The Operational Semantics of Dual Logic Programming

Eric A. Martin
School of Computer Science and Engineering
The University of New South Wales, Australia
emartin@cse.unsw.edu.au

Technical Report UNSW-CSE-TR-0706

February 2007

Abstract

Many developments in the field of Logic programming reveal poor choices of basic concepts, and misleading views on negation. We show that Logic programming can enjoy a general, simple and clean foundation, provided that the basic concepts be revisited, and that any nonclassical form of negation, in particular, negation as finite failure, be disregarded. We propose a framework that starts with a ruled-based notion of *dual logic program*, allowing negation to appear in the heads as well as in the bodies of the rules. Dual logic programs can be treated as very general logic programs, but it is conceptually more beneficial to view them as redefined logic programs. Classical semantics, such as the Kripke-Kleene semantics, the well-founded semantics, and the stable model semantics, are redefined as natural program transformations of the same kind, resulting into dual logic programs whose behaviors can be described in terms of the same notion of logical consequence in a classical semantics. A kind of inference "in the style of Logic programming" from arbitrary theories is presented, together with a natural method to transform arbitrary theories into dual logic programs.

1 Some remarks on the field of Logic programming

The field of Logic programming is more evocative of the bush than of a formal garden. This is not because of an intricate nature; rather, the tangle of notions and results sprouts from an unfortunate choice of basic concepts. This regrettable state of affairs comes from two erroneous beliefs.

- The erroneous belief that negation should be treated nonclassically, and that finding out precisely how it should be treated is a hard task, as if the world of Logic programming were one where negation is a sibyl, requiring researchers in the field to dedicate themselves to understanding her answers.
- The erroneous belief that the operational and the logical interpretations of a logic program can be adequately expressed by using the same language and writing down the same set of sentences, as if the world of Logic programming were one where the same words could describe both how a dish is prepared and how it tastes, by just switching between two possible meanings.

Arguing against both beliefs and proposing an alternative requires two papers. In this paper, we show that life is better in worlds without sibyls.

In [14], Rondogiannis and Wadge write: "*There is a general feeling* [. . .] *that when one seeks a unique model, then the well-founded semantics is the right approach to negation-as-failure.* [. . .] *Our goal is to remove the last doubts surrounding the well-founded model* [. . .]. This is symptomatic of much of the work in Logic programming, where negation-as-failure is conceived of as some form of physical reality that has to be modeled. It is hard to understand how nearly thirty years of research in the field result in *feelings* and *doubts* about the meaning of a few sequences of symbols like the following.

$$\begin{aligned} p &\leftarrow \neg p \\ q &\leftarrow \neg p \vee r \\ r &\leftarrow r \end{aligned}$$

Rather than starting with a clear notion and formalizing it as statements in an appropriate language endowed with an appropriate semantics, most research in Logic programming starts with statements expressed in a given language, and embarks on discovering how to endow that language with a semantics appropriate to reveal 'some' notion, as if sequences of symbols like the one above were a natural object of study, a kind of hieroglyphs for modern Champollions.

Many overviews of the various semantics for logic programs that have been proposed are available [1, 2], and we suggest that the interested reader refers to

them. Here we only point to some aspects of some of those semantics that are particularly disturbing, with no claim nor attempt at doing this exercise exhaustively or systematically.

In Logic programming, a statement of the form $p \leftrightarrow \neg p$ is usually not viewed as degenerate (inconsistent); it has prompted researchers to move from a 2-valued semantics to a 3-valued semantics, and then to a 4-valued semantics, the latter being eventually generalized to bilattices such as $\mathcal{FOUR}$. See [3] for a comparison between 3- and 4-valued logics.

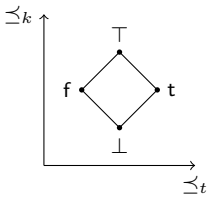

Figure 1: The bilattice $\mathcal{FOUR}$

With the *knowledge ordering* $\preceq_k$, we can conceive of an agent who, presented with a statement, could have a definite idea about its truth—expressed as either `t` or `f`—, or be unsure—expressed as $\bot$—, or be confused—expressed as $\top$. Still it is difficult to understand what $\top$ and $\bot$ represent with respect to the *truth ordering* $\preceq_t$. It could be argued that a statement can be neither true nor false in case it fails to denote properly the objects and the properties of an intended model, hence the question whether that statement is true or false would not make sense, but that would require one extra value besides `t` and `f`, not two, and that value would not correspond at all to the intuitive meaning of either $\bot$ or $\top$ in the knowledge ordering. This is all the more puzzling that the semantics of logic programs based on bilattices entwines $\preceq_k$ and $\preceq_t$ by computing fixed points with respect to the truth ordering, and then selecting those fixed points that are minimal with respect to the knowledge ordering [7].

Rondogiannis and Wadge argue that *a purely declarative semantics for logic programs with negation-as-failure should be based on an infinite-valued logic*. By using *should*, Rondogiannis and Wadge oppose Fitting who claims that *bilattices are the right abstract tool* [7]. In their semantics, Rondogiannis and Wadge do not only introduce two *infinitesimal truth values* $\texttt{true}_\alpha$ and $\texttt{false}_\alpha$ for every ordinal α, unconvincingly argued to be *closely related to the idea of infinitesimals used in Nonstandard Analysis*; they also need a value larger than any $\texttt{false}_\alpha$ and smaller than any $\texttt{true}_\alpha$. This value is the counterpart to the $\bot$ of the 3- and 4-valued semantics, and basically means *neither provable nor refutable*. So Rondogiannis

and Wadge's approach is a variation on the principle that by calling *provable* `true`, *refutable* `false` and *neither provable nor refutable* $\bot$, a semantics has been defined, as if calling *provable* good, *refutable* evil and *neither provable nor refutable* faulty would define a morality.

2 The intuitive notion of a logic program

Logic programming is the discipline that studies logic programs. So the first question to ask is: what is a logic program? A logic program is a set of *rules* of the form

$$head \leftarrow body$$

where *head* is a *simple* statement and *body* a *compound statement*, built from simple statements and a few operators (so simple statements are particular kinds of compound statements). As possible operators, take *disjunction* $\bigvee$, *conjunction* $\bigwedge$, *existential quantification* $\exists$, and *universal quantification* $\forall$.

Rules can *fire* and simple statements can be *directly generated*. The directly generated simple statements can be combined into compound statements in all possible ways that, interpreted using the laws of *classical logic* applied to *intended interpretations*, allow one to talk about *indirectly generated* compound statements. For instance, if exactly two simple statements, say p and q, have been directly generated, then the compound statements $p \wedge q$ and $p \vee r$ have been indirectly generated, whereas the compound statement $p \wedge r$ has not.

Of particular importance is the framework where rules are allowed to fire any finite number of times, as this immediately suggests obvious implementations. But we can think theoretically and assume that rules are allowed to fire transfinitely many times. So after all rules have fired any finite number of times, they could fire for the ω-th time, and then for the $(\omega+1)$-st time, and then for the $(\omega+2)$-nd time . . . and then for the $(\omega\times 2)$-nd time, etc. For instance, if all simple statements of the form $p(\overline{n})$, $n \in \mathbb{N}$, have been directly generated before stage ω, and if all individuals in the domains of all possible interpretations are denoted by a term of the form $\overline{n}$, then $\forall x p(x)$ has been indirectly generated and at stage ω, any rule whose body is $\forall x p(x)$ can fire.

What is a simple statement? It could be any *atom*. That works nicely and unproblematically, and the resulting framework is Logic programming without negation. It can also be any *literal*, *i.e.*, atom or negated atom. That works nicely and unproblematically, and the resulting framework is Dual Logic programming, which is the object of this paper.

Without negation, compound statements are indirectly generated *constructively*: $p \vee q$ will be indirectly generated iff p or q has been directly generated; pro-

vided that the class of intended interpretations has been defined properly, $\exists x p(x)$ will be indirectly generated iff $p(t)$ has been directly generated for at least one closed term t. There is absolutely no reason why this constructiveness property should be abandoned when negation enters the stage. The constructiveness property adequately models one of the key properties of a mechanistic device; it matches the intuitive notion of a logic program. So if $p \vee \neg p$ is the body of a rule, that rule will fire iff either p or $\neg p$ has been directly generated.

