Journal of Logic Programming*, 19-20(Supplement 1):9–71, May-July 1994.

[4] François Bry. Logic programming as constructivism: a formalization and its application to databases. In *PODS '89: Proceedings of the eighth ACM SIGACT-SIGMOD-SIGART symposium on Principles*, pages 34–50, 1989.

[5] Keith L. Clark. Negation as failure. In Matthew L. Ginsberg, editor, *Readings in nonmonotonic reasoning*, pages 311–325. Morgan Kaufmann Publishers Inc., 1987.

[6] Marc Denecker, Maurice Bruynooghe, and Victor Marek. Logic programming revisited: logic programs as inductive definitions. *ACM Transactions on Computational Logic*, 2(4):623–654, October 2001.

[7] Maarten H. Van Emden and Robert A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the Association for Computing Machinery*, 23(4):733–742, October 1976.

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

[9] Allan Van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *Journal of the Association for Computing Machinery*, 38(3):620–650, July 1991.

[10] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth A. Bowen, editors, *Logic programming: proceedings of the fifth international conference and symposium*, volume 2 of *MIT Press series in logic programming*, pages 1070–1080. MIT Press, 1988.

[11] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3–4):365–385, 1991.

[12] John W. Lloyd. *Foundations of logic programming.* Symbolic computation, artificial intelligence. Springer-Verlag, second edition, 1987.

[13] Wiktor Marek and Miroslaw Truszczyński. Autoepistemic logic. *Journal of the Association for Computing Machinery*, 38(3):587–618, July 1991.

[14] Eric Martin. Quantification over names and modalities. In Guido Governatori, Ian Hodkinson, and Yde Venema, editors, *Advances in Modal logic*, volume 6, pages 353–372. College Publications, 2006.

[15] Jack Minker. An overview of nonmonotonic reasoning and logic programming. *Journal of Logic Programming*, 17(2–4):95–126, November 1993.

[16] Robert C. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25(1):75–94, January 1985.