TLP **12** (6): 843–887, 2012. © Cambridge University Press 2011
doi:10.1017/S1471068411000378 First published online 25 July 2011

Contextual hypotheses and semantics of logic programs

ÉRIC A. MARTIN
School of Computer Science and Engineering,
The University of New South Wales,
Sydney, NSW 2052, Australia
(*e-mail:* `emartin@cse.unsw.edu.au`)

submitted 12 October 2009; revised 5 May 2010; accepted 28 March 2011

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 the answer-set semantics standing out as the most successful. We show that all aforementioned semantics are particular cases of a generic semantics, in a framework where classical negation is the unique form of negation and where the literals in the bodies of the rules can be 'marked' to indicate that they can be the targets of hypotheses. A particular semantics then amounts to choosing a particular marking scheme and choosing a particular set of hypotheses. When a literal belongs to the chosen set of hypotheses, all marked occurrences of that literal in the body of a rule are assumed to be true, whereas the occurrences of that literal that have not been marked in the body of the rule are to be derived in order to contribute to the firing of the rule. Hence the notion of hypothetical reasoning that is presented in this framework is not based on making global assumptions, but more subtly on making local, contextual assumptions, taking effect as indicated by the chosen marking scheme on the basis of the chosen set of hypotheses. Our approach offers a unified view on the various semantics proposed in logic programming, classical in that only classical negation is used, and links the semantics of logic programs to mechanisms that endow rule-based systems with the power to harness hypothetical reasoning.

KEYWORDS: Kripke–Kleene semantics, answer-set semantics, stable model semantics, well-founded semantics, classical negation, contextual hypotheses, hypothetical reasoning

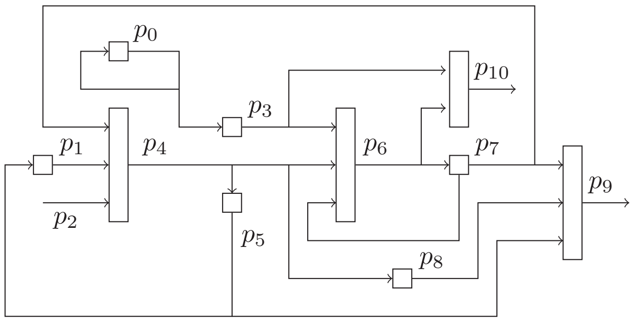
1 Motivation

In this paper, we present a small part of a general framework called parametric logic, some of whose concepts have very practical motivations; the notion of *contextual hypothesis* and the associated notion of *hypothetical reasoning* are two concepts of this kind. A contextual hypothesis, according to which a condition can be assumed to be true in some specific contexts rather than globally, is important in web search, as the information sought by users occurs in documents and is found to be relevant, thanks to the contextual relationships it bears to the input keywords. Hypothetical reasoning is relevant to the development of decision support systems. Effective systems must give users the power to explore and experiment so that they can better understand the domain; they must provide the degree of control that users demand, which the type of hypothetical reasoning to be described here offers. We commence the paper with a simple worked example that introduces the practical aspects of the notions of contextual hypothesis and hypothetical reasoning, and motivates the formal material that follows.

Many decision support systems encode expert knowledge as a set of rules. In practice, knowledge bases may have to deal with thousands of rules. Keeping that order of magnitude in mind, imagine a toy example of a knowledge base consisting of rules all of the form *if condition*$_1$ *and* …*and condition*$_n$ *then conclusion* abstracted as follows:

$$\begin{array}{llll}
p_0 \leftarrow p_0 & p_1 \leftarrow p_5 & p_3 \leftarrow p_0 & p_4 \leftarrow p_1 \wedge p_2 \wedge p_7 \\
p_5 \leftarrow p_4 & p_6 \leftarrow p_3 \wedge p_4 \wedge p_7 & p_7 \leftarrow p_6 & p_8 \leftarrow p_4 \\
p_9 \leftarrow p_5 \wedge p_7 \wedge p_8 & p_{10} \leftarrow p_3 \wedge p_6 & &
\end{array}$$

We could consider a more general set of rules in which some conclusions would be associated with more than one conjunction of conditions, but that would not bring any additional insight. Some of the conditions and conclusions could also carry out negative information, hence be of the form $\neg p$; this example only uses positive information in order to simplify notation at no conceptual cost, since what will be said of the previous set of rules would be said mutatis mutandis of a set of rules that also encodes negative information (the fact that we treat positive and negative information similarly is one of the hallmarks of our approach, as will be seen throughout the paper). The above set of rules can be represented by the following diagram, which can be read as a Boolean circuit with nothing but *and* gates (with a more general set of rules, we would also have *or* gates, and some conditions would be preceded by a *not* gate).



An important feature of the circuit is that it models a reactive process: it contains a number of loops, which reveal circular arguments. This does not necessarily indicate that the representation of knowledge is flawed. For instance, the deflationary economic model posits that a drop in prices delays consumption, which increases inventories, which forces companies to sell their stock at a lower price. Knowledge representation, when applied to domains where amplifiers and reinforcement mechanisms are at work, usually results in knowledge bases with loops.

In order to establish a correct diagnosis or take the right course of actions, a user will often want to query the decision support system on how a given conclusion can be derived; logically speaking, this is a form of *abductive reasoning*. Getting back to our example, let us query how p_9 can be derived. The system could provide a (possibly minimal) set X of pieces of information that, added to the set of rules, permit p_9 to be derived; the members of X would then be assumed to be true *globally*. But the system can do better. It can provide a (possibly minimal) set X of pieces of information that, supposed to be true only in some rules, at some locations, make the resulting, stronger set of rules able to derive p_9; the members of X are then assumed to be true *locally*. Moreover, under this scenario, the system can also indicate which conditions can be *confirmed* (inferred alongside p_9). With our running example, this can be done in many different ways. For instance, using check marks to indicate which occurrences of conditions to select and imposing that they be minimal, the system could return

$$\begin{array}{llll}
p_0 \leftarrow p_0 & p_1 \leftarrow \underset{\checkmark}{p_5} & p_3 \leftarrow p_0 & p_4 \leftarrow p_1 \wedge \underset{\checkmark}{p_2} \wedge \underset{\checkmark}{p_7} \\
p_5 \leftarrow p_4 & p_6 \leftarrow \underset{\checkmark}{p_3} \wedge p_4 \wedge \underset{\checkmark}{p_7} & p_7 \leftarrow p_6 & p_8 \leftarrow p_4 \\
p_9 \leftarrow p_5 \wedge p_7 \wedge p_8 & p_{10} \leftarrow p_3 \wedge p_6 & &
\end{array}$$

and indicate that making p_2, p_3, p_5, and p_7 true at the selected locations allows one to infer p_9 and confirm p_5 and p_7, but neither p_2 nor p_3. Or it could return

$$\begin{array}{llll}
p_0 \leftarrow p_0 & p_1 \leftarrow p_5 & p_3 \leftarrow p_0 & p_4 \leftarrow p_1 \wedge \underset{\checkmark}{p_2} \wedge p_7 \\
p_5 \leftarrow \underset{\checkmark}{p_4} & p_6 \leftarrow \underset{\checkmark}{p_3} \wedge \underset{\checkmark}{p_4} \wedge p_7 & p_7 \leftarrow \underset{\checkmark}{p_6} & p_8 \leftarrow p_4 \\
p_9 \leftarrow p_5 \wedge p_7 \wedge p_8 & p_{10} \leftarrow p_3 \wedge p_6 & &
\end{array}$$

and indicate that making p_2, p_3, p_4, and p_6 true at the selected locations allows one to infer p_9 and confirm p_4 and p_6, but neither p_2 nor p_3. Since they are not confirmed, p_2 and p_3 make it possible to derive p_9 by playing a 'foundational' role, and their marked occurrences indicate where that role is played in the underlying derivation of p_9. On the other hand, p_5 and p_7 (first marked set of rules), or p_4 and p_6 (second marked set of rules), being confirmed, make it possible to derive p_9, thanks to relationships of 'interdependence', and their marked occurrences indicate which rules use them as hypotheses in the underlying derivation of p_9 before these relationships take effect and the hypotheses become unnecessary as they get confirmed[1]. An output of this kind is of great interest to users who, given a confirmed selected condition φ, can investigate further the feedbacks in which φ is involved, which might result in valuable findings or prompt users to amend the knowledge base—they will be

[1] The minimality constraint is essential in this discussion: if the rules were $q_3 \leftarrow q_2$ and $q_2 \leftarrow q_1$, the aim was to derive q_3, and the system returned $q_3 \leftarrow \underset{\checkmark}{q_2}$ and $q_2 \leftarrow \underset{\checkmark}{q_1}$, then q_2 would be confirmed though q_2 is not in a relationship of interdependence to itself.

all the more prepared to this eventuality the number of rules is larger. Of course, a well-designed system will assist in this task.

Other scenarios of interest are possible. Users could first select some occurrences of conditions to indicate the contexts in which those conditions can be assumed to be true, and then list some of those conditions. For instance, users could select, in some rules, some occurrences of the conditions p_1, p_5, and p_6, before listing successively p_1, then p_1 and p_5, then p_1 and p_6, then p_1, p_5, and p_6, to 'activate' first only some, and eventually all, selected occurrences of conditions, and find out what the implications are, what can or cannot be derived as more or fewer assumptions are made in the preselected contexts. Or users could first list conditions, say p_1, p_5, and p_6, and then experiment by selecting various occurrences of those conditions, starting, for instance, with all of them (so ignoring the context), and then removing some occurrences, hence taking the context into account to find out how that affects the conclusions that can be derived, or the conditions that can be confirmed. Extra constraints can be imposed on which conditions should be confirmed or not, or on the relationships between confirmed conditions, etc.

What does all this have to do with the semantics of logic programs, which is what this paper focuses on? Well, we will see that the notions of hypothetical reasoning and contextual hypothesis can do more than enrich the field of logic programming. They allow one to look at its fundamental semantics from a novel perspective. We will see that these fundamental semantics can be all unified under the umbrella of contextual and hypothetical reasoning. They correspond to particular, highly constrained, ways of selecting occurrences of conditions and of choosing hypotheses. The notion of confirmation plays a pivotal role in one semantics (the well-founded semantics). In the previous example, conditions and conclusions were all positive, making it impossible to derive a contradiction. When conditions and conclusions can be negative, a notion of *nonrefutation* naturally enters the stage to express that contextually hypothesising p does not allow one to infer $\neg p$, or that contextually hypothesising $\neg p$ does not allow one to infer p. The notion of nonrefutation plays a crucial role in other semantics (the stable model and the answer-set semantics). Revisiting the fundamental semantics of logic programs under the light of hypothetical reasoning and contextual hypothesis is of conceptual and theoretical interest; in particular, it supports the view that, in contrast to the traditional work in the field, logic programming does not need nonclassical negation, and that positive and negative information can obey a duality principle. As importantly, we can capitalise on the fact that the traditional semantics have been extensively studied and are very well understood, and be confident that expressing them as particular forms of hypothetical reasoning will help understand the latter and come up with valuable constraints and fruitful strategies to exploit it fully.

We have introduced the key idea of choosing a set X of conditions and contextually selecting in some rules some occurrences of conditions, which will be made 'active' if they belong to X. Formalising this idea precisely, omitting no detail, in a very general setting (in particular because it is first-order rather than propositional) requires a bit of work, but the informal description where check marks are used to capture the notion of contextual hypothesis actually says it all and does not hide any essential technicality. We will use this description again. It does not only support the intuition and illustrate the mathematical developments. It actually suggests a very practical, concrete interface, where users click on some occurrences of conditions, activating or deactivating them as they interact with their decision support system. But users will need to be guided, they should not click arbitrarily, they will need and request some constraints that will help them experiment effectively and beneficially. One can imagine, for instance, that some occurrences of conditions are dimmed out in real time to indicate that under the present circumstances, they are 'unclickable'. This paper will not dwell into these considerations; the first task is to provide the theoretical foundations to practical problems of this kind.

2 Background

Since its inception, the field of logic programming has embraced an increasingly complex diaspora of rules, that is, pairs of formulas referred to as 'body' and 'head', with the intention that if the body is true then the head is true. The bodies of the rules have eventually been allowed to contain both classical negation and nonclassical negation—classical negation being used to assert falsity, and nonclassical negation, a form of nonprovability. Classical negation and disjunction have made their ways into the heads of the rules (see Minker and Seipel 2002 for a survey). It has even been advocated to use more than two kinds of negation (Alferes *et al.* 1996; Alferes *et al.* 1998). Also, a large number of constraints on the rules that make up a logic program have been proposed, based on syntactic constraints or definability properties (e.g., Jäger and Stärk 1993) or on proof-theoretic criteria (e.g., Pedreschi *et al.* 2002). All these developments took place as part of the advances in the field of nonmonotonic reasoning (Minker 1993).

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, which 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, which give every individual a unique name. Alongside the various model-theoretic semantics, proof-theoretic techniques and fixed-point constructions have been developed (see Apt and Bol 1994 for a survey). As the number of approaches increased, a natural line of research has been to exhibit possible relationships between the various frameworks and seek unifications, with (Loyer *et al.* 2003) and (Hitzler 2005) as examples of work conducted in the last decade. This study belongs to that category of papers, but differs from previous work in many essential ways:

- It offers a model of hypothetical reasoning for knowledge-based systems where a hypothesis is not conceived of globally as a new fact, but as a statement meant to be assumed locally and contextually (at some locations in the bodies of some rules), which can be subjected to confirmation (if that statement is eventually derived), or subjected to not being refuted (if no rule ever produces its negation), or subjected to other constraints, possibly involving the whole set of chosen hypotheses, or some particular subset.

- Rather than seeking relationships between various semantics of logic programs, it proposes a natural base semantics, and complements it with a generic notion of transformation of a logic program. So rather than proposing, for a given logic program $\mathscr{P}$, a picture of the form

$$\begin{tikzcd} & \text{Semantics 1 of } \mathcal{P} \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & \\ \text{Semantics 2 of } \mathcal{P} \arrow[rr, leftrightarrow] & & \text{Semantics 3 of } \mathcal{P} \end{tikzcd}$$

it proposes a picture of the form

$$\begin{tikzcd} & \text{Base semantics of } \mathcal{P} \arrow[dl] \arrow[d] \arrow[dr] & \\ \begin{array}{l}\text{Base semantics}\\ \text{of } \mathcal{P} +_{\Omega_1} E_1\end{array} & \begin{array}{l}\text{Base semantics}\\ \text{of } \mathcal{P} +_{\Omega_2} E_2\end{array} & \begin{array}{l}\text{Base semantics}\\ \text{of } \mathcal{P} +_{\Omega_3} E_3\end{array} \end{tikzcd}$$

where $\mathscr{P} +_\Omega E$ represents the transformation of $\mathscr{P}$ into a new logic program the bodies of whose rules are possibly weaker than the bodies of the corresponding rules of $\mathscr{P}$, thanks to a construction that uses a set E of literals conceived of as potential hypotheses and a set Ω of occurrences of literals in the bodies of $\mathscr{P}$'s rules conceived of as possible targets of the hypotheses (these notions and others used in the semi-formal presentation of Section 2 will be precisely defined from Section 3 onward). Semantics 1, 2, and 3 of $\mathscr{P}$ then correspond to particular choices of E and Ω, and intuitively receive the interpretation: *in the bodies of $\mathscr{P}$'s rules, make use of the hypotheses in E locally and contextually as indicated by Ω, and apply the base semantics.*
- It attains a high degree of unification between the semantics of logic programs considered in this paper, namely, Kripke–Kleene semantics (Fitting 1985), the well-founded semantics (Gelder *et al.* 1991), the stable model semantics (Gelfond and Lifschitz 1988), and the answer-set semantics (Gelfond and Lifschitz 1991). Kripke–Kleene semantics is closely related to our base semantics, while each of the other three is obtained by instantiating general principles (constraints on Ω and E) that determine families of semantics. Other families would be determined by other principles. Some members of those families, different to the particular members of the particular families considered here, might be worth investigating and have practical use.
- It is classical, in the sense that it uses a unique form of negation, interpreted classically, and is based on interpretations that assign one of the classical truth values of true or false to every formula. This is in contrast to many approaches, for instance, frameworks based on Belnap's four-valued logic (see Fitting 1999),

or on the extension of the logic of here-and-there, N_5, with its five truth values (see Pearce 2006).