What if a rule fires and directly generates p, and another rule fires and directly generates $\neg p$? Logic programs where an atom and its negation can both be generated are a form of inconsistent theory. There is no need to torture oneself because of the existence of inconsistent logic programs, and expect researchers to design intricate semantics meant to turn them into allegedly more respectable objects. Inconsistent logic programs can and should just be seen as degenerate cases. In particular, a consistent logic program might contain a rule of the form $p \leftarrow \neg p$, but that rule will never fire, and neither $\neg p$ nor p will ever be generated, which is no more problematic than a theory not being complete. Indeed, for $p \leftarrow \neg p$ to fire, it would be necessary for $\neg p$ to have been first directly generated from a rule whose head is $\neg p$—something a consistent logic program does not do.

Obviously, if a logic program $\mathcal{P}$ contains a rule of the form $p \leftarrow p$, that rule cannot ever fire: if it fires then it fires for a first time, which requires p to have been generated and the rule to have fired earlier. Still most semantics for logic programs are based on the assumption that if a logic program contains $p \leftarrow p$ then p should be false in all intended models of that logic program. How researchers have come to accept this claim as an obvious truth is beyond the author's understanding. But we can conceive that it is possible to *transform* a logic program and make sure that it generates all the simple statements generated by the original program, plus possibly more. That is fine, as long as it is clear that we are dealing with the transformed logic program, not the original one.

The well-founded semantics [17] and the stable model semantics [8] can be seen as particular examples of a transformation principle. The transformation can be made generic and treat nonnegated atoms on a par with negated atoms. For instance, consider the logic program whose only rules are $p \leftarrow p$ and $\neg p \leftarrow \neg p$. This logic program can be transformed into another logic program that generates $\neg p$, but not p, in accordance with the well-founded and stable model semantics. It can also be transformed into yet another logic program that generates p, but not $\neg p$. Dual Logic programming is the nonlimping form of Logic programming where nonnegated atoms and negated atoms play a dual role.

3 Negation-as-finite-failure and classical negation

Let $\mathcal{P}$ be a logic program whose rules are derived from *definite clauses*; they are of the form $\psi \leftarrow \exists \vec{y} \chi$ where χ is a negation and quantifier free formula, and $\vec{y}$ consists of the variables that occur in χ but not in ψ. Give $\mathcal{P}$ to Prolog; it will do a nice job at answering the kind of queries it was meant to answer, namely, *definite queries*—statements of the form $\exists \vec{x} \varphi$ where φ is a quantifier and negation free formula all of whose variables are amongst $\vec{x}$. Who would resist the temptation of asking more general queries, with a few malicious negations inserted before some of the atoms in φ? Surely, only a sibyl would know the answer to such queries, since the set of definite clauses does not allow one to derive any negative information. Prolog is not a sibyl; it just performs its derivations on the basis of a logic program that is different to the one it has been provided with.

For an example, take for $\mathcal{P}$ the logic program consisting of the fact $even(\overline{0})$ and the rule $even(s(s(x))) \leftarrow even(x)$. The logic program that Prolog uses is the logic program $\mathcal{P}'$ that can be represented as follows.

$$\begin{array}{cc} even(x) \leftarrow \exists y \big(x = s(s(y)) \wedge even(y)\big) & even(\overline{0}) \\ \neg even(x) \leftarrow \forall y \big(x \neq s(s(y)) \vee \neg even(y)\big) & \neg even(s(\overline{0})) \end{array}$$

When asked to answer a definite query, Prolog *could* apply forward chaining, as the only rules it will have to use are the first two. When asked to answer a query that contains negation, it *has to* do something else (of course, we all know that it *always* does something else, whether it has to or not). It has to do something else because the body of the third rule is universal, not existential, and given a closed term t, Prolog cannot generate $\neg even(t')$ or $t \neq t'$ for infinitely many closed terms t'. It can fire rules finitely many times only, not ω times or more.

Actually, Prolog never fires any rule, as it proceeds backwards, from the heads of the rules to their bodies. This is very smart: Prolog can produce precisely the set of simple statements that can be produced by the kind of firing process that has been described, with rules still firing at stage ω, $\omega+1$, etc. We are quite happy to have a system that can *simulate* the transfinite behavior of our 'abstract' logic programs. So we did not only think theoretically after all, since we can find a way to overcome the physical limitations of our model.

The previous discussion does not really bring anything that anyone who knows a bit about Logic programming does not know. The firing process is the application of the immediate consequence operator [16]. The logic program $\mathcal{P}'$ is a variation on Clark's completion of $\mathcal{P}$ [5]. So what is all the fuss about? Well, the fuss is about realizing that two attitudes are possible.

- Think that classical negation is appropriate and that fundamentally, no other

form of negation is needed. Develop the theory on the basis of the logic programs that one *really* has in mind.

- Think that negation is a source of problem and that classical negation is inadequate. Distinguish between the logic program that one writes down ($\mathcal{P}$) and the logic program that one has in mind ($\mathcal{P}'$ or other), still having strong doubts about what one really has in mind (Kripke-Kleene semantics versus other semantics). Convince oneself that even though one has something else in mind than what one writes down, it is a brilliant idea not to write down what one has in mind, and that one should explain the behavior of what one has in mind on the basis of what one writes down, pretending or not to ignore that this will likely create a big mess.

Besides that, there is no need for a fuss.

4 Background notions and notation

$\mathbb{N}$ denotes the set of natural numbers and Ord the class of ordinals. We denote by $\mathcal{V}$ a countable *vocabulary*, *i.e.*, a countable set of (possibly nullary) function symbols and (possibly nullary) predicate symbols, one of which might be equality. When we assume that $\mathcal{V}$ contains a constant $\overline{0}$ and a unary function symbol s, we use $\overline{n}$ to refer to the term obtained from $\overline{0}$ by n applications of s. We denote by $\mathrm{Prd}(\mathcal{V})$ the set of predicate symbols in $\mathcal{V}$ distinct from equality. For all $n \in \mathbb{N}$, we denote by $\mathrm{Prd}(\mathcal{V}, n)$ the set of members of $\mathrm{Prd}(\mathcal{V})$ of arity n. We fix a countably infinite set of (first-order) variables, and a repetition-free enumeration $(v_i)_{i\in\mathbb{N}}$ of this set. By *term* we mean term over $\mathcal{V}$. We say that a term is *closed* if it contains no variable. The set $\mathcal{L}_{\omega_1\omega}(\mathcal{V})$ is the smallest set that:

- contain all *atoms*, *i.e.*, expressions of the form $p(t_1, \ldots, t_n)$ where $n \in \mathbb{N}$, $p \in \mathrm{Prd}(\mathcal{V}, n)$, and $t_1, \ldots, t_n$ are terms—written as $t_1 = t_2$ in case of p being equality, and their negations $\neg p(t_1, \ldots, t_n)$—written as $t_1 \neq t_2$ in case of p being equality;

- contains all expressions of the form $\bigvee X$ and $\bigwedge X$ where X is a countable (and possibly empty) set of formulas;

- contains all expressions of the form $\exists x\varphi$ and $\forall x\varphi$ where x is a variable that is free in φ.

Members of $\mathcal{L}_{\omega_1\omega}(\mathcal{V})$ are called *formulas*. A *closed* formula, that is, a formula that contains no free variable, is also called a *sentence*. We will talk about *formula over*

$\mathcal{V}'$ and *sentence over* $\mathcal{V}'$ when we need to consider a vocabulary $\mathcal{V}'$ different to $\mathcal{V}$ and its associated notions of formula and sentence, respectively.

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

A *literal* refers to an atom or a negated atom. A set X of closed literals is said to be *complete* just in case for all closed atoms φ, X contains either φ or $\neg\varphi$, and not both.

Given a formula φ, $\mathrm{fv}(\varphi)$ denotes the set of free variables of φ, $\forall\varphi$ denotes a universal closure of φ, and $\exists\varphi$ denotes an existential closure of φ. Given $n \in \mathbb{N}$, terms $t_1, \ldots, t_n$, and a formula φ, we write $\varphi[t_1/v_1, \ldots, t_n/v_n]$ for the result of substituting simultaneously in φ all free occurrences of $v_1, \ldots, v_n$ by $t_1, \ldots, t_n$, respectively.

Though negation can be applied to atoms only, we need to be able to semantically negate a formula in a syntactically friendly way, 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 the negation of an atom ψ;
- $\bigwedge\{\sim\psi \mid \psi \in X\}$ if φ is $\bigvee X$, and $\bigvee\{\sim\psi \mid \psi \in X\}$ if φ is $\bigwedge X$;
- $\forall x\sim\psi$ if φ is $\exists x\psi$, and $\exists x\sim\psi$ if φ is $\forall x\psi$.

Given two formulas φ_1 and φ_2, both $\varphi_1 \vee \varphi_2$ and $\varphi_2 \vee \varphi_1$ are abbreviations for the formula $\bigvee\{\varphi_1, \varphi_2\}$, both $\varphi_1 \wedge \varphi_2$ and $\varphi_2 \wedge \varphi_1$ are abbreviations for the formula

$\bigwedge\{\varphi_1, \varphi_2\}$, $\varphi_1 \to \varphi_2$ is an abbreviation for the formula $\sim\varphi_1 \vee \varphi_2$, and $\varphi_1 \leftrightarrow \varphi_2$ is an abbreviation for the formula $(\varphi_1 \to \varphi_2) \wedge (\varphi_2 \to \varphi_1)$.

A *standard structure* is a set of closed atoms. Given a standard structure $\mathfrak{M}$ and a sentence φ, we write $\mathfrak{M} \vDash \varphi$ to express that φ is true in $\mathfrak{M}$, or that $\mathfrak{M}$ is a model of φ, interpreted routinely (but slightly more generally than with Herbrand structures, which are another name for the standard structures in the case where $\mathcal{V}$ contains at least one constant and no nullary predicate symbol). Note that in the particular case where $\mathcal{V}$ contains nullary predicate symbols only, a standard structure is equivalent to a propositional interpretation. We denote by $\mathcal{W}$ the set of all standard structures.

Given a set T of sentences and a sentence φ, we write $T \vDash_{\mathcal{W}} \varphi$ if every standard model of T is a model of φ, and $T \nvDash_{\mathcal{W}} \varphi$ otherwise; if $T \vDash_{\mathcal{W}} \varphi$ then we say that T *logically implies* φ *in* $\mathcal{W}$ or that φ *is a logical consequence of* T *in* $\mathcal{W}$. A sentence is said to be *satisfiable in* $\mathcal{W}$ if it has a standard model, and *unsatisfiable in* $\mathcal{W}$ otherwise. A sentence is said to be *valid in* $\mathcal{W}$ if it is true in all standard structures, and *invalid in* $\mathcal{W}$ otherwise. Two formulas φ and ψ are said to be *logically equivalent in* $\mathcal{W}$ if $\forall(\varphi \leftrightarrow \psi)$ is valid in $\mathcal{W}$. The same notation and terminology applies to sets of sentences instead of sentences.

5 Dual logic programs

Recall the fixed enumeration $(v_i)_{i\in\mathbb{N}}$ of the set of variables. A dual logic program has to provide, for every $n \in \mathbb{N}$ and $p \in \mathrm{Prd}(\mathcal{V}, n)$, two rules: one whose head is $p(v_1, \ldots, v_n)$ (which is nothing but p if $n = 0$), and one whose head is $\neg p(v_1, \ldots, v_n)$ (which is nothing but $\neg p$ if $n = 0$). So to define a dual logic program, we only need to provide the antecedents of both rules associated with a predicate symbol and its negation. This is formalized in Definition 1, which generalizes the classical notion of a general logic program. It is to different to the class of logic programs considered in Answer Set Programming [9], as classical negation is the only form of negation that we allow. The fact that disjunction cannot appear in the head is not such an essential difference, as will be seen in Section 9.

Definition 1. A *dual logic program over* $\mathcal{V}$ is defined as a family $\mathcal{P}$ of pairs of formulas over $\mathcal{V} \cup \{=\}$,[1] say $\left((\varphi_p^+, \varphi_p^-)\right)_{p\in\mathrm{Prd}(\mathcal{V})}$, such that for all $n \in \mathbb{N}$ and for all $p \in \mathrm{Prd}(\mathcal{V}, n)$, $\mathrm{fv}(\varphi_p^+) \cup \mathrm{fv}(\varphi_p^-)$ is included in $\{v_1, \ldots, v_n\}$.

The condition on variables is at no loss of generality and is imposed so as to simplify subsequent notation, and is also often used in classical Logic programming;

[1] $\mathcal{V}$ might contain equality, in which case $\mathcal{V} \cup \{=\}$ is $\mathcal{V}$.

it just states that a variable that occurs free in the body of a rule ocurs in the head of that rule.

Note that the rules of a dual logic program can be infinitary. Sometimes, infinitary formulas can be seen as an alternative way of representing an infinite set of formulas, *e.g.*, $\bigvee\{p_i \mid i \in \mathbb{N}\} \to q$ can be seen as an alternative representation of $\{p_i \to q \mid i \in \mathbb{N}\}$.

We will say "logic program" rather than "dual logic program over $\mathcal{V}$." When we need to consider a vocabulary $\mathcal{V}'$ different to $\mathcal{V}$, we say "logic program over $\mathcal{V}'$."

Example 2. Suppose that $\mathcal{V}$ consists of a constant $\overline{0}$, a unary function symbol s, 5 nullary predicate symbols $q_1, \ldots, q_5$, and 4 unary predicate symbols $p_1, \ldots, p_4$. An example of logic program is given by the following formulas.

$$\varphi^+_{p_1} \equiv v_1 = \overline{0} \vee \exists v_0(v_1 = s(s(v_0)) \wedge p_1(v_0))$$
$$\varphi^-_{p_1} \equiv v_1 \neq \overline{0} \wedge \forall v_0(v_1 \neq s(s(v_0)) \vee \neg p_1(v_0))$$

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

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

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

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

Let us take advantage of Example 2 to illustrate how Definition 1 is put to use to represent rules. Recall that $\bigwedge \emptyset$ is valid and $\bigvee \emptyset$ is invalid. Hence

- $\varphi^+_{q_1}$ 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 $q_5 \leftarrow \neg q_5$.

The formulas $\varphi^+_{p_i}$, $i \in \{1,2\}$, represent the rule

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

which could be rewritten as the following two rules.

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

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

$$\neg p_1(v_1) \leftarrow v_1 \neq \overline{0} \wedge \forall v_0 (v_1 \neq s(s(v_0)) \vee \neg p_1(v_0))$$

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

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

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

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

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 from the positive rule. Finally, the formulas $\varphi^+_{p_4}$ 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.

Definition 3. Let a logic program $\mathcal{P} = \left((\varphi^+_p, \varphi^-_p)\right)_{p \in \mathrm{Prd}(\mathcal{V})}$ be given.

We define the *classical logical form of* $\mathcal{P}$ as

$$\Big\{\forall(\varphi^+_p \rightarrow p(v_1, \ldots, v_n)),\ \forall(\varphi^-_p \rightarrow \neg p(v_1, \ldots, v_n)) \mid n \in \mathbb{N},\ p \in \mathrm{Prd}(\mathcal{V}, n)\Big\}.$$

We let $\mathrm{CLF}(\mathcal{P})$ denote the classical logical form of $\mathcal{P}$.

Of course, the classical logical form of $\mathcal{P}$ does not adequately capture the logical meaning of $\mathcal{P}$. For instance, if $\mathcal{P}$ is the logic program described in Example 2 then q_5 is a logical consequence of $\mathrm{CLF}(\mathcal{P})$, but $\mathcal{P}$'s rules do not generate q_5. The traditional attitude is to blame classical negation for the inappropriateness of the classical logical form of a logic program $\mathcal{P}$ as a logical interpretation of $\mathcal{P}$. We do not incriminate negation. Our logical translation of a logic program will leave classical negation in peace. But doing this logical translation will require another paper.

There are two natural conditions that make it impossible to fire a rule that generates a closed atom, and also fire a rule that generates the negation of that atom. They are formalized in Definition 4.

Definition 4. Let a logic program $\mathcal{P} = \left((\varphi_p^+, \varphi_p^-)\right)_{p \in \mathrm{Prd}(\mathcal{V})}$ be given.

- We say that $\mathcal{P}$ is *locally consistent in* $\mathcal{W}$ iff for all $p \in \mathrm{Prd}(\mathcal{V})$, $\exists(\varphi_p^+ \wedge \varphi_p^-)$ is unsatisfiable in $\mathcal{W}$.
- We say that $\mathcal{P}$ is *globally consistent in* $\mathcal{W}$ iff $\mathrm{CLF}(\mathcal{P})$ is consistent in $\mathcal{W}$.