- Using classical negation as unique form of negation, it remains outside the realm of nonmonotonic reasoning. It is monotone in both Ω and E: increasing the targets of hypotheses or the set of potential hypotheses results in stronger programs, which generate more literals.
- It is symmetric, in the sense that it treats negated atoms and atoms on a par, and emphasises that the stable model and well-founded semantics, dedicated to interpreting a nonclassical form of negation, give rise to semantics that are fully biased toward negated atoms, while the general principle underlying these semantics is consistent with being totally biased toward nonnegated atoms, or with being committed to achieving a balance between negated and nonnegated atoms.
- It applies to general sets of rules, whose heads can be negated atoms and whose bodies can (but do not have to) be arbitrary infinitary first-order formulas.
- It does not require that intended interpretations be restricted to the class of Herbrand interpretations.

2.1 Two key principles

Our framework 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 (Clark 1987). 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 unique name axioms. Our formalisation is 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 inference mechanism 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 (Fitting 1985). 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, the stable model semantics, and the answer-set semantics. 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, which we introduce now. Consider two formulas, φ, of the form

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

and ψ, of the form

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

where $\doteq$ represents identity (denotation by the same closed term of two individuals). 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 hypothesise 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. More generally, we will formalise the notion of 'transforming a formula into another by making some contextual hypotheses in the former', similar 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' toward 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 $\mathscr{P}$ 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 $\mathscr{P}$, the well-founded semantics makes all of $p(\overline{n})$, $n \in \mathbb{N}$, false in its intended model of $\mathscr{P}$, 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, which will be formalised in the body of the paper, consistent with a large number of models of $\mathscr{P}$, including in particular

- structures in which $p(\overline{n})$ is false for all ns;
- structures in which $p(\overline{n})$ is true for all ns;

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

So this more general principle isolates a number of Herbrand models one of which is maximally biased toward negative information, which happens to be the intended model advocated by the well-founded semantics; but this more general principle can be instantiated to 'cousin semantics' of the well-founded semantics, some of which could be of interest. One could be maximally biased toward positive information—a form of dual well-founded semantics—or one could try and keep a balance between positive and negative information.

2.2 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 amounts 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 the answer-set semantics into our framework. We also have a denotational semantics but will make it the subject of another 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 *head* $\leftarrow$ *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 the answer-set semantics. We will see that it is also natural in relation to Kripke–Kleene, the well-founded and the stable model semantics, thanks to the notions of *duality* of a formula and of *symmetry* of a logic program, which we introduce now. 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 deleting all negation signs (before 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 $\mathscr{P}$ 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 symbols $\wp$, $\mathscr{P}$ contains exactly two rules of respective form $\wp(v_1, \dots, v_n) \leftarrow \varphi^+_\wp$ and $\neg\wp(v_1, \dots, v_n) \leftarrow \varphi^-_\wp$ 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 Kripke–Kleene semantics, the well-founded semantics and the stable model semantics as applied to symmetric logic programs. The working hypothesis is that all three semantics *do* 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, first, negation normal form is not restrictive; second, a straightforward syntactic transformation allows one to merge all rules whose heads are built from a given predicate symbol; and third, 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 (Emden and Kowalski 1976; Denecker *et al.* 2001). So after all rules have fired any finite number of times, they could fire for the ωth 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 stage $5n$, 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. Formalising the process by which rules fire transfinitely often determines the set of literals $[\mathscr{P}]$ generated by a set $\mathscr{P}$ of (positive and negative) rules. It is an operational semantics, previously referred to as the base semantics of $\mathscr{P}$. No other semantics will be proposed: what is presented as an alternative semantics of $\mathscr{P}$ will be viewed as the base semantics of a program obtained from $\mathscr{P}$ in a particular way, which captures the essence of the alternative semantics and is an instance of a generic class of transformations.

2.3 Making contextual hypotheses

Let us describe a bit more precisely the relationships between the base semantics and the well-founded semantics, the stable model semantics and the answer-set semantics. Consider a set E of literals. Also consider a function Ω, defined on the set of bodies of the rules in $\mathscr{P}$, which returns, for the body φ of each rule in $\mathscr{P}$, a selected set of occurrences of literals in φ. We can represent this function graphically by 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 $\mathscr{P}$ a new set of rules, denoted $\mathscr{P} +_\Omega E$, which formalises the intuitive request: 'in the bodies of the rules of $\mathscr{P}$, use E as a set of hypotheses in the contexts indicated by Ω'. For instance, if $\mathscr{P}$ 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 $\mathscr{P} +_\Omega E$ 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 $\mathscr{P}$ is symmetric, we can choose Ω and E in such a way that $[\mathscr{P} +_\Omega E]$ captures the well-founded semantics applied to the positive rules of $\mathscr{P}$; moreover, this choice of Ω and E is a particular case of choices made according to a simple principle, which happens to be maximally biased toward negative information. Still in case $\mathscr{P}$ is symmetric, we can also choose Ω and E in ways such that $[\mathscr{P} +_\Omega E]$ are the stable models of the positive rules of $\mathscr{P}$; similarly, these choices of Ω and E are particular cases of choices made according to a simple principle, which happen to be maximally biased toward negative information. Importantly, these correspondences are between a framework where negation is classical and frameworks where negation is meant not to be classical. The answer-set semantics 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*. Given a set of rules $\mathscr{P}$ in the bodies of which both $\neg$ and *not* might occur, with *not* applied only to atoms and classical negations of atoms, consider the set of rules $\mathscr{P}'$ obtained from $\mathscr{P}$ by replacing all occurrences of *not literal* with $\sim$*literal* (so only classical negation occurs in $\mathscr{P}'$). Then set Ω to select precisely the occurrences of literals in the bodies of the rules of $\mathscr{P}'$ that have replaced an occurrence of an expression of the form *not literal* in the bodies of the corresponding rules of $\mathscr{P}$. For instance, if $\mathscr{P}$ contains the rule

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

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

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

that Ω will mark as

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

It is then easy to choose E in ways such that $[\mathscr{P}' +_{\Omega} E]$ are the answer-sets for $\mathscr{P}$ (one answer-set for each choice of E).

3 Logical background

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

3.1 Syntax

Definition 1
A *vocabulary* is a nonempty set of (possibly nullary) function symbols and (possibly nullary) predicate symbols none of which is the distinguished binary predicate symbol $\doteq$ (identity), such that if $\mathcal{V}$ contains at least one nonnullary predicate or function symbol then $\mathcal{V}$ contains at least one nullary function symbol.

As usual, a constant refers to a nullary function symbol[2]. 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 formalise all notions in a setting that can be either purely propositional, or purely first-order, or hybrid.

Notation 1
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 2
We denote by $\mathcal{V}$ a vocabulary.

Notation 3
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 2
The set $\mathcal{L}_{\omega_1\omega}(\mathcal{V})$ of (*infinitary*) *formulas* (*over* $\mathcal{V}$) is inductively defined as the smallest set that satisfies the following conditions.

[2] Vocabularies that would contain at least one nonnullary predicate or function symbol but no constant would be degenerate in this setting, and are better ruled out, though they would be perfectly legitimate in the usual treatment of first-order logic. But see the discussion at the beginning of Section 4.1 on how full generality is obtained despite the restrictions imposed on vocabularies.

- 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 $\mathcal{L}_{\omega_1\omega}(\mathcal{V})$.
- If $\mathcal{V}$ contains at least one constant then all *identities* and *distinctions* (*over* $\mathcal{V}$), namely, all expressions of the form $t \doteq t'$ or $\neg t \doteq t'$, the latter being usually written $t \not\doteq t'$, where t and t' are terms over $\mathcal{V}$, belong to $\mathcal{L}_{\omega_1\omega}(\mathcal{V})$.
- All expressions of the form $\bigvee X$ or $\bigwedge X$ with X a countable set of formulas over $\mathcal{V}$, belong to $\mathcal{L}_{\omega_1\omega}(\mathcal{V})$.
- All expressions of the form $\exists x\, \varphi$ or $\forall x\, \varphi$ where x is a variable and φ is a formula 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 formulas that have the quantified variables as free variables, and identities and distinctions are ruled out in case $\mathcal{V}$ contains no constant, 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 the infinitary propositional language built on $\mathcal{V}$.
- Third, if we wanted to sometimes restrict some concepts to finite formulas, then we would still be happy with disjunction and conjunction defined as unary operators on finite sets of formulas. This treatment of disjunction and conjunction, which contrasts with the traditional view of binary operators on pairs of formulas, does more than let $\mathcal{L}_{\omega_1\omega}(\mathcal{V})$ naturally extend 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 $\bigwedge \varnothing$ is valid and can play the role of `true`, and $\bigvee \varnothing$ is invalid and can play the role of `false`.
- Fourth, we distinguish between identity and equality. Identity is treated as a logical symbol, and its interpretation built into the logic, whereas equality is treated as a nonlogical symbol (a possible member of $\mathcal{V}$), whose intended interpretation needs to be explicitly provided. This will be discussed at greater length in Section 4.1.

Let us emphasise that our framework does not need the power of infinitary languages. Readers interested only in finite logic programs can ignore the qualifier ‘infinitary’ and replace $\mathcal{L}_{\omega_1\omega}(\mathcal{V})$ by $\mathcal{L}_{\omega\omega}(\mathcal{V})$. But $\mathcal{L}_{\omega_1\omega}(\mathcal{V})$ offers an elegant way to work with logic programs consisting of infinitely many rules built from a vocabulary with a finite number of predicate symbols, possibly obtained by grounding a finite logic program expressed in a first-order language whose set of closed terms is infinite, without making any of the formal notions and proofs more complicated than their finitary counterparts.

Let a formula 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 formula or a term. Given $n \in \mathbb{N}$, 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 formulas or terms. We say that e' is a closed instance of e iff there exists $n \in \mathbb{N}$, distinct variables $x_1, \ldots, x_n$ and closed terms $t_1, \ldots, t_n$ such that $x_1, \ldots, x_n$ are the variables that occur free in e and 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 formula in a syntactically friendly manner which is achieved in the following usual way: given a formula φ, $\sim\varphi$ denotes

- $\neg\varphi$ if φ is an atom;
- ψ if φ is of the form $\neg\psi$;
- $t \not\doteq t'$ if φ is of the form $t \doteq t'$;
- $t \doteq t'$ if φ is of the form $t \not\doteq t'$;
- $\bigwedge\{\sim\psi \mid \psi \in X\}$ 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 formulas, we let $\sim X$ denote $\{\sim\varphi \mid \varphi \in X\}$. A set X of literals is said to be *consistent* just in case there is no closed atom φ that is an instance of both a member of X and a member of $\sim X$. A set of literals is said to be *inconsistent* iff it is not consistent. 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 formulas $\varphi_1, \ldots, \varphi_n$, we use $\varphi_1 \vee \cdots \vee \varphi_n$ and $\varphi_1 \wedge \cdots \wedge \varphi_n$ as abbreviations for $\bigvee\{\varphi_i \mid 1 \leqslant i \leqslant n\}$ and $\bigwedge\{\varphi_i \mid 1 \leqslant i \leqslant n\}$, respectively. Also, given two formulas 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: $\leftarrow$ has been used before and will be used later to represent rules in the traditional way, separating the head of a rule from its body. In the operational semantics that is the subject of this paper, $\leftarrow$ is not meant to be interpreted as logical implication.

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

Let a formula 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 3
Given a formula φ and a literal ψ, an *occurrence of* ψ *in* φ is defined as a set O of formulas that contains both φ and ψ and such that

- all members of O are subformulas of φ;
- ψ is a subformula of all members of O;
- for all members ψ_1 and ψ_2 of O, either ψ_1 is a subformula of ψ_2 or ψ_2 is a subformula of ψ_1;
- 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 1
Suppose that $\mathcal{V}$ contains three 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.

3.2 Semantics

Definition 4
Let a set S of literals be given.

For all formulas φ, we inductively define the notion S *forces* φ, denoted $S \Vdash \varphi$, as follows. If S is inconsistent then S forces all formulas. Assume that S is consistent, then

- for all formulas φ, $S \Vdash \varphi$ iff S forces all closed instances of φ;
- 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 $S \Vdash t_1 \not\doteq t_2$ in case t_1 and t_2 are distinct;
- for all closed literals φ, $S \Vdash \varphi$ iff φ is an instance of a member of S;
- for all countable sets X of closed formulas, $S \Vdash \bigvee X$ iff S forces some member of X, and $S \Vdash \bigwedge X$ iff S forces all members of X;
- for all formulas 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 T of formulas, we say that S *forces* T, denoted $S \Vdash T$, just in case S forces all members of T.

Definition 5
A *standard structure* (*over* $\mathcal{V}$) is a set of closed atoms.

Note the following particular cases:

- if $\mathcal{V}$ contains 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.

The following is the usual notion of truth of a formula in a structure, concisely expressed in terms of the notion introduced in Definition 4, which of course is meant to serve other purposes.

Definition 6
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 formulas φ, we say that φ *is true in* $\mathfrak{M}$, or that $\mathfrak{M}$ *is a model of* φ, iff $X \Vdash \varphi$.

Notation 4
Let a standard structure $\mathfrak{M}$ be given. Given a formula φ, we write $\mathfrak{M} \models \varphi$ if $\mathfrak{M}$ is a model of φ, and $\mathfrak{M} \not\models \varphi$ otherwise. Given a set T of formulas, we write $\mathfrak{M} \models T$ if $\mathfrak{M}$ is a model of all members of T, and $\mathfrak{M} \not\models T$ otherwise.

Notation 5
We denote by $\mathcal{W}$ the set of all standard structures (over $\mathcal{V}$).

Given a set T of formulas and a formula φ, we write $T \models_{\mathcal{W}} \varphi$ if every standard model of T is a model of φ; if $T \models_{\mathcal{W}} \varphi$ 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 also applies to sets of formulas. Two formulas φ and ψ are said to be *logically equivalent in* $\mathcal{W}$ iff they have the same models in $\mathcal{W}$.

4 Formal logic programs and their denotational semantics

4.1 *Formal logic programs*

The concepts introduced in the previous section might suggest that we are considering a notion of logical consequence, namely, $\models_{\mathcal{W}}$, which, because of its focus on standard structures, is stronger than the classical notion of logical consequence. To make sure that this is not necessarily the case and achieve full generality, we distinguish between two kinds of vocabularies, namely, a vocabulary meant to *describe* a structure and a vocabulary meant to *talk about* a structure. The vocabulary $\mathcal{V}$ introduced in Notation 2 is of the first kind; it is meant to express what a structure is 'made of', but it might not be 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 6
We denote by $\mathcal{V}^\star$ a (possibly finite) countable subset of $\mathcal{V}$.

$\mathcal{V}^\star$ is the vocabulary to be used when we talk about a structure by writing down theories, axioms, theorems: all must consist of formulas 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 function symbol of arity one or more that is not in $\mathcal{V}^\star$. Then for all sets T of formulas over $\mathcal{V}^\star$ and for all formulas 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 formulas that can be 'spoken out' on the left-hand side and with formulas that can be 'spoken out' on the right-hand side, is equivalent to the classical notion of logical consequence (Martin 2006). 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.