For instance, the logic program described in Example 2 is both locally and globally consistent in $\mathcal{W}$. If $\varphi_{q_5}^-$ were defined as q_5 rather than $\bigvee \emptyset$, then it would be locally consistent in $\mathcal{W}$, but not globally consistent in $\mathcal{W}$. If $\varphi_{q_5}^-$ were defined as $\neg q_5$ rather than $\bigvee \emptyset$, then it would be globally consistent in $\mathcal{W}$, but not locally consistent in $\mathcal{W}$. The notion of local consistency in $\mathcal{W}$ is of particular interest in relation to the transformations that correspond to the well-founded and stable model semantics.

With Clark's completion and Kripke-Kleene semantics in mind, it is legitimate to claim that classical Logic programming is developed on the basis of the particular case of our framework described in the next definition, with $(\varphi_p^-)_{p \in \mathrm{Prd}(\mathcal{V})}$ being left implicit.

Definition 5. Let a logic program $\mathcal{P} = \left((\varphi_p^+, \varphi_p^-)\right)_{p \in \mathrm{Prd}(\mathcal{V})}$ be given. We say that $\mathcal{P}$ is *symmetric* iff for all $p \in \mathrm{Prd}(\mathcal{V})$, $\varphi_p^- = {\sim}\varphi_p^+$.

Property 6. *A symmetric logic program is locally consistent in* $\mathcal{W}$.

6 Generated literals

6.1 The tableau-like approach

Consider a vocabulary $\mathcal{V}$ containing 8 nullary predicate symbols $p_1, \ldots, p_8$, and assume that $\mathcal{P}$ is defined in such a way that we can represent it as a set that

contains the following nine rules.

$$\begin{array}{lll} p_1 \leftarrow p_2 \wedge \neg p_3 & p_2 \leftarrow p_4 \vee p_5 & \neg p_3 \leftarrow \neg p_6 \vee p_7 \vee p_8 \\ p_7 \leftarrow \neg p_7 & \neg p_7 \leftarrow p_7 & p_8 \leftarrow p_8 \\ p_4 \leftarrow \bigwedge \emptyset & p_5 \leftarrow \bigvee \emptyset & \neg p_6 \leftarrow \bigwedge \emptyset \end{array}$$

Suppose that we want to try and generate p_1. We will have to generate a set of literals that includes $\{p_1\}$. In order to generate p_1, it is necessary to generate both p_2 and $\neg p_3$, hence we will have to generate a set of literals that includes $\{p_1, p_2, \neg p_3\}$. In order to generate p_2, it is necessary to generate either p_4 or p_5, and in order to generate $\neg p_3$, it is necessary to generate either $\neg p_6$ or p_7 or p_8, hence we will have to generate a set of literals that includes one of the following sets.

$$\begin{array}{lll} \{p_1, p_2, \neg p_3, p_4, \neg p_6\} & \{p_1, p_2, \neg p_3, p_4, p_7\} & \{p_1, p_2, \neg p_3, p_4, p_8\} \\ \{p_1, p_2, \neg p_3, p_5, \neg p_6\} & \{p_1, p_2, \neg p_3, p_5, p_7\} & \{p_1, p_2, \neg p_3, p_5, p_8\} \end{array}$$

Observe how at every step, we have taken the union of a set of literals with one of a new set of literals. When it comes to generating p_4, p_5 and $\neg p_6$, all members of $\emptyset$ should be added to any set of literals that contains p_4 or $\neg p_6$, and some member of $\emptyset$ should be added to any set of literals that contains p_5. Finally, we come to the conclusion that we will have to generate one of the following sets.

$$\{p_1, p_2, \neg p_3, p_4, \neg p_6\} \qquad \{p_1, p_2, \neg p_3, p_4, p_7, \neg p_7\} \qquad \{p_1, p_2, \neg p_3, p_4, p_8\}$$

Note how $\bigwedge \emptyset$ "inhibits" the request for the generation of an additional set of literals, whereas $\bigvee \emptyset$ "cancels" the previous requests. Both literals p_7 and $\neg p_7$ keep requesting that the other be generated. Also, p_8 keeps requesting that it generates itself, hence successfully generating p_1 will turn out to be equivalent to successfully generating the members of $\{p_1, p_2, \neg p_3, p_4, \neg p_6\}$. What we have described is the construction of a kind of tableau [15] for p_1, emphasizing the convenient role played by $\bigwedge \emptyset$ and $\bigvee \emptyset$ in the construction. The construction is more generally captured by the following definition, where E can be seen as a set of 'extra assumptions.'

Definition 7. Let a logic program $\mathcal{P} = \left((\varphi_p^+, \varphi_p^-)\right)_{p \in \mathrm{Prd}(\mathcal{V})}$ and a set E of closed literals be given. Let f be the (unique) function from $\mathfrak{L}_{\omega_1\omega}(\mathcal{V} \cup \{=\})$ into the set of sets of literals which satisfies the following conditions.

- For all subsets X of $\mathfrak{L}_{\omega_1\omega}(\mathcal{V} \cup \{=\})$,
  - $f(\bigvee X)$ equals $\bigcup\{f(\varphi) \mid \varphi \in X\}$;

– $f(\bigwedge X)$ is the set of all sets of the form

$$\bigcup\{Y_\varphi \mid \varphi \in X,\ Y_\varphi \in f(\varphi)\}.$$

- For all $\varphi \in \mathcal{L}_{\omega_1\omega}(\mathcal{V} \cup \{=\})$ and variables x with $x \in \mathrm{fv}(\varphi)$,
  - $f(\exists x\varphi)$ equals $\bigcup\{f(\varphi[t/x]) \mid \text{closed term } t\}$;
  - $f(\forall x\varphi)$ is the set of all sets of the form

$$\bigcup\{Y_t \mid \text{closed term } t,\ Y_t \in f(\varphi[t/x])\}.$$

- For all $n \in \mathbb{N}$, $p \in \mathrm{Prd}(\mathcal{V}, n)$ and terms $t_1, \ldots, t_n$,
  - $f(p(t_1, \ldots, t_n))$ equals $\{\{p(t_1, \ldots, t_n)\}\}$ if $p(t_1, \ldots, t_n) \in E$, and

$$\{\{p(t_1, \ldots, t_n)\} \cup X \mid X \in f(\varphi_p^+[t_1/v1, \ldots, t_n/v_n])\}$$

    otherwise;
  - $f(\neg p(t_1, \ldots, t_n))$ equals $\{\{\neg p(t_1, \ldots, t_n)\}\}$ if $\neg p(t_1, \ldots, t_n) \in E$, and

$$\{\{\neg p(t_1, \ldots, t_n)\} \cup X \mid X \in f(\varphi_p^-[t_1/v1, \ldots, t_n/v_n])\}$$

    otherwise.

- For all terms t, $f(t = t) = \{\emptyset\}$ and $f(t \neq t) = \emptyset$.
- For all distinct terms t_1 and t_2, $f(t_1 = t_2) = \emptyset$ and $f(t_1 \neq t_2) = \{\emptyset\}$.

For all closed members φ of $\mathcal{L}_{\omega_1\omega}(\mathcal{V} \cup \{=\})$, $f(\varphi)$ is a set of sets of closed literals; we denote it by $\mathcal{P}[E, \varphi]$, and also more simply by $\mathcal{P}[\varphi]$ in case $E = \emptyset$.

Intuitively, if φ is closed then the members of $\mathcal{P}[E, \varphi]$ can be seen as the set of all closed literals that occur in one of the possible "constructive proofs" of φ from E, applying backward chaining to $\mathcal{P}$, starting from φ, and provided that the process does not 'loop.' We can now define the set of closed literals generated by $\mathcal{P}$. For another example of the use of inductive definitions to capture the meaning of logic programs, see [6].

Definition 8. Let a logic program $\mathcal{P} = \left((\varphi_p^+, \varphi_p^-)\right)_{p \in \mathrm{Prd}(\mathcal{V})}$ be given.

We inductively define a family $([\mathcal{P}]^\alpha)_{\alpha \in \mathrm{Ord}}$ of sets of literals as follows. Let an ordinal α be given, and assume that $[\mathcal{P}]^\beta$ has been defined for all $\beta < \alpha$. Then $[\mathcal{P}]^\alpha$ is the set of all literals φ for which there exists $n \in \mathbb{N}$, $p \in \mathrm{Prd}(\mathcal{V}, n)$, and closed terms $t_1, \ldots, t_n$ such that one of the following conditions holds:

- φ is $p(t_1, \ldots, t_n)$ and some member of $\mathcal{P}[\varphi_p^+[t_1/v_1, \ldots, t_n/v_n]]$ is included in $\bigcup_{\beta<\alpha}[\mathcal{P}]^\beta$;
- φ is $\neg p(t_1, \ldots, t_n)$ and some member of $\mathcal{P}[\varphi_p^-[t_1/v_1, \ldots, t_n/v_n]]$ is included in $\bigcup_{\beta<\alpha}[\mathcal{P}]^\beta$.

We set $[\mathcal{P}] = \bigcup_{\alpha\in\mathrm{Ord}}[\mathcal{P}]^\alpha$.

Example 9. If $\mathcal{P}$ is the logic program of Example 2 then for all $n \in \mathbb{N}$,

$$[\mathcal{P}]^n = \bigcup_{1\leq i\leq 2} \Big(\{p_i(\overline{2m}) \mid m \leq n\} \cup \{\neg p_i(\overline{2m+1}) \mid m \leq n\}\Big) \cup \{p_3(\overline{2m}) \mid m \leq \frac{n}{2}\} \cup \{\neg p_3(\overline{2m+1}) \mid m \leq \frac{n-1}{2}\} \cup \{q_1\}.$$

Property 10. *Given a logic program $\mathcal{P}$,* $\mathrm{CLF}(\mathcal{P}) \vDash_{\mathcal{W}} [\mathcal{P}]$.

By Property 10, if a logic program $\mathcal{P}$ is globally consistent in $\mathcal{W}$ then $[\mathcal{P}]$ is consistent. It is immediately verified that if a logic program $\mathcal{P}$ is locally consistent in $\mathcal{W}$, then $[\mathcal{P}]$ is also consistent, yielding Property 11.

Property 11. *For all logic programs $\mathcal{P}$, if $\mathcal{P}$ is either locally or globally consistent in $\mathcal{W}$ then $[\mathcal{P}]$ is consistent.*

It is worth observing that Definition 8 can be rephrased in terms of a forcing relation. Definition 12 defines this relation, and Property 13 uses it to provide an alternative to Definition 8.

Definition 12. Let S be a set of closed literals.

For all sentences φ, we inductively define the notion S *forces* φ *in* $\mathcal{W}$, denoted $S \Vdash_{\mathcal{W}} \varphi$, as follows.

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

Given a set T of sentences, we say that S *forces* T *in* $\mathcal{W}$, denoted $S \Vdash_{\mathcal{W}} T$, iff S forces all members of T in $\mathcal{W}$.

Property 13. *For all logic programs* $\mathcal{P} = \left((\varphi_p^+, \varphi_p^-)\right)_{p \in \mathrm{Prd}(\mathcal{V})}$ *and ordinals* α, $[\mathcal{P}]^\alpha$ *is the set of all literals* φ *for which there exists* $n \in \mathbb{N}$, $p \in \mathrm{Prd}(\mathcal{V}, n)$, *and closed terms* $t_1, \ldots, t_n$ *such that one of the following conditions holds:*

- φ *is* $p(t_1, \ldots, t_n)$ *and* $\bigcup_{\beta<\alpha}[\mathcal{P}]^\beta \Vdash_{\mathcal{W}} \varphi_p^+[t_1/v_1, \ldots, t_n/v_n]$;
- φ *is* $\neg p(t_1, \ldots, t_n)$ *and* $\bigcup_{\beta<\alpha}[\mathcal{P}]^\beta \Vdash_{\mathcal{W}} \varphi_p^-[t_1/v_1, \ldots, t_n/v_n]$.

6.2 The stratified approach

Another way of generating literals using the set of rules of a logic program can be described as follows.

- At the beginning, nothing has been generated.

  In the body of all closed instances of the positive rules, *i.e.*, the closed instances of the rules of the form $p(v_1, \ldots, v_n) \leftarrow \varphi_p^+$, $n \in \mathbb{N}$, $p \in \mathrm{Prd}(\mathcal{V}, n)$, take the pessimistic view that all negated atoms are false. This will allow one to generate a limited set S_0^+ of closed atoms.

  In the body of all closed instances of the negative rules, *i.e.*, the closed instances of the rules of the form $\neg p(v_1, \ldots, v_n) \leftarrow \varphi_p^-$, $n \in \mathbb{N}$, $p \in \mathrm{Prd}(\mathcal{V}, n)$, take the pessimistic view that all nonnegated atoms are false. This will allow one to generate a limited set S_0^- of closed negated atoms.

- We now know that all literals in $S_0^+ \cup S_0^-$ can be generated.

  In the body of all closed instances of the positive rules, take the pessimistic view that all negated atoms except those in S_0^- are false. This will allow one to generate a limited set S_1^+ of closed atoms that contains S_0^+.

  In the body of all closed instances of the negative rules, take the pessimistic view that all nonnegated atoms except those in S_0^+ are false. This will allow one to generate a limited set S_1^- of closed negated atoms that contains S_0^-.

- Etc.

The construction can be iterated transfinitely. Eventually, a set of closed literals of the form $\bigcup_{\alpha \in \mathrm{Ord}}(E_\alpha^+ \cup E_\alpha^-)$ will be generated.

The construction bears a clear resemblance with the various notions of stratification [12, 13]. In order to formalize it, we need a preliminary definition.

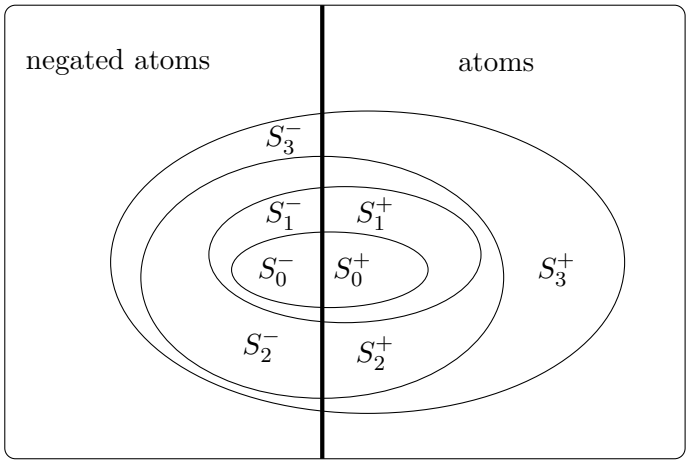

Figure 2: The literals generated by a logic program $\mathcal{P}$

Definition 14. Let an ordinal α, a set E of closed literals, and a formula φ over $\mathcal{V} \cup \{=\}$ be given.

We denote by $\oplus_E \varphi$ the formula over $\mathcal{V} \cup \{=\}$ obtained from φ by replacing all occurrences of negated atoms, say $\neg p(t_1, \ldots, t_n)$, with[2]

$$\bigvee \{ \bigwedge_{1 \leq i \leq n} t_i = t'_i \mid \neg p(t'_1, \ldots, t'_n) \in E \}.$$

We denote by $\ominus_E \varphi$ the formula over $\mathcal{V} \cup \{=\}$ obtained from φ by replacing all occurrences of nonnegated atoms, say $p(t_1, \ldots, t_n)$, with

$$\bigvee \{ \bigwedge_{1 \leq i \leq n} t_i = t'_i \mid p(t'_1, \ldots, t'_n) \in E \}.$$

Note that for all formulas φ over $\mathcal{V} \cup \{=\}$,

- $\oplus_\emptyset \varphi$ is the formula over $\mathcal{V} \cup \{=\}$ obtained from φ by replacing all occurrences of negated atoms with $\bigvee \emptyset$;
- $\ominus_\emptyset \varphi$ is the formula over $\mathcal{V} \cup \{=\}$ obtained from φ by replacing all occurrences of nonnegated atoms with $\bigvee \emptyset$.

Example 15. Suppose that $\mathcal{P}$ is the logic program of Example 2 and

$$E = \{ \neg p_3(s(\overline{0})),\ \neg p_3(s(s(\overline{0}))),\ p_4(s(\overline{0})),\ \neg q_5 \}.$$

[2]In case $n = 0$, the replacing expression is $\bigvee \bigwedge \emptyset$—which is logically equivalent to $\bigwedge \emptyset$—if $\neg p \in E$, and $\bigvee \emptyset$ if $\neg p \notin E$.