The availability of both $\mathcal{V}$ and $\mathcal{V}^\star$ therefore provides a uniform and simple way to express that a result holds for the classical notion of logical consequence as well as for the more restricted notion of logical consequence that rules out nonstandard structures—by not imposing any condition on $\mathcal{V}^\star$ in the statement of that result—or to force a result to hold for one notion of logical consequence only—by imposing the right condition on the relationship between $\mathcal{V}$ and $\mathcal{V}^\star$. Restricting in different ways a given vocabulary offers some advantages over the more traditional approach of expanding in different ways a given vocabulary, as done in the Henkin proof of the completeness of first-order logic or in so-called Herbrand semantics of first-order logic (see, for instance, Kaminski and Rey 2002).

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 (Lloyd 1987). 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 in the sense that all notions defined in this paper will not require any prior assumption on the relationship between $\mathcal{V}^\star$ and $\mathcal{V}$; but we will sometimes have to suppose that $\mathcal{V}^\star$ is equal to $\mathcal{V}$ in the statements of some results. 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 7
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 formulas can be infinitary and can contain occurrences of $\doteq$, and since $\bigvee\varnothing$ is a formula that can be used as the body of a rule such as $q \leftarrow \bigvee\varnothing$ to express that q is neither a fact nor the head of a rule that can be activated, it is enough to provide, for every $n \in \mathbb{N}$ and $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$, two rules: one whose head is $\wp(v_1, \dots, v_n)$ (which is nothing but $\wp$ if $n = 0$), and one whose head is $\neg\wp(v_1, \dots, v_n)$ (which is nothing but $\neg\wp$ if $n = 0$). For instance, $\{p_1(\overline{2n}) \leftarrow p_2(\overline{2n+1}) \mid n \in \mathbb{N}\}$ can be represented as $p_1(v_1) \leftarrow \bigvee\{p_2(s(v_1)) \wedge v_1 \doteq \overline{2n} \mid n \in \mathbb{N}\}$.

For the purpose of providing, for every $\wp \in \mathrm{Prd}(\mathcal{V}^\star)$, the positive and negative rules associated with $\wp$, and for the purpose of making some definitions more compact, we introduce the following notation (in which $+$ and $-$ could be replaced by 1 and 0, but using $+$ and $-$ will be easier to read).

Notation 8
We let $\mathbb{I}(\mathcal{V}^\star)$ denote $\mathrm{Prd}(\mathcal{V}^\star) \times \{+,-\}$.

Given $n \in \mathbb{N}$, $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$ and terms $t_1, \ldots, t_n$, we also write $\wp(t_1,\ldots,t_n)$ as $\wp^+(t_1,\ldots,t_n)$ and $\neg\wp(t_1,\ldots,t_n)$ as $\wp^-(t_1,\ldots,t_n)$.

Definition 7
We define a *formal logic program* (*over* $\mathcal{V}^\star$) as an $\mathbb{I}(\mathcal{V}^\star)$-family of formulas over $\mathcal{V}^\star$, say $(\varphi_\wp^\epsilon)_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$, such that for all $n \in \mathbb{N}$, $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$ and $\epsilon \in \{+,-\}$, $\mathrm{fv}(\varphi_\wp^\epsilon) \subseteq \{v_1,\ldots,v_n\}$.

Since there is no restriction on the use of quantifiers in the body of a rule, 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 finite rules over $\mathcal{V}^\star$ are naturally translated into finite formal logic programs.

Example 2
Suppose that $\mathcal{V}^\star$ consists of a constant $\overline{0}$, a unary function symbol s, five nullary predicate symbols $q_1, \ldots, q_5$, and four unary predicate symbols $p_1, \ldots, p_4$. The following formulas provide an example of a formal logic program:

$$\begin{aligned}
\varphi_{p_1}^+ &\equiv v_1 \doteq \overline{0} \vee \exists v_0\big(v_1 \doteq s(s(v_0)) \wedge p_1(v_0)\big),\\
\varphi_{p_1}^- &\equiv v_1 \not\doteq \overline{0} \wedge \forall v_0\big(v_1 \not\doteq s(s(v_0)) \vee \neg p_1(v_0)\big),\\[4pt]
\varphi_{p_2}^+ &\equiv v_1 \doteq \overline{0} \vee \exists v_0\big(v_1 \doteq s(s(v_0)) \wedge p_2(v_0)\big),\\
\varphi_{p_2}^- &\equiv v_1 \doteq s(\overline{0}) \vee \exists v_0\big(v_1 \doteq s(s(v_0)) \wedge \neg p_2(v_0)\big),\\[4pt]
\varphi_{p_3}^+ &\equiv v_1 \doteq \overline{0} \vee \exists v_0\big(v_1 \doteq s(v_0) \wedge \neg p_3(v_0)\big),\\
\varphi_{p_3}^- &\equiv \exists v_1\big(v_1 \doteq s(v_0) \wedge p_3(v_0)\big),
\end{aligned}$$

$$\begin{array}{lll}
\varphi_{p_4}^+ \equiv p_4(s(s(v_1))), & \varphi_{q_1}^+ \equiv \bigwedge\varnothing, & \varphi_{q_2}^+ \equiv q_3,\\
\varphi_{p_4}^- \equiv \neg p_4(s(s(v_1))), & \varphi_{q_1}^- \equiv \bigvee\varnothing, & \varphi_{q_2}^- \equiv \neg q_3,\\[4pt]
\varphi_{q_3}^+ \equiv q_2, & \varphi_{q_4}^+ \equiv q_4, & \varphi_{q_5}^+ \equiv \neg q_5,\\
\varphi_{q_3}^- \equiv \neg q_2, & \varphi_{q_4}^- \equiv \bigvee\varnothing, & \varphi_{q_5}^- \equiv \bigvee\varnothing.
\end{array}$$

As $\mathcal{V}^\star$ contains both nullary and nonnullary predicate symbols, Example 2 describes a ‘hybrid’ formal logic program, but of a simple kind as it consists of a purely first-order part and a purely propositional part. Let us take advantage of this example to illustrate how Definition 7 is put to use to represent rules. Recall that $\bigwedge\varnothing$ is valid and $\bigvee\varnothing$ 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 $q_5 \leftarrow \neg q_5$.

Now to the first-order rules.

- The formulas $\varphi^+_{p_i}$, $i \in \{1,2\}$, 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 fact and rule:

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

  So for $i \in \{1,2\}$, $\varphi^+_{p_i}$ allows one to generate all literals of the form $p_i(\overline{2n})$, $n \in \mathbb{N}$. It is easily verified that for $i \in \{1,2\}$, $\varphi^-_{p_i}$ allows one to generate all literals of the form $\neg p_i(\overline{2n+1})$, $n \in \mathbb{N}$. More precisely, the rule represented by $\varphi^-_{p_1}$, 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 very similar to the rule $p_2(v_1) \leftarrow \varphi^+_{p_2}$, and could be rewritten as

  $$\neg p_2(s(\overline{0})),$$
  $$\neg p_2(s(s(v_1))) \leftarrow \neg p_2(v_1)$$

  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 formulas $\varphi^+_{p_3}$ 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 with the positive rule.
- Finally, $\varphi^+_{p_4}$ and $\varphi^-_{p_4}$ represent the rules

  $$p_4(v_1) \leftarrow p_4(s(s(v_1))),$$
  $$\neg p_4(v_1) \leftarrow \neg p_4(s(s(v_1))),$$

  and would not generate any literal.

The second item in Definition 4 captures the unique name axioms that come with the definition of Clark's completion of a logic program. Making $\doteq$ a logical symbol amounts to building into the logic a notion of identity stronger than the usual, less restrictive notion of equality, which in our framework is nonlogical and has to be axiomatised if needed: $=$ is then put into $\mathcal{V}^\star$ and its intended interpretation captured by any formal logic program $(\varphi^\epsilon_\wp)_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$ such that $\varphi^+_=$ is of the form $\bigvee X$ where X is a superset of

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

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

$$\exists v_{n+1} \dots \exists v_{2n}\big(v_1 = v_{n+1} \wedge \dots \wedge v_n = v_{2n} \wedge \wp^\epsilon(v_{n+1},\dots,v_{2n})\big).$$

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 axiomatising equality from $\mathcal{V}^\star$, the vocabulary used to talk about a structure. As a consequence, equality and Herbrand structures are not interdependent: if infinitely many closed terms are 'unspeakable of' then equality as axiomatised above behaves equivalently to the way it behaves w.r.t. the classical notion of logical consequence.

Definition 8
Given a formal logic program $\mathscr{P} = (\varphi^\epsilon_\wp)_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$, the *classical logical form of* $\mathscr{P}$ is defined as

$$\{\varphi^\epsilon_\wp \rightarrow \wp^\epsilon(v_1,\dots,v_n) \mid n \in \mathbb{N},\ \wp \in \mathrm{Prd}(\mathcal{V}^\star, n),\ \epsilon \in \{+,-\}\}.$$

Of course, $\models_\mathcal{W}$ applied to the classical logical form of a formal logic program $\mathscr{P}$ does not adequately capture the logical meaning of $\mathscr{P}$. 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 (Moore 1985; Marek and Truszczyński 1991). We will complete this task in another paper.

Notation 9
Given a formal logic program $\mathscr{P}$, we let $\mathrm{Clf}(\mathscr{P})$ denote the classical logical form of $\mathscr{P}$.

The general logic programs that are the object of Kripke–Kleene semantics, the well-founded semantics and the stable model semantics can 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 9
Let a formal logic program $\mathscr{P} = (\varphi^{\epsilon}_{\wp})_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^{\star})}$ be given. We say that $\mathscr{P}$ is *symmetric* iff for all $\wp \in \mathrm{Prd}(\mathcal{V}^{\star})$, $\varphi^{-}_{\wp}$ is identical to $\sim\varphi^{+}_{\wp}$.

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

Definition 10
We say that a formal logic program $(\varphi^{\epsilon}_{\wp})_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^{\star})}$ 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 1
A symmetric formal logic program is locally consistent.

4.2 Generated literals

The mechanistic view on (the rules of) a formal logic program $\mathscr{P}$ presented in Section 2.2 allows one to talk about the literals over $\mathcal{V}^{\star}$ generated by $\mathscr{P}$; these literals make up a set that we denote by $[\mathscr{P}]$. More precisely, a literal ψ over $\mathcal{V}^{\star}$ is generated by $\mathscr{P}$ and put into $[\mathscr{P}]$ if it is possible to successively fire rules, starting with rules whose body can be unconditionally activated (such as $\bigwedge\varnothing$), till enough literals have been generated and put into $[\mathscr{P}]$ so that there exists a rule in $\mathscr{P}$ 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 $[\mathscr{P}]$; the notation that follows will allow us to easily refer to a formula (determined by ψ and $\mathscr{P}$) of the form $\xi\theta''$ where θ'' is θ with some of the variables in its range being possibly renamed so that none of the variables that occurs in ψ (that is, in $\chi\theta$) is captured when applying the substitution θ'' to ξ.

Notation 10
Let a formal logic program $\mathscr{P}$ be given. For all $n \in \mathbb{N}$, $\wp \in \mathrm{Prd}(\mathcal{V}^{\star}, n)$, terms $t_1, \ldots, t_n$ and $\epsilon \in \{+,-\}$, we let $\mathscr{P}[\wp^{\epsilon}(t_1,\ldots,t_n)]$ denote a formula of the form $\varphi^{\epsilon}_{\wp}[t'_1/v_1,\ldots,t'_n/v_n]$ whose closed instances are precisely the formulas of the form $\varphi^{\epsilon}_{\wp}[t''_1/v_1,\ldots,t''_n/v_n]$ with $(t''_1,\ldots,t''_n)$ any closed instance of $(t_1,\ldots,t_n)$.

In the context of Notation 10, observe that if none of the variables occurring in one of $t_1, \ldots, t_n$ is captured by quantifiers in $\varphi^{\epsilon}_{\wp}$ when simultaneously substituting $v_1, \ldots, v_n$ in $\varphi^{\epsilon}_{\wp}$ by $t_1, \ldots, t_n$, respectively, then a natural choice for $\mathscr{P}[\wp^{\epsilon}(t_1,\ldots,t_n)]$ is $\varphi^{\epsilon}_{\wp}[t_1/v_1,\ldots,t_n/v_n]$ itself. Using Notation 10, one can then concisely define the set of literals over $\mathcal{V}^{\star}$ generated by a formal logic program as a fixed point.

Notation 11
Given a formal logic program $\mathscr{P}$, we denote by $[\mathscr{P}]$ the (unique) $\subseteq$-minimal set of literals over $\mathcal{V}^\star$ that forces $\mathscr{P}[\psi]$ for all $\psi \in [\mathscr{P}]$.

Of course, we could alternatively define $[\mathscr{P}]$ in terms of a transfinite construction and collect at some round, indexed by an ordinal, the set of literals over $\mathcal{V}^\star$ that can be generated from $\mathscr{P}$ by activating some instances of the bodies of some of $\mathscr{P}$'s rules, thanks to the literals generated at previous rounds. This construction is defined in Notation 12, and the fact that it is an alternative definition to $[\mathscr{P}]$ is stated as Property 2.

Notation 12
Let a formal logic program $\mathscr{P}$ be given. Inductively, define a sequence $([\mathscr{P}]_\alpha)_{\alpha\in\mathrm{Ord}}$ of sets of literals over $\mathcal{V}^\star$ as follows. Let an ordinal α be given and assume that for all $\beta < \alpha$, $[\mathscr{P}]_\beta$ has been defined. Then denote by $[\mathscr{P}]_\alpha$ the set of all literals ψ over $\mathcal{V}^\star$ with $\bigcup_{\beta<\alpha}[\mathscr{P}]_\beta \Vdash \mathscr{P}[\psi]$.

Property 2
For all formal logic programs $\mathscr{P}$, $[\mathscr{P}] = \bigcup_{\alpha\in\mathrm{Ord}}[\mathscr{P}]_\alpha$.

Example 3
If $\mathscr{P}$ is the formal logic program of Example 2 then

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

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 3
For all formal logic programs $\mathscr{P}$ and literals ψ over $\mathcal{V}^\star$, $[\mathscr{P}] \Vdash \psi$ iff $\psi \in [\mathscr{P}]$.

Local consistency as introduced in Definition 10 will play a pivotal role in the statements of some propositions, but the more general notion of plain consistency given next is a better counterpart to the classical concept of a consistent theory.

Definition 11
A formal logic program $\mathscr{P}$ is said to be *consistent* just in case $[\mathscr{P}]$ is consistent.

Property 4
Every locally consistent formal logic program is consistent.

Let a formal logic program $\mathscr{P} = (\varphi_\wp^\epsilon)_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$ be given. When $\mathcal{V}^\star = \mathcal{V}$, the definition of $[\mathscr{P}]$ can involve closed literals only—a consequence of Property 3 and the next property. In the general case, $[\mathscr{P}]$ is a set of possibly nonclosed literals, and some rules might fire because their bodies are activated, thanks to 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 $[\mathscr{P}]$ contains $p(v_0)$, hence it contains q. Also, $[\mathscr{P}]$ 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 at least one of $\mathcal{V}$'s constants does not belong to $\mathcal{V}^\star$, then the set of all closed members of $[\mathscr{P}]$ 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 5
Let a formal logic program $\mathscr{P}$ be given. If $\mathcal{V}^\star = \mathcal{V}$ then the set of closed members of $[\mathscr{P}]$ is the (unique) $\subseteq$-minimal set X of closed literals with $X \Vdash \{\mathscr{P}[\psi] \mid \psi \in X\}$.

The classical logical form of a formal logic program $\mathscr{P}$, formalised in Definition 8, does not capture in a satisfactory way the logical meaning of $\mathscr{P}$, but it is still well behaved, as expressed by the property and the corollary that follow.

Property 6
For all formal logic programs $\mathscr{P}$, $\mathrm{Clf}(\mathscr{P}) \vDash_{\mathcal{W}} [\mathscr{P}]$.

Corollary 1
For all formal logic programs $\mathscr{P}$, if $[\mathscr{P}]$ is complete then the set of closed instances of atoms in $[\mathscr{P}]$ is a model of $\mathrm{Clf}(\mathscr{P})$.

4.3 Characterisation of Kripke–Kleene semantics

The characterisation is based on the following definition.

Definition 12
A *partial interpretation* (*over* $\mathcal{V}$) is a consistent set of closed literals.

Kripke–Kleene semantics is usually presented in a three-valued logical setting. The relationship between Definition 12 and a three-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` if $\varphi \in M$, to `false` if $\neg\varphi \in M$, and to a third value or to 'undefined' otherwise. Definition 13 then generalises 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 13
Let a formal logic program $\mathscr{P} = (\varphi^\epsilon_\wp)_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$ be given. A *partial model of* $\mathscr{P}$ is a partial interpretation M such that for all $n \in \mathbb{N}$, $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$, closed terms $t_1, \ldots, t_n$ and $\epsilon \in \{+,-\}$, M contains $\wp^\epsilon(t_1,\ldots,t_n)$ iff M forces $\varphi^\epsilon_\wp[t_1/v_1,\ldots,t_n/v_n]$.

Given a formal logic program $\mathscr{P}$, a $\subseteq$*-minimal partial model of* $\mathscr{P}$ is referred to more simply as a minimal partial model of $\mathscr{P}$. Proposition 1 expresses that the generalisation of Kripke–Kleene semantics given in Definition 13 is equivalent to our base semantics of a consistent formal logic program, 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 1 still applies to more general frameworks as it deals with formal logic programs that might not be symmetric.

Proposition 1
Assume that $\mathcal{V}^\star = \mathcal{V}$. Let a consistent formal logic program $\mathscr{P}$ be given. Then $\mathscr{P}$ has a unique minimal partial model, which is nothing but the set of closed instances of members of $[\mathscr{P}]$.

Proof
Let X denote the set of partial models of $\mathscr{P}$. It is immediately verified that

- the set of closed instances of members of $[\mathscr{P}]$ is included in $\bigcap X$;
- the set of closed instances of members of $[\mathscr{P}]$ belongs to X.

Hence $\bigcap X$, being equal to the set of closed instances of members of $[\mathscr{P}]$, is a partial model of $\mathscr{P}$. $\square$

5 Extensors, and relationships to particular semantics

5.1 Extensors

We now formalise the operation, discussed in Section 2.3, of transforming a formal logic program $\mathscr{P}$ into another formal logic program $\mathscr{P} +_{\Omega} E$, where Ω selects some occurrences of literals in the bodies of $\mathscr{P}$'s rules and E is a set of literals, the intended meaning of $\mathscr{P} +_{\Omega} E$ being: 'in $\mathscr{P}$, assume E in the contexts indicated by Ω. Definition 14 defines the kind of formal object denoted by Ω. Notation 13 specifies three particular cases the first two of which will play a special role in relation to the stable model and the well-founded semantics. Recall Definition 3 for the notion of an occurrence of a literal in a formula.

Definition 14
Let a formal logic program $\mathscr{P} = (\varphi^{\epsilon}_{\wp})_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$ be given. A *literal marker for* $\mathscr{P}$ is a sequence of the form $(O^{\epsilon}_{\wp})_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$ where for all $\wp \in \mathrm{Prd}(\mathcal{V}^\star)$ and $\epsilon \in \{+,-\}$, $O^{\epsilon}_{\wp}$ is a set of occurrences of literals in $\varphi^{\epsilon}_{\wp}$.

Notation 13
Let a formal logic program $\mathscr{P}$ and a literal marker Ω for $\mathscr{P}$ be given. Write $\mathscr{P}$ as $(\varphi^{\epsilon}_{\wp})_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$ and Ω as $(O^{\epsilon}_{\wp})_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$.

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

In Section 2.3, 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 $\mathscr{P}$ be a formal logic program, say $(\varphi^{\epsilon}_{\wp})_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$, 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 $\Omega = (O^{\epsilon}_{\wp})_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$ be the literal marker for $\mathscr{P}$ such that O^{+}_{p} is equal to

$$\{\{\varphi^{+}_{p},\, p(v_1) \vee q(v_1),\, q(v_1)\},\, \{\varphi^{+}_{p},\, p(v_1) \vee r(v_1),\, p(v_1)\}\},$$

which corresponds to marking φ^{+}_{p} as

$$\big(p(v_1) \vee \underset{\checkmark}{q(v_1)}\big) \wedge \big(\underset{\checkmark}{p(v_1)} \vee\ r(v_1)\big).$$

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

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

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

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

The eventual definition of $\mathscr{P} +_\Omega E$ for arbitrary choices of $\mathscr{P}$, Ω and E, will be a straightforward generalisation of those examples. One should keep in mind that E will eventually be chosen as 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 implies 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 formula φ can be shown to subsume some member of a set E of formulas.

Notation 14
Given a formula φ, $n \in \mathbb{N}$, distinct variables $x_1, \ldots, x_n$ with $\mathrm{fv}(\varphi) = \{x_1,\ldots,x_n\}$, and a set E of formulas, we let $\mathrm{Unif}(\varphi, E)$ denote the set of all formulas of the form

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

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 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.

In the propositional case, $n = 0$ and Notation 14 simplifies the definition of $\mathrm{Unif}(\varphi, E)$ to $\{\bigwedge\varnothing\}$ if $\varphi \in E$, and to $\varnothing$ otherwise. The next notation will describe the operations of strengthening or weakening some occurrences of literals in a formula: given a formula φ, a set O of occurrences of literals in φ and a set E of literals,

- $\odot^O_E\varphi$ will be a formula obtained from φ by assuming that every occurrence of a literal in φ that belongs to O is false unless it subsumes some member of E;
- $\circledcirc^O_E\varphi$ will be a formula obtained from φ by assuming that every occurrence of a literal in φ that belongs to O is true if it subsumes some member of E.

Note that $\odot^O_E\varphi$ and $\circledcirc^O_E\varphi$ do not denote two notions that differ only in that one refers to 'false' when the other refers to 'true'. This may be more easily observed, thanks to the following alternative, but less precise, informal description of $\odot^O_E\varphi$ and $\circledcirc^O_E\varphi$: an occurrence of a literal in φ that belongs to O is true in $\odot^O_E\varphi$ iff it subsumes some member of E, whereas an occurrence of a literal in φ that belongs to O is true in $\circledcirc^O_E\varphi$ iff it subsumes a member of E or if it can be shown to be true.

The first operation prepares the technical definition of a formal logic program obtained from $\mathscr{P}$ and E, and denoted $\mathscr{P}\,|_\Omega\, E$, which will be useful for formalising in our setting the answer-set and the stable model semantics. The second operation prepares the definition of $\mathscr{P} +_\Omega E$. Hence to develop our framework, only the notation $\circledcirc^O_E\varphi$ is needed: the notation $\odot^O_E\varphi$ is used only to reformulate the usual definitions of the answer-set and the stable model semantics in a way that will make it easier to establish their relationship to our setting. To relate our framework to the stable model semantics, Ω will be set to $\langle -,- \rangle_{\mathscr{P}}$, and to relate it to the well-founded semantics, Ω will be set to either $\langle -,- \rangle_{\mathscr{P}}$ or $\langle \cdot,- \rangle_{\mathscr{P}}$ (both options are equally suitable), which prompts for a special notation, that of Notation 16. Both $\mathscr{P}\,|_\Omega\, E$ and $\mathscr{P} +_\Omega E$ are formally defined in Notation 17.

Notation 15

Let E be a set of literals. We inductively define for all formulas φ and sets O of occurrences of literals in φ two formulas $\odot^O_E\varphi$ and $\circledcirc^O_E\varphi$. 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 an identity, a distinction, or a literal.
  - If $O = \varnothing$ then both $\odot^O_E\varphi$ and $\circledcirc^O_E\varphi$ are φ.
  - If $O = \{\{\varphi\}\}$ then $\odot^O_E\varphi$ is $\bigvee \mathrm{Unif}(\varphi, E)$ and $\circledcirc^O_E\varphi$ is $\bigvee\{\varphi\} \cup \mathrm{Unif}(\varphi, E)$.
- Suppose that φ is of the form $\bigvee X$ or $\bigwedge X$. For all $\psi \in X$, let O_ψ be the (unique) set of occurrences of literals in ψ, say o, with $o \cup \{\varphi\} \in O$.
  - If φ is the formula $\bigvee X$ then $\odot^O_E\varphi$ is $\bigvee\{\odot^{O_\psi}_E\psi \mid \psi \in X\}$ and $\circledcirc^O_E\varphi$ is $\bigvee\{\circledcirc^{O_\psi}_E\psi \mid \psi \in X\}$.
  - If φ is the formula $\bigwedge X$ then $\odot^O_E\varphi$ is $\bigwedge\{\odot^{O_\psi}_E\psi \mid \psi \in X\}$ and $\circledcirc^O_E\varphi$ is $\bigwedge\{\circledcirc^{O_\psi}_E\psi \mid \psi \in X\}$.
- Suppose that φ is of the form $\exists x\,\psi$ or $\forall x\,\psi$. Let O_ψ be the (unique) set of occurrences of literals in ψ, say o, with $o \cup \{\varphi\} \in O$.
  - If φ is the formula $\exists x\,\psi$ then $\odot^O_E\varphi$ and $\circledcirc^O_E\varphi$ are the existential closure of $\odot^{O_\psi}_E\psi$ and $\exists x\,\circledcirc^{O_\psi}_E\psi$, respectively[3].
  - If φ is the formula $\forall x\,\psi$ then $\odot^O_E\varphi$ and $\circledcirc^O_E\varphi$ are the universal closure of $\odot^{O_\psi}_E\psi$ and $\forall x\,\circledcirc^{O_\psi}_E\psi$, respectively[4].

Notation 16

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 $\odot^O_E\varphi$ and $\circledcirc^-_E\varphi$ for $\circledcirc^O_E\varphi$.

Example 4

Suppose that $\mathscr{P}$ is the formal logic program of Example 2 and

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

[3] We cannot write $\exists x\,\odot^{O_\psi}_E\psi$ as x might not occur free in $\odot^{O_\psi}_E\psi$.

[4] Similarly, we cannot write $\forall x\,\odot^{O_\psi}_E\psi$ as x might not occur free in $\odot^{O_\psi}_E\psi$.

- As $\mathrm{Unif}(\neg p_3(v_0), E)$ is $\{v_0 \doteq \overline{1}, v_0 \doteq \overline{2}\}$, $\odot_E^- \varphi_{p_3}^+$ and $\circledcirc_E^- \varphi_{p_3}^+$ are
$$v_1 \doteq \overline{0} \vee \exists v_0 \big(v_1 \doteq s(v_0) \wedge (v_0 \doteq \overline{1} \vee v_0 \doteq \overline{2})\big)$$
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.
- As $\varphi_{p_4}^+$ does not contain any occurrence of a negated atom, $\odot_E^- \varphi_{p_4}^+$ and $\circledcirc_E^- \varphi_{p_4}^+$ are both identical to $\varphi_{p_4}^+$.
- As $\neg p_4(s(s(v_1)))$ does not unify with $\neg p_4(\overline{1})$ and $\neg q_3$ does not belong to E, $\odot_E^- \varphi_{p_4}^-$ and $\odot_E^- \varphi_{q_2}^-$ are both identical 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.
- As $\neg q_5$ belongs to E, $\mathrm{Unif}(\neg q_5, E)$ is $\{\bigwedge \varnothing\}$, and $\odot_E^- \varphi_{q_5}^+$ and $\circledcirc_E^- \varphi_{q_5}^+$ are both logically equivalent in $\mathcal{W}$ to $\bigwedge \varnothing$.

Notation 17
Let a formal logic program $\mathscr{P}$ and a literal marker Ω for $\mathscr{P}$ be given. Write $\mathscr{P}$ as $(\varphi_\wp^\epsilon)_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$ and Ω as $(O_\wp^\epsilon)_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$. Let a set E of literals be given.

- We let $\mathscr{P}\,|_\Omega\, E$ denote $(\odot_E^{O_\wp^\epsilon} \varphi_\wp^\epsilon)_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$.
- We let $\mathscr{P} +_\Omega E$ denote $(\circledcirc_E^{O_\wp^\epsilon} \varphi_\wp^\epsilon)_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$.

Property 7
For all formal logic programs $\mathscr{P}$, literal markers Ω for $\mathscr{P}$ and sets E of literals, $[\mathscr{P}] \subseteq [\mathscr{P} +_\Omega E]$.

The next property will be applied in the particular case where one of E and F denotes a set of literals over $\mathcal{V}^\star$, and the other the set of its closed instances.

Property 8
Let a formal logic program $\mathscr{P}$, a literal marker Ω for $\mathscr{P}$, and two sets E and F of literals be such that E and F have the same closed instances. Then $[\mathscr{P}\,|_\Omega\, E]$ is equal to $[\mathscr{P}\,|_\Omega\, F]$ and $[\mathscr{P} +_\Omega E]$ is equal to $[\mathscr{P} +_\Omega F]$.

The next lemma will play a crucial role in relating our framework to the answer-set and the stable model semantics.

Lemma 1
Let a formal logic program $\mathscr{P}$, a literal marker Ω for $\mathscr{P}$, and a set E of literals be given. Then $[\mathscr{P}\,|_\Omega\, E] \subseteq [\mathscr{P} +_\Omega E]$. Also, if all closed instances of members of $[\mathscr{P}\,|_\Omega\, E]$ are instances of members of E then $[\mathscr{P} +_\Omega E] = [\mathscr{P}\,|_\Omega\, E]$.

Proof
Write $\Omega = (O^{\epsilon}_{\wp})_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$.

To verify the first part of the lemma, it suffices by Property 2 to show that for all ordinals α, $[\mathscr{P}\,|_\Omega\, E]_\alpha \subseteq [\mathscr{P} +_\Omega E]_\alpha$. Proof is by induction. Let $\alpha \in \mathrm{Ord}$ be given, and assume that for all $\beta < \alpha$, $[\mathscr{P}\,|_\Omega\, E]_\beta \subseteq [\mathscr{P} +_\Omega E]_\beta$. Let $n \in \mathbb{N}$, $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$, terms $t_1, \ldots, t_n$ over $\mathcal{V}^\star$ and $\epsilon \in \{+,-\}$ be such that $\wp^\epsilon(t_1,\ldots,t_n) \in [\mathscr{P}\,|_\Omega\, E]_\alpha$. Then $\bigcup_{\beta<\alpha}[\mathscr{P}\,|_\Omega\, E]_\beta \Vdash \odot_E^{O^\epsilon_\wp}\mathscr{P}\big[\wp^\epsilon[t_1/v_1,\ldots,t_n/v_n]\big]$. Moreover, it is immediately verified that $\{\odot_E^{O^\epsilon_\wp}\mathscr{P}\big[\wp^\epsilon[t_1/v_1,\ldots,t_n/v_n]\big]\}$ logically implies $\circledcirc_E^{O^\epsilon_\wp}\mathscr{P}\big[\wp^\epsilon[t_1/v_1,\ldots,t_n/v_n]\big]$ in $\mathcal{W}$. This together with the inductive hypothesis implies that $\bigcup_{\beta<\alpha}[\mathscr{P} +_\Omega E]_\beta$ forces $\circledcirc_E^{O^\epsilon_\wp}\mathscr{P}\big[\wp^\epsilon[t_1/v_1,\ldots,t_n/v_n]\big]$; hence $\wp^\epsilon(t_1,\ldots,t_n) \in [\mathscr{P} +_\Omega E]_\alpha$, completing the verification that $[\mathscr{P}\,|_\Omega\, E] \subseteq [\mathscr{P} +_\Omega E]$.