- $\oplus_E \varphi_{p_3}^+$ is logically equivalent to

$$v_1 = \overline{0} \vee \exists v_0 \Big(v_1 = s(v_0) \wedge \big(v_0 = s(\overline{0}) \vee v_0 = s(s(\overline{0})) \big) \Big).$$

- $\ominus_E \varphi_{p_4}^+$ is logically equivalent to $s(s(v_1)) = s(\overline{0})$.
- $\oplus_E \varphi_{q_5}^+$ is logically equivalent to $\bigwedge \emptyset$.

The construction outlined at the beginning of the section can now be formalized.

Definition 16. Let a logic program $\mathcal{P} = \big((\varphi_p^+, \varphi_p^-)\big)_{p \in \mathrm{Prd}(\mathcal{V})}$ be given.

We inductively define a family $([\mathcal{P}]_\alpha^\gamma)_{\gamma,\alpha \in \mathrm{Ord}}$ of sets of closed literals as follows. Let ordinals α and γ be given, and assume that $[\mathcal{P}]_\beta^\delta$ has been defined for all pairs of ordinals $(\beta, \delta) < (\alpha, \gamma)$.[3] Then $[\mathcal{P}]_\alpha^\gamma$ is defined as the set of all closed literals φ for which there exists $n \in \mathbb{N}$, $p \in \mathrm{Prd}(\mathcal{V}, n)$, and closed terms $t_1, \ldots, t_n$ such that one of the following conditions holds:

- φ is $p(t_1, \ldots, t_n)$ and $\oplus_{\bigcup_{\beta<\alpha}[\mathcal{P}]_\beta} \varphi_p^+[t_1/v_1, \ldots, t_n/v_n]$ is a logical consequence in $\mathcal{W}$ of the set of nonnegated atoms in $\bigcup_{(\beta,\delta)<(\alpha,\gamma)}[\mathcal{P}]_\beta^\delta$;
- φ is $\neg p(t_1, \ldots, t_n)$ and $\ominus_{\bigcup_{\beta<\alpha}[\mathcal{P}]_\beta} \varphi_p^-[t_1/v_1, \ldots, t_n/v_n]$ is a logical consequence in $\mathcal{W}$ of the set of negated atoms in $\bigcup_{(\beta,\delta)<(\alpha,\gamma)}[\mathcal{P}]_\beta^\delta$.

For all ordinals α, we set $[\mathcal{P}]_\alpha = \bigcup_{\gamma \in \mathrm{Ord}} [\mathcal{P}]_\alpha^\gamma$.

Example 17. If $\mathcal{P}$ is the logic program of Example 2 then for all $n \in \mathbb{N}$,

$$[\mathcal{P}]_n = \big\{ p_i(\overline{2m}), \neg p_i(\overline{2m+1}) \mid m \in \mathbb{N}, 1 \leq i \leq 2 \big\} \cup \big\{ p_3(\overline{2m}) \mid m \leq \frac{n}{2} \big\} \cup \big\{ \neg p_3(\overline{2m+1}) \mid m \leq \frac{n-1}{2} \big\} \cup \{ q_1 \}.$$

6.3 Equivalence of both approaches

Proposition 18. *For all logic programs $\mathcal{P}$, $[\mathcal{P}] = \bigcup_{\alpha \in Ord} [\mathcal{P}]_\alpha$.*

Proof. Let a logic program $\mathcal{P} = \big((\varphi_p^+, \varphi_p^-)\big)_{p \in \mathrm{Prd}(\mathcal{V})}$ be given. Proof of the proposition is by induction. Let $n \in \mathbb{N}$, $p \in \mathrm{Prd}(\mathcal{V}, n)$, and closed terms $t_1, \ldots, t_n$ be given. Let ξ be either $p(t_1, \ldots, t_n)$ or $\neg p(t_1, \ldots, t_n)$. Let χ be $\varphi_p^+[t_1/v_1, \ldots, t_n/v_n]$ if ξ is equal to $p(t_1, \ldots, t_n)$, and $\varphi_p^-[t_1/v_1, \ldots, t_n/v_n]$ if ξ is equal to $\neg p(t_1, \ldots, t_n)$.

Let an ordinal α be given. Assume that for all $\gamma < \alpha$, $[\mathcal{P}]^\gamma \subseteq \bigcup_{\beta \in \mathrm{Ord}} [\mathcal{P}]_\beta$. Let a member X of $\mathcal{P}[\chi]$ be included in $\bigcup_{\gamma<\alpha} [\mathcal{P}]^\gamma$. By inductive hypothesis, let ordinal

[3] Here $<$ denotes the strict lexicographic ordering over Ord^2.

λ be such that $\bigcup_{\beta<\lambda}[\mathcal{P}]_\beta$ includes X. Using the definition of $\mathcal{P}[\chi]$, it is easy to verify that $X \vDash_{\mathcal{W}} \oplus_{\bigcup_{\beta<\lambda}[\mathcal{P}]_\beta}\chi$ if ξ is equal to $p(t_1,\ldots,t_n)$, and $X \vDash_{\mathcal{W}} \ominus_{\bigcup_{\beta<\lambda}[\mathcal{P}]_\beta}\chi$ if ξ is equal to $\neg p(t_1,\ldots,t_n)$. Hence $\xi \in [\mathcal{P}]_\lambda$. Hence $[\mathcal{P}]^\alpha \subseteq \bigcup_{\beta\in \mathrm{Ord}}[\mathcal{P}]_\beta$.

Conversely, let ordinals λ_1 and λ_2 be given, and assume that for all ordinals β and δ with $(\beta,\delta) < (\lambda_1,\lambda_2)$, $[\mathcal{P}]^\delta_\beta \subseteq [\mathcal{P}]$. Set $\psi = \oplus_{\bigcup_{\beta<\lambda_1}[\mathcal{P}]_\beta}\chi$ if ξ is equal to $p(t_1,\ldots,t_n)$, and $\psi = \ominus_{\bigcup_{\beta<\lambda_1}[\mathcal{P}]_\beta}\chi$ if ξ is equal to $\neg p(t_1,\ldots,t_n)$. Suppose that ψ is a logical consequence in $\mathcal{W}$ of the set of nonnegated atoms in $\bigcup_{(\beta,\delta)<(\lambda_1,\lambda_2)}[\mathcal{P}]^\delta_\beta$ if ξ is equal to $p(t_1,\ldots,t_n)$, and of the set of negated atoms in $\bigcup_{(\beta,\delta)<(\lambda_1,\lambda_2)}[\mathcal{P}]^\delta_\beta$ otherwise. It follows from the inductive hypothesis that ψ is a logical consequence in $\mathcal{W}$ of the set of nonnegated atoms in $[\mathcal{P}]$ if ξ is equal to $p(t_1,\ldots,t_n)$, and of the set of negated atoms in $[\mathcal{P}]$ otherwise. Using the inductive hypothesis again, it is then easy to verify that some member of $\mathcal{P}[\psi]$ is included in $[\mathcal{P}]$. Hence $\xi \in [\mathcal{P}]$. Hence $[\mathcal{P}]^{\lambda_2}_{\lambda_1} \subseteq [\mathcal{P}]$. □

7 Extensions

7.1 Unrestricted extensions

Consider a logic program $\mathcal{P}$. If we were working in the spirit of classical Logic programming, we would give various semantics for $\mathcal{P}$. A given literal would be true in all intended models of $\mathcal{P}$ w.r.t. one semantics, but true in some intended models and false in others w.r.t. another semantics. We will propose a unique semantics in another paper. But we can retrieve the counterpart to the major semantics of classical Logic programming by considering various transformations of $\mathcal{P}$. For instance, instead of proposing three different semantics for $\mathcal{P}$, we would describe three logic programs closely related to $\mathcal{P}$ (one of which could be $\mathcal{P}$ itself) that would be interpreted w.r.t. this unique semantics. All transformations of $\mathcal{P}$ will be of the form: choose a set E of closed literals, and

- assume that a member of E is true when it occurs in the body of a closed instance of one of $\mathcal{P}$'s rules;
- assume that a member ψ of E is false when $\sim\psi$ occurs in the body of a closed instance of one of $\mathcal{P}$'s rules.

This transformation is formalized in Definition 22, with the help of a notation given in Definition 19.

Definition 19. Let a formula φ over $\mathcal{V} \cup \{=\}$ and a set E of closed literals be given. We denote by $\circledcirc_E\varphi$ the formula over $\mathcal{V} \cup \{=\}$ obtained from φ by

- replacing all occurrences of nonnegated atoms, say $p(t_1, \ldots, t_n)$, with

$$\Big(p(t_1, \ldots, t_n) \vee \bigvee\{\textstyle\bigwedge_{1\leq i\leq n} t_i = t'_i \mid p(t'_1, \ldots, t'_n) \in E\}\Big) \wedge \bigvee\{\textstyle\bigwedge_{1\leq i\leq n} t_i = t'_i \mid \neg p(t'_1, \ldots, t'_n) \notin E\};$$

- replacing all occurrences of negated atoms, say $\neg p(t_1, \ldots, t_n)$, with

$$\Big(\neg p(t_1, \ldots, t_n) \vee \bigvee\{\textstyle\bigwedge_{1\leq i\leq n} t_i = t'_i \mid \neg p(t'_1, \ldots, t'_n) \in E\}\Big) \wedge \bigvee\{\textstyle\bigwedge_{1\leq i\leq n} t_i = t'_i \mid p(t'_1, \ldots, t'_n) \notin E\}.$$

Proposition 20. *Let E be a consistent set of closed literals. Then for all sentences φ over $\mathcal{V} \cup \{=\}$, $\circledcirc_E\varphi$ is logically equivalent in $\mathcal{W}$ to $\sim \circledcirc_E \sim\varphi$.*

Proof. This follows immediately from the fact that for all $n \in \mathbb{N}$, $p \in \mathrm{Prd}(\mathcal{V}, n)$ and terms $t_1, \ldots, t_n$,

- $\sim\bigvee\{\bigwedge_{1\leq i\leq n} t_i = t'_i \mid p(t'_1, \ldots, t'_n) \in E\}$ is logically equivalent in $\mathcal{W}$ to $\bigvee\{\bigwedge_{1\leq i\leq n} t_i = t'_i \mid p(t'_1, \ldots, t'_n) \notin E\}$;
- $\sim\bigvee\{\bigwedge_{1\leq i\leq n} t_i = t'_i \mid \neg p(t'_1, \ldots, t'_n) \in E\}$ is logically equivalent in $\mathcal{W}$ to $\bigvee\{\bigwedge_{1\leq i\leq n} t_i = t'_i \mid \neg p(t'_1, \ldots, t'_n) \notin E\}$.

□

Example 21. Suppose that $\mathcal{P}$ is the logic program of Example 2 and

$$E = \{\neg p_3(s(\overline{0})),\ \neg p_3(s(s(\overline{0}))),\ p_3(s(s(s(\overline{0})))),\ p_4(s(\overline{0})),\ q_4,\ q_5\}.$$

- $\circledcirc_E\varphi^+_{p_3}$ is logically equivalent to

$$v_1 = \overline{0} \vee \exists v_0\Big(v_1 = s(v_0) \wedge (\neg p_3(v_0) \vee v_0 = s(\overline{0}) \vee v_0 = s(s(\overline{0}))) \wedge \bigvee\{v_0 = \overline{n} \mid n \in \mathbb{N} \setminus \{3\}\}\Big).$$

- $\circledcirc_E\varphi^+_{p_4}$ is logically equivalent to $p_4(s(s(v_1))) \vee s(s(v_1)) = s(\overline{0})$.
- $\circledcirc_E\varphi^-_{p_4}$ is logically equivalent to

$$\neg p_4(s(s(v_1))) \wedge \bigvee\{s(s(v_1)) = \overline{n} \mid n \in \mathbb{N} \setminus \{1\}\}.$$

- $\circledcirc_E \varphi^+_{q_2}$ is logically equivalent to q_3.
- $\circledcirc_E \varphi^+_{q_4}$ is logically equivalent to $\bigwedge \emptyset$.
- $\circledcirc_E \varphi^+_{q_5}$ is logically equivalent to $\bigvee \emptyset$.

Definition 22. Given a logic program $\mathcal{P} = \left((\varphi^+_p, \varphi^-_p)\right)_{p \in \mathrm{Prd}(\mathcal{V})}$ and a set E of closed literals, we denote by $\mathcal{P} + E$ the sequence $\left((\circledcirc_E \varphi^+_p, \circledcirc_E \varphi^-_p)\right)_{p \in \mathrm{Prd}(\mathcal{V})}$.

The key property of the class of logic programs that are locally consistency in $\mathcal{W}$ is given by Corollary 27—an immediate consequence of Lemma 26, that itself uses Lemma 25. It implies in particular that if we start with a symmetric logic program $\mathcal{P}$ and apply to $\mathcal{P}$ one of the transformations that correspond to the well-founded or stable model semantics, then we obtain a logic program that is locally consistent in $\mathcal{W}$, hence generates a consistent set of closed literals. Before we state and prove the lemmas needed to proved this claim, let us note that the empty extension yields, as can be expected, a "neutral" transformation of a logic program.

Property 23. *For all logic programs $\mathcal{P}$ and sentences φ over $\mathcal{V} \cup \{=\}$, $\mathcal{P}[\varphi]$ is logically equivalent to $(\mathcal{P} + \emptyset)[\varphi]$.*

Corollary 24. *For all logic programs $\mathcal{P}$ and ordinals α, $[\mathcal{P} + \emptyset]_\alpha = [\mathcal{P}]_\alpha$.*

Lemma 25. *For all consistent sets E of closed literals and for all closed literals φ, $\circledcirc_E \varphi \wedge \circledcirc_E \neg\varphi$ is unsatisfiable in $\mathcal{W}$ and $\circledcirc_E \varphi \vee \circledcirc_E \neg\varphi$ is valid in $\mathcal{W}$.*

Proof. Let a consistent set E of closed literals and a closed literal φ be given. If neither φ nor $\neg\varphi$ belongs to E then $\circledcirc_E \varphi$ is logically equivalent in $\mathcal{W}$ to φ and $\circledcirc_E \neg\varphi$ is logically equivalent in $\mathcal{W}$ to $\neg\varphi$. If φ belongs to E then $\circledcirc_E \varphi$ is logically equivalent in $\mathcal{W}$ to $\bigwedge \emptyset$ and $\circledcirc_E \neg\varphi$ is logically equivalent in $\mathcal{W}$ to $\bigvee \emptyset$. If $\neg\varphi$ belongs to E then $\circledcirc_E \varphi$ is logically equivalent in $\mathcal{W}$ to $\bigvee \emptyset$ and $\circledcirc_E \neg\varphi$ is logically equivalent in $\mathcal{W}$ to $\bigwedge \emptyset$. The lemma follows immediately. □

Lemma 26. *For all sentences φ, ψ over $\mathcal{V} \cup \{=\}$ and consistent sets E of closed literals, if $\varphi \wedge \psi$ is unsatisfiable in $\mathcal{W}$ then $\circledcirc_E \varphi \wedge \circledcirc_E \psi$ is unsatisfiable in $\mathcal{W}$.*