Assume that all closed instances of members of $[\mathscr{P}\,|_\Omega\, E]$ are instances of members of E. Suppose for a contradiction that $[\mathscr{P} +_\Omega E] \not\subseteq [\mathscr{P}\,|_\Omega\, E]$. By Property 2, let ordinal α be least with $[\mathscr{P} +_\Omega E]_\alpha \not\subseteq [\mathscr{P}\,|_\Omega\, E]$. Let $n \in \mathbb{N}$, $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$, terms $t_1, \ldots, t_n$ over $\mathcal{V}^\star$ and $\epsilon \in \{+,-\}$ be such that $\wp^\epsilon(t_1,\ldots,t_n)$ belongs to $[\mathscr{P} +_\Omega E]_\alpha \setminus [\mathscr{P}\,|_\Omega\, E]$. Then $\bigcup_{\beta<\alpha}[\mathscr{P} +_\Omega E]_\beta \Vdash \circledcirc_E^{O^\epsilon_\wp}\mathscr{P}\big[\wp^\epsilon[t_1/v_1,\ldots,t_n/v_n]\big]$; so by the choice of α, $[\mathscr{P}\,|_\Omega\, E] \Vdash \circledcirc_E^{O^\epsilon_\wp}\mathscr{P}\big[\wp^\epsilon[t_1/v_1,\ldots,t_n/v_n]\big]$, which together with the assumption on E, easily implies that $[\mathscr{P}\,|_\Omega\, E]$ forces $\odot_E^{O^\epsilon_\wp}\mathscr{P}\big[\wp^\epsilon[t_1/v_1,\ldots,t_n/v_n]\big]$. Hence $\wp^\epsilon(t_1,\ldots,t_n)$ belongs to $[\mathscr{P}\,|_\Omega\, E]$; contradiction. □

The transformation of a formal logic program $\mathscr{P}$ into a formal logic program of the form $\mathscr{P} +_\Omega E$ 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 15
Let a formal logic program $\mathscr{P}$ and a literal marker Ω for $\mathscr{P}$ be given. We call an *extensor for* $(\mathscr{P}, \Omega)$ any set E of literals over $\mathcal{V}^\star$ such that $E \cup [\mathscr{P} +_\Omega E]$ is consistent.

Before we end this section, we need one more technical notation. In relation to the well-founded semantics, it will be convenient to introduce an intermediate construction involving a family of extensors: a formal logic program $\mathscr{P}$ will be extended to a formal logic program of the form $\mathscr{P} +_\Omega E_0$, and then to a formal logic program of the form $(\mathscr{P} +_\Omega E_0) +_{\Omega_1} E_1$, and then to a formal logic program of the form $(\mathscr{P} +_\Omega E_0 \cup E_1) +_{\Omega_2} E_2$, etc. Now Ω_1, Ω_2, etc., will not be arbitrary: they will all select occurrences of literals in $\mathscr{P} +_\Omega E_0$, $\mathscr{P} +_\Omega E_0 \cup E_1$, etc., determined by Ω, even though these occurrences of literals are taken from different formal logic programs as $\mathscr{P}$ is being successively transformed. For instance, in Example 4, the occurrence of $\neg p_3(v_0)$ in $\varphi^+_{p_3}$ can be 'tracked down' in $\circledcirc^-_E\varphi^+_{p_3}$, though the (unique) occurrence of $\neg p_3(v_0)$ in $\varphi^+_{p_3}$ is of course different to the (unique) occurrence of $\neg p_3(v_0)$ in $\circledcirc^-_E\varphi^+_{p_3}$. The following notation will allow us to formally express Ω_1, Ω_2, etc., from Ω and E_0, E_1, etc., and write $\Omega + E_0$ for Ω_1, $\Omega + E_0 \cup E_1$ for Ω_2, etc.

Notation 18
For all formulas 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 formula in which

$o \setminus \{\varphi\}$ is an occurrence of a literal, and such that $o' \cup \{\varphi\} \in O$. Given a formula φ, a set O of occurrences of literals in φ, a set E of literals and a member o of O, set

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

Notation 19
Let a formal logic program $\mathscr{P}$, a literal marker Ω for $\mathscr{P}$ and a set E of literals be given. Write $\Omega = (O_\wp^\epsilon)_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$. We let $\Omega + E$ denote

$$\left(\{\circledcirc_E^{O_\wp^\epsilon} o \mid o \in O_\wp^\epsilon\}\right)_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}.$$

The property that follows justifies why the construction described before Notation 18 refers to a formal logic program of the form $(\mathscr{P} +_\Omega E_0 \cup E_1) +_{\Omega_2} E_2$ rather than to a formal logic program of the form $\big((\mathscr{P} +_\Omega E_0) +_{\Omega_1} E_1\big) +_{\Omega_2} E_2$.

Property 9
Let a formal logic program $\mathscr{P}$, a literal marker Ω for $\mathscr{P}$, and two sets E and F of literals be given. Then $[\mathscr{P} +_\Omega E \cup F] = [(\mathscr{P} +_\Omega E) +_{\Omega+E} F]$.

5.2 Special extensors

The task of casting the well-founded, the stable model, and the answer-set semantics 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 16
Let a formal logic program $\mathscr{P}$, a literal marker Ω for $\mathscr{P}$, and an extensor E for $(\mathscr{P}, \Omega)$ be given.

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

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

- We say that E is *implicative* iff $E \subseteq [\mathscr{P} +_\Omega E]$.
- We say that E is *supporting* iff for all $\psi \in E$, $[\mathscr{P}] \Vdash (\mathscr{P} +_\Omega E)[\psi]$.
- 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 ordinals $\beta < \alpha$, E_β is a supporting extensor for $(\mathscr{P} +_\Omega \bigcup_{\gamma<\beta} E_\gamma,\, \Omega + \bigcup_{\gamma<\beta} E_\gamma)$.
- 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 $(\mathscr{P}, \Omega)$.

Intuitively, an imperative extensor is a maximal set of hypotheses that will not be refuted, an implicative extensor is a set of hypotheses that will be confirmed, and a supporting extensor for $\mathscr{P}$ is a set of hypotheses that will be confirmed, thanks to themselves and to the literals generated by $\mathscr{P}$ (but not to any nonhypothesis generated by a rule that fires only thanks to some hypothesis that activates its body). When the literal marker Ω marks all occurrences of literals that can unify with a hypothesis (so any hypothesis can be used in any context), supporting extensors have an alternative definition. This is what the next property expresses, with a corollary that will be used in relation to the well-founded semantics.

Property 10
Let a formal logic program $\mathscr{P} = (\varphi^{\epsilon}_{\wp})_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$, a literal marker $\Omega = (O^{\epsilon}_{\wp})_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$ for $\mathscr{P}$, and an extensor E for $(\mathscr{P}, \Omega)$ be such that for all $n \in \mathbb{N}$, $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$, $\epsilon \in \{+, -\}$, literals ψ and occurrences o of ψ in $\varphi^{\epsilon}_{\wp}$, if some closed instance of ψ is an instance of a member of E then $o \in O^{\epsilon}_{\wp}$. Then E is supporting iff for all $\psi \in E$, $[\mathscr{P}] \cup E \Vdash \mathscr{P}[\psi]$.

Corollary 2
Let a formal logic program $\mathscr{P}$ be given. Let a supporting extensor E for $(\mathscr{P}, \langle\cdot,-\rangle_{\mathscr{P}})$ consist of negated atoms only. Then E is supporting iff for all $\psi \in E$, $[\mathscr{P}] \cup E \Vdash \mathscr{P}[\psi]$.

Recall that we have defined a set X of literals to be saturated iff every closed atom is an instance of a member of at least one of the sets X and $\sim X$.

Property 11
For all formal logic programs $\mathscr{P}$ and literal markers Ω for $\mathscr{P}$, all imperative extensors for $(\mathscr{P}, \Omega)$ are saturated.

Property 12
For all formal logic programs $\mathscr{P}$ and literal markers Ω for $\mathscr{P}$, all supporting extensors for $(\mathscr{P}, \Omega)$ are implicative.

Property 13
Let a formal logic program $\mathscr{P}$, a literal marker Ω for $\mathscr{P}$, and an extensor E for $(\mathscr{P}, \Omega)$ be given.

- For all ordinals α, if E is α-foundational then E is foundational.
- For all ordinals α, if E is α-foundational then E is β-foundational for all ordinals $\beta > \alpha$.
- If E is foundational then there is $\alpha \in \mathrm{Ord}$ such that E is α-foundational.

It will be shown that the well-founded semantics is related to foundational extensors, and the answer-set semantics 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 14
For all formal logic programs $\mathscr{P}$, literal markers Ω for $\mathscr{P}$ and complete sets E of literals over $\mathcal{V}^\star$, E is an implicative extensor for $(\mathscr{P}, \Omega)$ iff E is an imperative extensor for $(\mathscr{P}, \Omega)$.

It is fair to say that to cast the answer-set, the stable model 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 the answer-set 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 2
Suppose that $\mathcal{V} \setminus \mathcal{V}^\star$ contains a function symbol of arity 1 at least. Let a formal logic program $\mathscr{P}$, a literal marker Ω for $\mathscr{P}$, and an imperative extensor E for $(\mathscr{P}, \Omega)$ be given. Then for all $n \in \mathbb{N}$ and $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$, the set of members of $[\mathscr{P} +_\Omega E]$ of the form $\wp^\epsilon(t_1, \dots, t_n)$ is either empty or equal to the set of all atoms over $\mathcal{V}^\star$ of the form $\wp(t_1, \dots, t_n)$ or equal to the set of all negated atoms over $\mathcal{V}^\star$ of the form $\neg\wp(t_1, \dots, t_n)$.

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.

- Let X be the set of n-tuples of terms over $\mathcal{V}^\star$, say $(t_1, \dots, t_n)$, such that for all closed terms $t'_1, \dots, t'_n$, if $(t'_1, \dots, t'_n)$ is an instance of $(t_1, \dots, t_n)$ then both $\wp(t'_1, \dots, t'_n)$ and $\neg\wp(t'_1, \dots, t'_n)$ are instances of members of E.
- For all $\epsilon \in \{+, -\}$, let X^ϵ be the set of n-tuples of terms over $\mathcal{V}^\star$, say $(t_1, \dots, t_n)$, such that $\wp^\epsilon(t_1, \dots, t_n) \in [\mathscr{P} +_\Omega E]$.

Using the fact that E is an imperative extensor for $(\mathscr{P}, \Omega)$, it is easy to verify that X, X^+, and X^- are disjoint and that for all closed terms $t_1, \dots, t_n$, $(t_1, \dots, t_n)$ is an instance of some member of $X \cup X^+ \cup X^-$. Let a nonnullary function symbol f in $\mathcal{V} \setminus \mathcal{V}^\star$ be given. Then there exists an n-tuple $(\iota_1, \dots, \iota_n)$ of distinct closed terms that all start with f. Obviously, for all terms $t_1, \dots, t_n$ over $\mathcal{V}^\star$, if $(\iota_1, \dots, \iota_n)$ is an instance of $(t_1, \dots, t_n)$ then $t_1, \dots, t_n$ are distinct variables. So either all n-tuples of closed terms are instances of some member of X, in which case $[\mathscr{P} +_\Omega E]$ contains no literal over $\mathcal{V}^\star$ of the form $\wp(t_1, \dots, t_n)$ or $\neg\wp(t_1, \dots, t_n)$, or all n-tuples of closed terms are instances of some member of X^+, in which case $[\mathscr{P} +_\Omega E]$ contains all literals over $\mathcal{V}^\star$ of the form $\wp(t_1, \dots, t_n)$, or all n-tuples of closed terms are instances of some member of X^-, in which case $[\mathscr{P} +_\Omega E]$ contains all literals over $\mathcal{V}^\star$ of the form $\neg\wp(t_1, \dots, t_n)$, completing the proof of the proposition. $\square$

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 can be nondegenerate.

Example 5
Suppose that $\mathcal{V}$ consists of $\overline{0}$, s and a binary predicate symbol p, and assume that $\mathcal{V}^\star = \{s, p\}$. Let $\mathscr{P}$ be the formal logic program determined by $\varphi_p^+ \equiv v_1 \doteq v_2$ and $\varphi_p^- \equiv \bigvee \varnothing$. Let E be the set of literals defined as

$$\{p(v_1, v_2)\} \cup \{\neg p(s^n(v_0), v_0),\ \neg p(v_0, s^n(v_0)) \mid n \in \mathbb{N} \setminus \{0\}\}.$$

Set $\Omega = (\varnothing, \varnothing)$. Then E is an imperative extensor for $(\mathscr{P}, \Omega)$ and $[\mathscr{P} +_\Omega E]$, which is obviously equal to $[\mathscr{P}]$, is $\{p(s^n(v_i), s^n(v_i)) \mid n \in \mathbb{N}, i \in \mathbb{N}\}$.

To summarise the previous considerations, we have not assumed in Definition 16 that $\mathcal{V}^\star$ and $\mathcal{V}$ are equal simply because none of the results we want to establish needs that assumption to be made. But the notion of imperative extensor (which is the key notion in relation to the stable model and the answer-set semantics) is defined in such a way that it is only interesting when $\mathcal{V}^\star = \mathcal{V}$ or when $\mathcal{V}^\star$ and $\mathcal{V}$ take very specific values.

5.3 A few technical results

The results that follow will be used in the sequel.

Lemma 2
Let a formal logic program $\mathscr{P}$ and a literal marker Ω for $\mathscr{P}$ be given. For all sets E and F of literals, if $E \subseteq F$ then $[\mathscr{P} +_\Omega E] \subseteq [\mathscr{P} +_\Omega F]$.

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, $[\mathscr{P} +_\Omega E]_\alpha \subseteq [\mathscr{P} +_\Omega F]_\alpha$. We conclude with Property 2. $\square$

Lemma 3
Let a formal logic program $\mathscr{P}$ and a literal marker Ω for $\mathscr{P}$ be given. For all sets E and F of literals, $\big[\mathscr{P} +_\Omega [\mathscr{P} +_\Omega E] \cup F\big] \subseteq [\mathscr{P} +_\Omega E \cup F]$.

Proof
Let E and F be two sets of literals. Let ordinal λ be such that $[\mathscr{P} +_\Omega E]_\lambda$ is equal to $[\mathscr{P} +_\Omega E]_{\lambda+1}$. It is easy to verify by induction that for all ordinals α, $[\mathscr{P} +_\Omega [\mathscr{P} +_\Omega E] \cup F]_\alpha \subseteq [\mathscr{P} +_\Omega E \cup F]_{\lambda+\alpha}$. We conclude with Property 2. $\square$

Lemma 4
For all formal logic programs $\mathscr{P}$, literal markers Ω for $\mathscr{P}$ and implicative extensors E for $(\mathscr{P}, \Omega)$, $\big[\mathscr{P} +_\Omega [\mathscr{P} +_\Omega E]\big] = [\mathscr{P} +_\Omega E]$.

Proof
The lemma follows immediately from Lemmas 2 and 3. $\square$

Corollary 3
For all formal logic programs $\mathscr{P}$, literal markers Ω for $\mathscr{P}$ and implicative extensors E for $(\mathscr{P}, \Omega)$, $[\mathscr{P} +_\Omega E]$ is an implicative extensor for $(\mathscr{P}, \Omega)$.

Proposition 3
Let a formal logic program $\mathscr{P}$ be locally consistent. Let a literal marker Ω for $\mathscr{P}$ be given. Let a set X of implicative extensors for $(\mathscr{P}, \Omega)$ be such that $\bigcup X$ is consistent. Then $\bigcup X$ is an extensor for $(\mathscr{P}, \Omega)$.

Proof
Write $\mathscr{P} = (\varphi_\wp^\epsilon)_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$. Set $E = \bigcup X$. We show by induction that for all ordinals α, $E \cup [\mathscr{P} +_\Omega E]_\alpha$ is consistent. Let an ordinal α be given and assume that for all $\beta < \alpha$, $E \cup [\mathscr{P} +_\Omega E]_\beta$ is consistent. Since $\mathscr{P}$ 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 $E \cup \bigcup_{\beta<\alpha}[\mathscr{P} +_\Omega E]_\beta$ 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 $E \cup [\mathscr{P} +_\Omega E]_\alpha$ cannot be inconsistent unless the set of closed instances of members of $[\mathscr{P} +_\Omega E]_\alpha$ intersects the set of closed instances of members of $\sim E$. Assume that the set of closed instances of members of $[\mathscr{P} +_\Omega E]_\alpha$ indeed intersects the set of closed instances of members of $\sim E$. Since all members of X are implicative, any closed instance of any member of E is an instance of some member of $\bigcup_{F\in X}[\mathscr{P} +_\Omega F]$. Let ordinal λ be least such that there exists a closed literal φ with $\bigcup_{F\in X}[\mathscr{P} +_\Omega F]_\lambda \Vdash \varphi$ and $[\mathscr{P} +_\Omega E]_\alpha \Vdash \sim\varphi$. Let $F \in X$ and a closed literal φ be such that $[\mathscr{P} +_\Omega F]_\lambda \Vdash \varphi$ and $[\mathscr{P} +_\Omega E]_\alpha \Vdash \sim\varphi$. Set

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

We derive from 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 $\mathscr{P}$ is locally consistent. We conclude that $E \cup [\mathscr{P} +_\Omega E]_\alpha$ is consistent. $\square$

As an immediate consequence of Property 12 and Proposition 3:

Corollary 4
Let a formal logic program $\mathscr{P}$ be locally consistent. Let a literal marker Ω for $\mathscr{P}$ be given. Let X be a set of supporting extensors for $(\mathscr{P}, \Omega)$ such that $\bigcup X$ is consistent. Then $\bigcup X$ is a supporting extensor for $(\mathscr{P}, \Omega)$.

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 $[\mathscr{P} +_\Omega E]$ is a model of the classical logical form of $\mathscr{P}$. It is easy to answer that question positively for implicative extensors.

Proposition 4
Let a formal logic program $\mathscr{P}$, a literal marker Ω for $\mathscr{P}$, and an implicative extensor E for $(\mathscr{P}, \Omega)$ be such that $[\mathscr{P} +_\Omega E]$ is complete. Then the set of closed instances of atoms in $[\mathscr{P} +_\Omega E]$ is a model of $\mathrm{Clf}(\mathscr{P})$.

Proof
Obviously, for all formulas φ and sets O of occurrences of literals in φ, φ logically implies $\circledcirc^O_{[\mathscr{P}+_\Omega[\mathscr{P}+_\Omega E]]}\varphi$ in $\mathcal{W}$. It follows that $\mathrm{Clf}(\mathscr{P} +_\Omega [\mathscr{P} +_\Omega E])$ logically implies $\mathrm{Clf}(\mathscr{P})$ in $\mathcal{W}$. By Lemma 4, $[\mathscr{P} +_\Omega E] = \big[\mathscr{P} +_\Omega [\mathscr{P} +_\Omega E]\big]$, and we derive from Corollary 1 that $[\mathscr{P} +_\Omega E]$ logically implies $\mathrm{Clf}(\mathscr{P} +_\Omega [\mathscr{P} +_\Omega E])$ in $\mathcal{W}$. We conclude that $[\mathscr{P} +_\Omega E] \vDash_{\mathcal{W}} \mathrm{Clf}(\mathscr{P})$. $\square$

5.4 Relationship to the answer-set semantics

In this section, we consider the enrichment of $\mathfrak{L}_{\omega_1\omega}(\mathcal{V})$ with a second negation operator, written *not*, which 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 introduce some preliminary notation. Let a formula 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 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* ψ.
- Suppose that φ is of the form $\bigvee X$ or $\bigwedge X$. For all $\psi \in X$, let O_ψ be the (unique) set of occurrences of literals in ψ, say o, with $o \cup \{\varphi\} \in O$.
  - If φ is the formula $\bigvee X$ then $\varphi[O]$ is $\bigvee\{\psi[O_\psi] \mid \psi \in X\}$.
  - If φ is the formula $\bigwedge X$ 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 of occurrences of literals in ψ, say o, with $o \cup \{\varphi\} \in O$.
  - If φ is the formula $\exists x\,\psi$ then $\varphi[O]$ is $\exists x\,\psi[O_\psi]$.
  - If φ is the formula $\forall x\,\psi$ then $\varphi[O]$ is $\forall x\,\psi[O_\psi]$.

Now let a formal logic program $\mathscr{P}$ and a literal marker Ω for $\mathscr{P}$ be given. Write $\mathscr{P} = (\varphi^{\epsilon}_{\wp})_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$ and $\Omega = (O^{\epsilon}_{\wp})_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$, and set $\mathscr{P}[\Omega] = \left(\varphi^{\epsilon}_{\wp}[O^{\epsilon}_{\wp}]\right)_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$. Then $\mathscr{P}[\Omega]$ is what is known in the literature as an extended logic program, a logic program with two kinds of negation, $\neg$ and *not*. Conversely, let an extended 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 $\wp(v_1,\dots,v_n)$, one rule whose head is $\neg\wp(v_1,\dots,v_n)$, and no other rule whose head is of the form $\wp(t_1,\dots,t_n)$ or $\neg\wp(t_1,\dots,t_n)$. Then there exists a unique formal logic program $\mathscr{P}$ and a unique literal marker Ω for $\mathscr{P}$ with $G = \mathscr{P}[\Omega]$.

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

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

Suppose that Ω is given by the following sets.

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

So $(\mathscr{P}, \Omega)$ can be represented as

$$\begin{array}{llll}
p_1 \leftarrow p_2 \wedge p_3 & p_2 \leftarrow p_4 & p_3 \leftarrow p_3 & p_4 \leftarrow \underset{\checkmark}{\neg p_3} \\
\neg p_1 \leftarrow \underset{\checkmark}{p_2} \vee \underset{\checkmark}{\neg p_4} & \neg p_2 \leftarrow \neg p_3 & \neg p_3 \leftarrow \neg p_3 \wedge \underset{\checkmark}{p_2} &
\end{array}$$

and $\mathscr{P}[\Omega]$ is the extended logic program

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

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

- all subsets E of $\{p_1, \neg p_1, \neg p_2, p_3, p_4\}$, in which case $[\mathscr{P} +_\Omega E] = \emptyset$;
- 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 $[\mathscr{P} +_\Omega E] = \{\neg p_1\}$;
- 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 $[\mathscr{P} +_\Omega E] = \{\neg p_1, p_2, p_4\}$.

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 $\mathscr{P}[\Omega]$, namely $\{\neg p_1, p_2, p_4\}$.

Having realised that the class of extended logic programs is in one-to-one correspondence with the class of pairs $(\mathscr{P}, \Omega)$ where $\mathscr{P}$ is a formal logic program and Ω a literal marker for $\mathscr{P}$ (the correspondence in question putting a pair of the form $(\mathscr{P}, \Omega)$ in relation to $\mathscr{P}[\Omega]$), it is easy to see that if one assumes that $\mathcal{V}^\star$ is equal to $\mathcal{V}$, then Definition 17 amounts to the notion of an answer-set—recall the discussion at the end of Section 5.2 about not assuming that $\mathcal{V}^\star$ and $\mathcal{V}$ are equal.

Definition 17

Let a formal logic program $\mathscr{P}$ and a literal marker Ω for $\mathscr{P}$ be given. An *answer-set for* $(\mathscr{P}, \Omega)$ is a consistent set of closed literals M for which there exists a (necessarily 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 $[\mathscr{P} |_\Omega E]$.

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

Proposition 5

Let a formal logic program $\mathscr{P}$, a literal marker Ω for $\mathscr{P}$, 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 $(\mathscr{P}, \Omega)$ iff M is an answer-set for $(\mathscr{P}, \Omega)$.

Proof

Assume that E is an imperative extensor for $(\mathscr{P}, \Omega)$. By Definition 16, the set of closed instances of members of $[\mathscr{P} +_\Omega E]$ is consistent, is precisely equal to M, and is included in F, which implies by Lemma 1 that $[\mathscr{P} +_\Omega E] = [\mathscr{P} |_\Omega E]$. We conclude that M is an answer-set for $(\mathscr{P}, \Omega)$.

Conversely, assume that M is an answer-set for $(\mathscr{P}, \Omega)$. Hence M is consistent, and so $M \subseteq F$. By Definition 17 and Property 8, M consists of the closed instances of the members of $[\mathscr{P} |_\Omega E]$, and so by Lemma 1, consists of the closed instances of the members of $[\mathscr{P} +_\Omega E]$. Hence E is an imperative extensor for $(\mathscr{P}, \Omega)$. □

In the answer-set semantics, *not* φ intuitively means that φ is not provable, that is, not derived. The way to go from the usual presentation of the answer-set semantics to our setting is to let hypotheses of the form $\sim\varphi$ take effect in contexts where the answer-set framework has statements of the form 'φ is not provable'. The fact that φ is either provable or not is then mapped to the constraint, captured by the notion of imperative extensor, that either φ should be derived or $\sim\varphi$ should be assumed.

5.5 Relationship to the stable model semantics

The stable model semantics takes the sets of positive rules as the object of study; but as mentioned repeatedly, the class of these sets 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 18 captures the notion of stable model—again, recall the discussion at the end of Section 5.2 about not assuming that $\mathcal{V}^\star$ and $\mathcal{V}$ are equal. Note how Notation 15 is being used in Definition 18 to basically describe the Lloyd–Topor transformation.

Definition 18
Let a formal logic program $\mathscr{P} = (\varphi^{\epsilon}_{\wp})_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$ be given. A set of closed literals M is said to be *stable for* $\mathscr{P}$ 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

$$\{\odot^{-}_{E}\varphi^{+}_{\wp} \rightarrow \wp(v_1,\dots,v_n) \mid n \in \mathbb{N},\ \wp \in \mathrm{Prd}(\mathcal{V}^\star, n)\} \vDash_{\mathcal{W}} \varphi.$$

Note that the condition on E only depends on the positive rules of $\mathscr{P}$. In Definition 18, $\mathscr{P}$ is not assumed to be symmetric; but it is essential to assume that $\mathscr{P}$ is symmetric to obtain the result stated in the proposition that follows. Together with Property 14, 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 characterise the notion of stable model.

Proposition 6
For all symmetric formal logic programs $\mathscr{P}$ and complete sets E of literals over $\mathcal{V}^\star$, the set of closed instances of members of E is stable for $\mathscr{P}$ iff E is an implicative extensor for $(\mathscr{P}, \langle -,- \rangle_{\mathscr{P}})$.

Proof
Let a symmetric formal logic program $\mathscr{P} = (\varphi^{\epsilon}_{\wp})_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$ 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 $\mathscr{P}$. Then $\mathrm{Clf}(\mathscr{P} \mid_{\langle -,- \rangle_{\mathscr{P}}} E)$ logically implies E^+ in $\mathcal{W}$. Since negation does not occur in the left-hand side of any implication in $\mathrm{Clf}(\mathscr{P} \mid_{\langle -,- \rangle_{\mathscr{P}}} E)$, it follows that E^+ is a subset of $[\mathscr{P} \mid_{\langle -,- \rangle_{\mathscr{P}}} E]$. Let $n \in \mathbb{N}$, $\wp \in \mathrm{Prd}(\mathcal{V}^\star, n)$, and closed terms $t_1, \dots, t_n$ be given. Since E is complete and $\mathscr{P}$ is symmetric, E forces one and only one of $\varphi^{+}_{\wp}[t_1/v_1,\dots,t_n/v_n]$ and $\varphi^{-}_{\wp}[t_1/v_1,\dots,t_n/v_n]$. Suppose that $\neg\wp(t_1,\dots,t_n)$ is an instance of a member of E. If $E \Vdash \varphi^+_\wp[t_1/v_1,\dots,t_n/v_n]$ then $E^+ \Vdash \odot^-_E \varphi^+_\wp[t_1/v_1,\dots,t_n/v_n]$, hence there exists terms $t'_1, \dots, t'_n$ over $\mathcal{V}^\star$ such that $(t_1,\dots,t_n)$ is an instance of $(t'_1,\dots,t'_n)$ and $\wp(t'_1,\dots,t'_n) \in E$, contradicting the assumption that E is consistent. We infer that $E^+ \Vdash \odot^-_E \varphi^-_\wp[t_1/v_1,\dots,t_n/v_n]$, hence there exists terms $t'_1, \dots, t'_n$ over $\mathcal{V}^\star$ such that $(t_1,\dots,t_n)$ is an instance of $(t'_1,\dots,t'_n)$ and $\neg\wp(t'_1,\dots,t'_n)$ belongs to $[\mathscr{P} \mid_{\langle -,- \rangle_\mathscr{P}} E]$. Suppose that $\wp(t_1,\dots,t_n)$ is an instance of a member of E. Then $E \Vdash \varphi^+_\wp[t_1/v_1,\dots,t_n/v_n]$, hence $E \nVdash \varphi^-_\wp[t_1/v_1,\dots,t_n/v_n]$, hence E does not force $\odot^-_E \varphi^-_\wp[t'_1/v_1,\dots,t'_n/v_n]$ for any terms $t'_1, \dots, t'_n$ over $\mathcal{V}^\star$ such that $(t_1,\dots,t_n)$ is an instance of $(t'_1,\dots,t'_n)$. It is then easy to conclude that for all closed literals φ, $[\mathscr{P} \mid_{\langle -,- \rangle_\mathscr{P}} E] \Vdash \varphi$ iff φ is an instance of a member of E. Together with Lemma 1, this completes the verification that E is an implicative extensor for $(\mathscr{P}, \langle -,- \rangle_\mathscr{P})$.

Conversely, assume that E is an implicative extensor for $(\mathscr{P}, \langle -,- \rangle_\mathscr{P})$. Since E is complete, Lemma 1 again implies that $[\mathscr{P} +_{\langle -,- \rangle_\mathscr{P}} E] = [\mathscr{P} \mid_{\langle -,- \rangle_\mathscr{P}} E]$. Set

$$X = \{\odot^-_E \varphi^+_\wp \rightarrow \wp(v_1,\dots,v_n) \mid n \in \mathbb{N},\ \wp \in \mathrm{Prd}(\mathcal{V}^\star, n)\}.$$

Clearly, $\mathrm{Clf}(\mathscr{P} \mid_{\langle -,- \rangle_\mathscr{P}} E)$, 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 $\mathscr{P}$. $\square$

5.6 *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 12, supporting extensors are implicative extensors, which means that they consist of hypotheses that are guaranteed to be confirmed. But more is true. Intuitively, given a formal logic program $\mathscr{P}$ and a literal marker Ω for $\mathscr{P}$, a supporting extensor for $(\mathscr{P}, \Omega)$ is sufficiently rich in literals to 'generate itself' using $\mathscr{P}$ and Ω, and not contradict any literal generated by $\mathscr{P} +_\Omega E$. So for all members φ of a supporting extensor E for $(\mathscr{P}, \Omega)$, there exists a 'constructive proof' of φ, from the rules formalised as $\mathscr{P}$, such that the only literals that occur in the proof either are in $[\mathscr{P}]$ 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 $[\mathscr{P}]$ is actually needed in the 'constructive proofs'.

Example 6
If $\mathscr{P}$ is the formal logic program of Example 2 then the supporting extensors for $(\mathscr{P}, \langle \pm,\pm \rangle_\mathscr{P})$ which are disjoint from $[\mathscr{P}]$ are the consistent unions of

- $\{p_4(\overline{2n}) \mid n \geqslant m\}$ where m is an arbitrary member of $\mathbb{N}$,
- $\{\neg p_4(\overline{2n}) \mid n \geqslant m\}$ where m is an arbitrary member of $\mathbb{N}$,
- $\{p_4(\overline{2n+1}) \mid n \geqslant m\}$ where m is an arbitrary member of $\mathbb{N}$,
- $\{\neg p_4(\overline{2n+1}) \mid n \geqslant m\}$ where m is an arbitrary member of $\mathbb{N}$,

- $\{q_2, q_3\}$,
- $\{\neg q_2, \neg q_3\}$, and
- $\{q_4\}$.

Casting the well-founded semantics into our framework requires to focus on symmetric formal logic programs only. But the notion of $\subseteq$-maximal foundational extensor, which will be seen to formalise the key principle behind the well-founded semantics, can be applied to arbitrary formal logic programs, hence to $\mathscr{P}$ of Example 2. For this particular formal logic program, the notion of $\subseteq$-maximal foundational extensor reduces to that of $\subseteq$-maximal supporting extensor; this is because in this particular case, the process of transfinitely transforming $\mathscr{P}$ with a $\subseteq$-maximal supporting extensor converges after its first application. Also, because of its full bias toward negated atoms, the well-founded semantics elects the literal marker that marks all occurrences of all negated atoms in the bodies of all rules or, alternatively, all negated atoms in the bodies of all negative rules (both choices are equivalent). It will be seen that the well-founded semantics of $\mathscr{P}$ is captured by the $\subseteq$-maximal set of negated atoms over $\mathcal{V}^\star$ that is a foundational extensor for $(\mathscr{P}, \langle -,- \rangle_{\mathscr{P}})$ or $(\mathscr{P}, \langle \cdot,- \rangle_{\mathscr{P}})$; with respect to the previous example, that set is $\{\neg p_4(\overline{n}) \mid n \in \mathbb{N}\} \cup \{\neg q_2, \neg q_3\}$.

The 'dual' of that semantics would be fully biased toward nonnegated atoms, and would be captured by the $\subseteq$-maximal set of atoms over $\mathcal{V}^\star$ that is a foundational extensor for $(\mathscr{P}, \langle +,+ \rangle_{\mathscr{P}})$ or $(\mathscr{P}, \langle +,\cdot \rangle_{\mathscr{P}})$ (where $\langle +,+ \rangle_{\mathscr{P}}$ and $\langle +,\cdot \rangle_{\mathscr{P}}$ would denote the literal marker for $\mathscr{P}$ that marks all occurrences of all nonnegated atoms in the bodies of all rules or all positive rules, respectively); in the context of the previous example, that set is $\{p_4(\overline{n}) \mid n \in \mathbb{N}\} \cup \{q_2, q_3, q_4\}$.

A 'balanced' semantics in the family of the semantics determined by maximal foundational extensors could elect a $\subseteq$-maximal foundational extensor for $(\mathscr{P}, \langle \pm,\pm \rangle_{\mathscr{P}})$ that contains $\{p_4(\overline{2n}), \neg p_4(\overline{2n+1}) \mid n \in \mathbb{N}\}$, and allows one to transform $\mathscr{P}$ 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.

Subsuming the notion of foundational extensor given in Definition 16 is the notion of foundational chain, which we make explicit in order to easily investigate the properties of the foundational extensors. Given a formal logic program $\mathscr{P}$ and a literal marker Ω for $(\mathscr{P}, \Omega)$, a foundational chain for $(\mathscr{P}, \Omega)$ can be described as follows.

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

Formally, this translates into the following definition.

Definition 19

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

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

A *foundational chain for* $(\mathscr{P}, \Omega)$ 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 $(\mathscr{P}, \Omega)$.

The proposition that follows generalises Property 12.

Proposition 7

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

Proof

Proof is by induction. Let a formal logic program $\mathscr{P}$, a literal marker Ω for $\mathscr{P}$, and a foundational chain $(E_\alpha)_{\alpha\in\mathrm{Ord}}$ for $(\mathscr{P}, \Omega)$ be given. Let an ordinal α be given and suppose that for all $\beta < \alpha$, $\bigcup_{\gamma<\beta} E_\gamma \subseteq [\mathscr{P} +_\Omega \bigcup_{\gamma<\beta} E_\gamma]$. There is nothing to verify if $\alpha = 0$. If α is a limit ordinal then it follows immediately from Lemma 2 that $\bigcup_{\beta<\alpha} E_\beta$ is included in $[\mathscr{P} +_\Omega \bigcup_{\beta<\alpha} E_\beta]$. Suppose that α is of the form $\delta + 1$. By inductive hypothesis, $\bigcup_{\gamma<\delta} E_\gamma$ is included in $[\mathscr{P} +_\Omega \bigcup_{\gamma<\delta} E_\gamma]$. Moreover, it follows from Properties 9 and 12 that $\left[\mathscr{P} +_\Omega [\mathscr{P} +_\Omega \bigcup_{\gamma<\delta} E_\gamma] \cup E_\delta\right]$ contains E_δ. Lemma 3 then implies that $E_\delta \subseteq [\mathscr{P} +_\Omega \bigcup_{\gamma\leqslant\delta} E_\gamma]$. We conclude with Property 13. $\square$

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

Proposition 8

Let a formal logic program $\mathscr{P}$, a literal marker Ω for $\mathscr{P}$, and a foundational extensor E for $(\mathscr{P}, \Omega)$ be given. Then $[\mathscr{P} +_\Omega E]$ is a foundational extensor for $(\mathscr{P}, \Omega)$.

Proof

By Property 13, choose an ordinal α and an α-foundational chain $(E_\beta)_{\beta<\alpha}$ for $(\mathscr{P}, \Omega)$ with $\bigcup_{\beta<\alpha} E_\beta = E$. Set $E_\alpha = [\mathscr{P} +_\Omega E]$. Using Property 9, Lemma 4 and Proposition 7, it is easy to verify that $(E_\beta)_{\beta\leqslant\alpha}$ is an $(\alpha + 1)$ foundational chain for $(\mathscr{P}, \Omega)$. We conclude with Property 13 again. $\square$

As mentioned in the discussion following Example 6, our framework and the well-founded semantics of a formal logic program $\mathscr{P}$ can be related using either $\langle\cdot,-\rangle_{\mathscr{P}}$ or $\langle -,-\rangle_{\mathscr{P}}$; this will be a consequence of the property that follows.

Property 15

Let a formal logic program $\mathscr{P}$ and a set E of negated atoms over $\mathcal{V}^\star$ be given. Then E is a foundational extensor for $(\mathscr{P}, \langle\cdot,-\rangle_{\mathscr{P}})$ iff E is a foundational extensor for $(\mathscr{P}, \langle -,-\rangle_{\mathscr{P}})$.

We now state a counterpart to Corollary 4 for foundational chains.

Proposition 9
Let a formal logic program $\mathscr{P}$ be locally consistent. Let a literal marker Ω for $\mathscr{P}$ and a set I be given. Let a set of foundational chains for $(\mathscr{P}, \Omega)$ 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 $(\mathscr{P}, \Omega)$ 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 ordinals α, 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 $(\mathscr{P}, \Omega)$. Trivially, if $\alpha = 0$ or if α is a limit ordinal then $(F_\beta)_{\beta<\alpha}$ is an α-foundational chain for $(\mathscr{P}, \Omega)$. Assume that α is of the form $\delta + 1$. To complete the proof of the proposition, it is clearly sufficient to show that $F_\delta \cup \big[(\mathscr{P} +_\Omega \bigcup_{\gamma<\delta} F_\gamma) +_{\Omega+\bigcup_{\gamma<\delta} F_\gamma} F_\delta\big]$ is consistent. By Property 9, it suffices to verify that $F_\delta \cup [\mathscr{P} +_\Omega \bigcup_{\gamma\leqslant\delta} F_\gamma]$ is consistent. But this is an immediate consequence of the fact that by Propositions 3 and 7, $\bigcup_{\gamma\leqslant\delta} F_\gamma$, equal to $\bigcup_{\sigma\in I} \bigcup_{\gamma\leqslant\delta} E^\sigma_\gamma$, is an extensor for $(\mathscr{P}, \Omega)$. $\square$

As an application of Proposition 9, we can follow the main path in the field of logic programming, be biased toward negative information, and get the following proposition.

Proposition 10
Let a locally consistent formal logic program $\mathscr{P}$ be given.

- There exists a unique $\subseteq$-maximal set E of negated atoms over $\mathcal{V}^\star$ that is a foundational extensor for $(\mathscr{P}, \langle\cdot,-\rangle_{\mathscr{P}})$, or equivalently, for $(\mathscr{P}, \langle -,-\rangle_{\mathscr{P}})$.
- There exists a unique $\subseteq$-maximal set F of literals over $\mathcal{V}^\star$ that is a foundational extensor for $(\mathscr{P}, \langle\cdot,-\rangle_{\mathscr{P}})$, or equivalently, for $(\mathscr{P}, \langle -,-\rangle_{\mathscr{P}})$; moreover, F is equal to both $[\mathscr{P} +_{\langle\cdot,-\rangle_{\mathscr{P}}} E]$ and $[\mathscr{P} +_{\langle -,-\rangle_{\mathscr{P}}} E]$.

Proof
The existence of a unique $\subseteq$-maximal set E of negated atoms over $\mathcal{V}^\star$ that is a foundational extensor for $(\mathscr{P}, \langle\cdot,-\rangle_{\mathscr{P}})$, or equivalently, for $(\mathscr{P}, \langle -,-\rangle_{\mathscr{P}})$, follows immediately from Proposition 9 and Property 15. Let Ω denote either $\langle\cdot,-\rangle_{\mathscr{P}}$ or $\langle -,-\rangle_{\mathscr{P}}$. By Proposition 8, $[\mathscr{P} +_\Omega E]$ is a foundational extensor for $(\mathscr{P}, \Omega)$. Let $(F_\alpha)_{\alpha\in\mathrm{Ord}}$ be a foundational chain for $(\mathscr{P}, \Omega)$. For all ordinals α, let G_α be the set of negated atoms in F_α. We show that $(G_\alpha)_{\alpha\in\mathrm{Ord}}$ is a foundational chain for $(\mathscr{P}, \Omega)$. Proof is by induction, so let $\alpha \in \mathrm{Ord}$ be given, and assume that for all $\beta < \alpha$, $(G_\gamma)_{\gamma<\beta}$ is a β-foundational chain for $(\mathscr{P}, \Omega)$. Trivially, if $\alpha = 0$ or α is a limit ordinal then $(G_\beta)_{\beta<\alpha}$ is an α-foundational chain for $(\mathscr{P}, \Omega)$. Suppose that α is of the form $\delta + 1$. Obviously, $[\mathscr{P} +_\Omega \bigcup_{\beta<\delta} F_\beta] = [\mathscr{P} +_\Omega \bigcup_{\beta<\delta} G_\beta]$. This together with the fact that F_δ is a supporting extensor for $\big([\mathscr{P} +_\Omega \bigcup_{\beta<\delta} F_\beta], \Omega + \bigcup_{\beta<\delta} F_\beta\big)$ implies immediately that G_δ is a supporting extensor for $\big([\mathscr{P} +_\Omega \bigcup_{\beta<\delta} G_\beta], \Omega + \bigcup_{\beta<\delta} G_\beta\big)$, which completes the proof that $(G_\alpha)_{\alpha\in\mathrm{Ord}}$ is a foundational chain for $(\mathscr{P}, \Omega)$. Obviously, $[\mathscr{P} +_\Omega \bigcup_{\alpha\in\mathrm{Ord}} G_\alpha] = [\mathscr{P} +_\Omega \bigcup_{\alpha\in\mathrm{Ord}} F_\alpha]$. Moreover, $\bigcup_{\alpha\in\mathrm{Ord}} G_\alpha$ is a subset of E. Hence F, which is a subset of $[\mathscr{P} +_\Omega \bigcup_{\alpha\in\mathrm{Ord}} F_\alpha]$ by Proposition 7, is included in $[\mathscr{P} +_\Omega E]$ by Lemma 2. Hence $[\mathscr{P} +_\Omega E]$ is the unique $\subseteq$-maximal set F of literals over $\mathcal{V}^\star$ that is a foundational extensor for $(\mathscr{P}, \Omega)$. $\square$

5.7 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 unnecessarily 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 20 captures the notion of well-founded model. Here not assuming that $\mathcal{V}^\star$ and $\mathcal{V}$ are equal offers a genuine generalisation.

Definition 20

Let a formal logic program $\mathscr{P}$ 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 an ordinal α 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 the $\subseteq$-smallest set X of atoms over $\mathcal{V}^\star$ such that for all $\psi \in X$, $\bigcup_{\beta<\alpha} E^-_\beta \cup X$ forces $\mathscr{P}[\psi]$.
- 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 $\psi \in X$, $\bigcup_{\beta<\alpha} E^+_\beta \cup X$ forces $\mathscr{P}[\psi]$.

Set $E = \bigcup_{\alpha\in\mathrm{Ord}}(E^+_\alpha \cup E^-_\alpha)$. If E is consistent, then $\mathscr{P}$ is said to *have a well-founded model* and E is called the *well-founded model of* $\mathscr{P}$.

Property 16

Let a formal logic program $\mathscr{P} = (\varphi^\epsilon_\wp)_{(\wp,\epsilon)\in\mathbb{I}(\mathcal{V}^\star)}$ be given. Let $(E^+_\alpha)_{\alpha\in\mathrm{Ord}}$ and $(E^-_\alpha)_{\alpha\in\mathrm{Ord}}$ be the two sequences of sets of literals defined in Definition 20. Then for all ordinals α, $\{\odot^-_{\bigcup_{\beta<\alpha} E^-_\beta} \varphi^+_\wp \rightarrow \wp(v_1,\dots,v_n) \mid n \in \mathbb{N},\ \wp \in \mathrm{Prd}(\mathcal{V}^\star,n)\} \vDash_{\mathcal{W}} E^+_\alpha$.

Recall that by Proposition 10, we can talk about 'the $\subseteq$-maximal foundational extensor for $(\mathscr{P}, \langle\cdot,-\rangle_{\mathscr{P}})$' when $\mathscr{P}$ is locally consistent. The next proposition shows that this extensor fully characterises 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 11

Let $\mathscr{P}$ be a locally consistent formal logic program, and let F be the $\subseteq$-maximal foundational extensor for $(\mathscr{P}, \langle\cdot,-\rangle_{\mathscr{P}})$. Then $\mathscr{P}$ has a well-founded model, which is precisely the set of closed instances of members of F.

Proof

Let $(E^+_\alpha)_{\alpha\in\mathrm{Ord}}$ and $(E^-_\alpha)_{\alpha\in\mathrm{Ord}}$ be the sequences of literals defined in Definition 20. For all ordinals α, let D^+_α be the set of atoms over $\mathcal{V}^\star$ all of whose closed instances belong to E^+_α, and let D^-_α be the set of negated atoms over $\mathcal{V}^\star$ all of whose closed instances belong to E^-_α. Note that for all $\alpha \in \mathrm{Ord}$, E^+_α and E^-_α are the sets of closed instances of members of D^+_α and D^-_α, respectively. Set $E = \bigcup_{\alpha\in\mathrm{Ord}}(E^+_\alpha \cup E^-_\alpha)$. By Proposition 10, let $(F_\alpha)_{\alpha\in\mathrm{Ord}}$ be a foundational chain for $(\mathscr{P}, \langle\cdot,-\rangle_{\mathscr{P}})$ such that all members of $\bigcup_{\alpha\in\mathrm{Ord}} F_\alpha$ are negated atoms and $F = [\mathscr{P} +_{\langle\cdot,-\rangle_{\mathscr{P}}} \bigcup_{\alpha\in\mathrm{Ord}} F_\alpha]$.

We first show that for all ordinals α, $(D^-_\beta)_{\beta<\alpha}$ is an α-foundational chain for $(\mathscr{P}, \langle\cdot,-\rangle_\mathscr{P})$. Proof is by induction, so let ordinal α be given, and assume that for all $\beta < \alpha$, $(D^-_\gamma)_{\gamma<\beta}$ is a β-foundational chain for $(\mathscr{P}, \langle\cdot,-\rangle_\mathscr{P})$. Trivially, if $\alpha = 0$ or α is a limit ordinal then $(D^-_\beta)_{\beta<\alpha}$ is an α-foundational chain for $(\mathscr{P}, \langle\cdot,-\rangle_\mathscr{P})$. Suppose that α is of the form $\delta+1$. Let $\psi \in D^-_\delta$ be given. Then $\bigcup_{\beta<\delta} E^+_\beta \cup D^-_\delta$ forces $\mathscr{P}[\psi]$. Together with Property 16, this implies that $[\mathscr{P} +_{\langle\cdot,-\rangle_\mathscr{P}} \bigcup_{\beta<\delta} D^-_\beta\,] \cup D^-_\delta$ forces $\mathscr{P}[\psi]$, hence also $(\mathscr{P} +_{\langle\cdot,-\rangle_\mathscr{P}} \bigcup_{\beta<\delta} D^-_\beta)[\psi]$, which together with Corollary 2 implies that $(D^-_\beta)_{\beta<\alpha}$ is an α-foundational chain for $(\mathscr{P}, \langle\cdot,-\rangle_\mathscr{P})$, as wanted. Now by the definition of F and Proposition 8, $[\mathscr{P} +_{\langle\cdot,-\rangle_\mathscr{P}} \bigcup_{\alpha\in\mathrm{Ord}} D^-_\alpha\,]$ is included in F. We conclude with Lemma 2, Property 16 again and Property 8 that the set of closed instances of members of F contains E, which is therefore consistent. Hence $\mathscr{P}$ has a well-founded model, which is E.

To establish the converse, we show by induction that for all ordinals α,

- all closed instances of members of F_α belong to E, and
- all closed instances of members of $[\mathscr{P} +_{\langle\cdot,-\rangle_\mathscr{P}} \bigcup_{\beta<\alpha} F_\beta\,]$ belong to E.

So let an ordinal α be given and assume that (i) for all ordinals $\beta < \alpha$, all closed instances of members of F_β belong to E, and (ii) for all ordinals $\beta < \alpha$, all closed instances of members of $[\mathscr{P} +_{\langle\cdot,-\rangle_\mathscr{P}} \bigcup_{\gamma<\beta} F_\gamma\,]$ belong to E. Note the following:

($\star$) for all literals ψ over $\mathcal{V}^\star$, $(\mathscr{P} +_{\langle\cdot,-\rangle_\mathscr{P}} \bigcup_{\beta<\alpha} F_\beta)[\psi] \cup \bigcup_{\beta<\alpha} F_\beta$ forces $\mathscr{P}[\psi]$.

Let an ordinal δ be such that $E = E^+_\delta \cup E^-_\delta$. Using ($\star$), we obtain by induction that for all $\gamma \in \mathrm{Ord}$ and $\psi \in [\mathscr{P} +_{\langle\cdot,-\rangle_\mathscr{P}} \bigcup_{\beta<\alpha} F_\beta\,]_\gamma$, if $\bigcup_{\beta'<\gamma}[\mathscr{P} +_{\langle\cdot,-\rangle_\mathscr{P}} \bigcup_{\beta<\alpha} F_\beta\,]_{\beta'}$ forces $(\mathscr{P} +_{\langle\cdot,-\rangle_\mathscr{P}} \bigcup_{\beta<\alpha} F_\beta)[\psi]$ then $\bigcup_{\beta'<\gamma}[\mathscr{P} +_{\langle\cdot,-\rangle_\mathscr{P}} \bigcup_{\beta<\alpha} F_\beta\,]_{\beta'} \cup \bigcup_{\beta<\alpha} F_\beta$ forces $\mathscr{P}[\psi]$; this together with (i) easily implies that if X^+ and X^- denote the set of atoms and the set of negated atoms in $\bigcup_{\beta'<\gamma}[\mathscr{P} +_{\langle\cdot,-\rangle_\mathscr{P}} \bigcup_{\beta<\alpha} F_\beta\,]_{\beta'} \cup \{\psi\}$, respectively, then

- the set of closed instances of members of X^+ is included in $E^+_{\delta+1}$, and
- the set of closed instances of members of $X^- \cup \bigcup_{\beta<\alpha} F_\beta$ is included in $E^-_{\delta+1}$.

Hence all closed instances of members of $[\mathscr{P} +_{\langle\cdot,-\rangle_\mathscr{P}} \bigcup_{\beta<\alpha} F_\beta\,]$ belong to E. Using ($\star$) again, we obtain that for all $\psi \in F_\alpha$, since $[\mathscr{P} +_{\langle\cdot,-\rangle_\mathscr{P}} \bigcup_{\beta<\alpha} F_\beta\,] \cup F_\alpha$ forces $(\mathscr{P} +_{\langle\cdot,-\rangle_\mathscr{P}} \bigcup_{\beta<\alpha} F_\beta)[\psi]$ by Corollary 2, then $E \cup F_\alpha$ forces $\mathscr{P}[\psi]$; this easily implies that the set consisting of the closed instances of either the negated atoms in E or the members of F_α is included in $E^-_{\delta+1}$. Hence all closed instances of members of F_α belong to E. Since F is equal to $[\mathscr{P} +_{\langle\cdot,-\rangle_\mathscr{P}} \bigcup_{\alpha\in\mathrm{Ord}} F_\alpha\,]$, we have shown that all closed instances of members of F belong to E, which completes the proof of the proposition. $\square$

6 Conclusion

Given a formal logic program $\mathscr{P}$, we have defined the set $[\mathscr{P}\,]$ of literals generated by $\mathscr{P}$ following a process that can be intuitively described as: fire the rules in $\mathscr{P}$ transfinitely often, and at each stage interpret disjunction and existential quantification constructively to determine whether an instance of the body of a rule should be activated, the rule fired, and the corresponding instance of the head added to $[\mathscr{P}]$. The view that has been adopted is that $[\mathscr{P}]$ captures the operational semantics of $\mathscr{P}$. This view is closely related to Kripke–Kleene semantics (this is the contents of Proposition 1). We have introduced the notion of 'literal marker for $\mathscr{P}$' to formalise the intuitive idea of 'marking some literals in the bodies of some rules in $\mathscr{P}$'. Given such a literal marker Ω for $\mathscr{P}$ and a set E of literals conceived of as a collection of hypotheses, meant to be assumed only in the contexts authorised by Ω, we have formalised the intuitive operation of making these contextual, local assumptions, resulting in a new formal logic program, denoted $\mathscr{P}+_\Omega E$; the denotational semantics of that program is of course captured by $[\mathscr{P}+_\Omega E]$. For a given literal marker Ω for $\mathscr{P}$ and a given set E of literals, $[\mathscr{P}+_\Omega E]$ can also be conceived of as an alternative semantics to $\mathscr{P}$, and we have seen how to choose Ω and E in order to retrieve the answer-set, the stable model and the well-founded semantics.

- Answer-sets are captured by sets of the form $[\mathscr{P}+_\Omega E]$ in which Ω marks the occurrences of literals of the form $\neg$*atom*, represented in the usual setting as *not atom*, or of the form *atom*, represented in the usual setting as *not* $\neg$*atom*, and E is a maximal (in a strong sense) set of literals that $\mathscr{P}+_\Omega E$ does not refute (this is the contents of Proposition 5).
- Stable models are captured by sets of the form $[\mathscr{P}+_\Omega E]$ in which Ω marks all occurrences of negated atoms in the bodies of all rules, and E is a maximal set of negated atoms which determines a complete set of literals that $\mathscr{P}+_\Omega E$ confirms (this is the contents of Proposition 6).
- The well-founded model is the set $[\mathscr{P}+_\Omega E]$ in which Ω marks all occurrences of negated atoms in the bodies of all negative rules, and E is the maximal set of negated atoms that $\mathscr{P}+_\Omega E$ confirms in a strong sense, based on the concept of a set of hypotheses that can get 'self-confirmation' with no additional help but what can be derived from $\mathscr{P}$ itself, put into action transfinitely often (this is the contents of Propositions 10 and 11).

The relationships have actually been established for a class of logic programs more general than those usually considered in the literature, but for which those semantics could be naturally adapted. The classes of extensors (legitimate sets of hypotheses) that have been introduced can be subjected to natural variations; the choices for Ω can range from fully biased toward negated atoms to fully biased toward nonnegated atoms, or seek some balance between both kinds of literals, etc. Hence the three semantics captured by $[\mathscr{P}+_\Omega E]$ for the specific choices of Ω and E that have been described are members of families of semantics determined by a pair (Ω, E) that naturally satisfies more general properties. We have not investigated these alternative semantics for lack of space, but we think that one of the main contributions of this paper is to have laid the foundation for such a work, with applications to hypothetical reasoning in knowledge-based systems, where hypotheses are applied locally and contextually, and are constrained to satisfy variations on properties such as confirmation or nonrefutation.

Though Kripke–Kleene, the answer-set, the stable model and the well-founded semantics are expressed in terms of 'intended' or 'preferred' models, we do not view $[\mathscr{P}]$ as the intended model of what we have called the classical logical form, denoted

Clf($\mathscr{P}$), of the formal logic program $\mathscr{P}$. Indeed, 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 provide an operational semantics, which does not require to represent rules as logical implications. Another paper will present a declarative semantics, in such a way that $\{\Box\varphi \mid \varphi \in [\mathscr{P}]\}$ is precisely the set of formulas of the form $\Box\varphi$ with φ a literal that are logical consequences of a set of modal formulas that is obtained from Clf($\mathscr{P}$) by preceding all occurrences of literals with the modal operator of necessity. (The main work is to capture properly the transformation of $\mathscr{P}$ into $\mathscr{P} +_\Omega E$—marking literals has to find its logical counterpart—and to properly represent the hypotheses.) In this setting, 'logical consequence' is interpreted classically, that is, in reference to a notion of interpretation that generalises the interpretations used in epistemic logic, in which every formula is either true or false (not undefined), negation is classical, and the law of excluded middle holds but is irrelevant, because a rule such as $q \leftarrow p \vee \neg p$ is logically translated into $\Box p \vee \Box\neg p \rightarrow \Box q$: to derive q, derive p or derive $\neg p$, and $q \leftarrow p \vee \neg p$ does not automatically fire because $\Box p \vee \Box\neg p$ is not valid.

References

Alferes, J. J., Pereira, L. M. and Przymusinski, T. C. 1996. Strong and explicit negation in non-monotonic reasoning and logic programming. In *Proceedings of the European Workshop on Logics in Artificial Intelligence*. Lecture Notes in Computer Science, vol. 1126. Springer-Verlag, London, 143–163.

Alferes, J. J., Pereira, L. M. and Przymusinski, T. C. 1998. 'Classical' negation in nonmonotonic reasoning and logic programming. *Journal of Automated Reasoning 20*, 1–2 (April), 107–142.

Apt, K. R. and Bol, R. 1994. Logic programming and negation: A survey. *Journal of Logic Programming 19–20*, Supplement 1 (May–July), 9–71.

Clark, K. L. 1987. Negation as failure. In *Readings in Nonmonotonic Reasoning*, M. L. Ginsberg, Ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, 311–325.

Denecker, M., Bruynooghe, M. and Marek, V. 2001. Logic programming revisited: Logic programs as inductive definitions. *ACM Transactions on Computational Logic 2*, 4 (October), 623–654.

Emden, M. H. V. and Kowalski, R. A. 1976. The semantics of predicate logic as a programming language. *Journal of the Association for Computing Machinery 23*, 4 (October), 733–742.

Fitting, M. 1985. A Kripke–Kleene semantics for logic programs. *Journal of Logic Programming 2*, 4 (December), 295–312.

Fitting, M. 1999. Fixpoint semantics for logic programming – A survey. *Theoretical Computer Science 278*, 25–51.

Gelder, A. V., Ross, K. A. and Schlipf, J. S. 1991. The well-founded semantics for general logic programs. *Journal of the Association for Computing Machinery 38*, 3 (July), 620–650.

Gelfond, M. and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Logic Programming: Proceedings of the Fifth International Conference and Symposium*, R. A. Kowalski and K. A. Bowen, Eds. MIT Press Series in Logic Programming, vol. 2. MIT Press, Manchester, UK, 1070–1080.

Gelfond, M. and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing 9*, 3–4, 365–385.

Hitzler, P. 2005. Towards a systematic account of different semantics for logic programs. *Journal of Logic and Computation 15*, 3 (June), 391–404.

Jäger, G. and Stärk, R. F. 1993. The defining power of stratified and hierarchical logic programs. *Journal of Logic Programming 15*, 1–2 (January), 55–77.

Kaminski, M. and Rey, G. 2002. Revisiting quantification in autoepistemic logic. *ACM Transactions on Computational Logic 3*, 4 (October), 542–561.

Lloyd, J. W. 1987. *Foundations of Logic Programming*, 2nd ed. Springer-Verlag, New York.

Loyer, Y., Spyratos, N. and Stamate, D. 2003. Parametrized semantics of logic programs: A unifying framework. *Theoretical Computer Science 308*, 1–3 (November), 429–447.

Marek, W. and Truszczyński, M. 1991. Autoepistemic logic. *Journal of the Association for Computing Machinery 38*, 3 (July), 587–618.

Martin, E. 2006. Quantification over names and modalities. In *Advances in Modal Logic*, vol. 6, G. Governatori, I. Hodkinson and Y. Venema, Eds. College Publications, London, 353–372.

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

Minker, J. and Seipel, D. 2002. Disjunctive logic programming: A survey and assessment. In *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski*, A. C. Kakas and F. Sadri, Eds. Lecture Notes in Computer Science, vol. 2407–2408. Springer-Verlag, London, 472–511.

Moore, R. C. 1985. Semantical considerations on nonmonotonic logic. *Artificial Intelligence 25*, 1 (January), 75–94.

Pearce, D. 2006. Equilibrium logic. *Annals of Mathematics and Artificial Intelligence 47*, 1–2 (June), 3–41.

Pedreschi, D., Ruggieri, S. and Smaus, J.-G. 2002. Classes of terminating logic programs. *Theory and Practice of Logic Programming 2*, 3 (May), 369–418.