Proof. Let φ and ψ be two sentences over $\mathcal{V} \cup \{=\}$ such that $\varphi \wedge \psi$ is unsatisfiable in $\mathcal{W}$. Let E be a consistent set of closed literals. Let $\widehat{\varphi}$ be a disjunctive normal form equivalent of the formula over $\mathcal{V} \cup \{=\}$ obtained from φ by successively replacing all subformulas of the form $\exists x \xi$ with $\bigvee\{\xi[t/x] \mid \text{closed term } t\}$ and all subformulas of the form $\forall x \xi$ with $\bigwedge\{\xi[t/x] \mid \text{closed term } t\}$. Let $\widehat{\psi}$ be a conjunctive normal form equivalent of the formula over $\mathcal{V} \cup \{=\}$ obtained from ψ by successively replacing

all subformulas of the form $\exists x\xi$ with $\bigvee\{\xi[t/x] \mid \text{closed term } t\}$ and all subformulas of the form $\forall x\xi$ with $\bigwedge\{\xi[t/x] \mid \text{closed term } t\}$. Write $\widehat{\varphi}$ as $\bigvee_{i\in\mathbb{N}}\bigwedge X_i$, and write $\widehat{\psi}$ as $\bigwedge_{i\in\mathbb{N}}\bigvee Y_i$; so for all $i \in \mathbb{N}$, X_i and Y_i are (possibly empty) sets of closed atoms, negated closed atoms, closed equalities, or closed inequalities. It is easy to verify that $\bigvee_{i\in\mathbb{N}}\bigwedge\{\circledcirc_E\xi \mid \xi \in X_i\}$ is logically equivalent in $\mathcal{W}$ to $\circledcirc_E\varphi$, and $\bigwedge_{i\in\mathbb{N}}\bigvee\{\circledcirc_E\xi \mid \xi \in Y_i\}$ is logically equivalent in $\mathcal{W}$ to $\circledcirc_E\psi$. For all $i \in \mathbb{N}$, if $\bigwedge X_i$ is inconsistent in $\mathcal{W}$ then either X_i contains an equality or an inequality that is invalid in $\mathcal{W}$, or X_i contains both a closed atom and its negation, and it follows from Lemma 25 that $\bigwedge\{\circledcirc_E\xi \mid \xi \in X_i\}$ is unsatisfiable in $\mathcal{W}$. For all $i \in \mathbb{N}$, if $\bigvee Y_i$ is valid in $\mathcal{W}$ then either Y_i contains an equality or an inequality that is valid in $\mathcal{W}$, or X_i contains both a closed atom and its negation, and it follows from Lemma 25 that $\bigvee\{\circledcirc_E\xi \mid \xi \in Y_i\}$ is valid in $\mathcal{W}$. Let i and j in $\mathbb{N}$ be given, and assume that neither $\bigwedge X_i$ is unsatisfiable in $\mathcal{W}$ nor $\bigvee Y_j$ is valid in $\mathcal{W}$, and $\bigwedge X_i$ logically implies $\bigvee Y_j$ in $\mathcal{W}$. Then there exists a set F of equalities and inequalities that are valid in $\mathcal{W}$, such that $X_i \subseteq Y_j \cup F$. Obviously, $\{\circledcirc_E\varphi \mid \varphi \in X_i\} \subseteq \{\circledcirc_E\varphi \mid \varphi \in Y_j\} \cup F$, which shows that $\bigwedge\{\circledcirc_E\varphi \mid \varphi \in X_i\}$ logically implies $\bigvee\{\circledcirc_E\varphi \mid \varphi \in Y_j\}$ in $\mathcal{W}$. Hence $\bigvee_{i\in\mathbb{N}}\bigwedge\{\circledcirc_E\varphi \mid \varphi \in X_i\}$ logically implies $\bigwedge_{i\in\mathbb{N}}\bigvee\{\circledcirc_E\varphi \mid \varphi \in Y_j\}$ in $\mathcal{W}$, completing the proof the lemma. □

Corollary 27. *Let a logic program $\mathcal{P}$ be locally consistent in $\mathcal{W}$. For all sets E of closed literals, if E is consistent then $\mathcal{P} + E$ is locally consistent in $\mathcal{W}$.*

Given a logic program $\mathcal{P}$ and a set E of closed literals whose union with $[\mathcal{P}]$ is consistent, $\mathcal{P}+E$ can be seen as an extension of $\mathcal{P}$, in the sense that $[\mathcal{P}]$ is a subset of $[\mathcal{P} + E]$. Corollary 29—an immediate consequence of Property 28—states that $\mathcal{P} + E$ is an extension of $\mathcal{P}$ in an even stronger sense.

Property 28. *Let a logic program $\mathcal{P}$, a set E of closed literals, and a sentence φ over $\mathcal{V} \cup \{=\}$ be given.*

- *Let a member X of $\mathcal{P}[E, \varphi]$ be such that $X \cup E$ is consistent. Let Y be a set obtained from X by deleting some members of E. Then Y belongs to $(\mathcal{P} + E)[\varphi]$.*
- *Let a member X of $(\mathcal{P} + E)[\varphi]$ be given. Then there exists a member Y of $\mathcal{P}[E, \varphi]$ such that X is obtained from Y by deleting some members of E.*

Corollary 29. *Let a logic program $\mathcal{P}$ and a set E of closed literals be such that $[\mathcal{P}] \cup E$ is consistent. Then for all ordinals α, $[\mathcal{P}]_\alpha \subseteq [\mathcal{P} + E]_\alpha$.*

7.2 Supporting extensions

Corollary 29 is nice and good, but it does not imply that given a logic program $\mathcal{P}$ and a set E of closed literals whose union with $[\mathcal{P}]$ is consistent, E is included in $[\mathcal{P}+E]$—an additional property we would clearly expect for $\mathcal{P}+E$ to fully qualify as an extension of $\mathcal{P}$. The reason is that Definition 22 transforms $\mathcal{P}$ by looking at the occurrences of the members of E in the (closed instances of the) bodies of $\mathcal{P}$'s rules, with no concern for the heads of the rules of $\mathcal{P}$ some of whose closed instances belong to E. For instance, if $\mathcal{P}$ and E are defined as in Example 21 then $p_4(s(\overline{0}))$ is a member of E that does not belong to $[\mathcal{P}+E]$. We need to impose some condition on E in order that E be guaranteed to be included in $[\mathcal{P}+E]$. In this section and the next, we look at two possible conditions, one being a particular case of the other. The more general condition will be related to the well-founded semantics. In Section 8.3, we will look at a third condition, related to the stable model sematics. The more specific condition studied in this section is formalized in the next definition. The second clause in Definition 30 expresses that E is sufficiently rich to "generate itself" using $\mathcal{P}$. Intuitively, for all $\varphi \in E$, there exists a "constructive proof" of φ from E using $\mathcal{P}$ such that only members of E occur in this proof.

Definition 30. Given a logic program $\mathcal{P}$, a *supporting extension for* $\mathcal{P}$ is defined as any set E of closed literals such that:

- $[\mathcal{P}] \cup E$ is consistent;
- for all $\varphi \in E$, E includes some member of $\mathcal{P}[\varphi]$.

Example 31. Suppose that $\mathcal{P}$ is the logic program of Example 2. Then the supporting extensions for $\mathcal{P}$ which are disjoint from $[\mathcal{P}]$ are the consistent unions of

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

So with $\mathcal{P}$ as defined in Example 2, a particular supporting extension for $\mathcal{P}$ is

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

It will allow one to transform $\mathcal{P}$ into a 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.

In Example 31, we have not considered the supporting extensions for $\mathcal{P}$ that contain some members of $[\mathcal{P}]$. The next property gives an example of some of the subsets of $[\mathcal{P}]$ that could be added to any supporting extension for a logic program $\mathcal{P}$.

Property 32. *For all logic programs $\mathcal{P}$ and ordinals α, if $[\mathcal{P}]$ is consistent then $[\mathcal{P}]^\alpha$ and $[\mathcal{P}]_\alpha$ are supporting extensions for $\mathcal{P}$.*

Example 31 also illustrates the general fact expressed in the next property.

Property 33. *Let $\mathcal{P}$ be a logic program. For any set X of supporting extensions for $\mathcal{P}$, $\bigcup X$ is a supporting extension for $\mathcal{P}$ iff $[\mathcal{P}] \cup \bigcup X$ is consistent.*

A supporting extension E for a logic program $\mathcal{P}$ addresses the issue we raised at the beginning of the section even better than by just making E a subset of $[\mathcal{P} + E]$, as expressed in the next property.

Property 34. *For all logic programs $\mathcal{P}$ and sets E of closed literals, if E is a supporting extension for $\mathcal{P}$ then $E \subseteq [\mathcal{P} + E]_0$.*

Given a logic program $\mathcal{P}$ and a supporting extension E for $\mathcal{P}$, $[\mathcal{P} + E]$ usually encompasses more literals than those in $[\mathcal{P}] \cup E$. A nice property is that not only is $E \cup [\mathcal{P}]$ a supporting extensions for $\mathcal{P}$—as we already know from Properties 32 and 33: $[\mathcal{P} + E]$ is also a supporting extension for $\mathcal{P}$.

Proposition 35. *Let $\mathcal{P}$ be a logic program that is locally consistent in $\mathcal{W}$, and let E be a supporting extension for $\mathcal{P}$. Then $[\mathcal{P} + E]$ is a supporting extension for $\mathcal{P}$.*

Proof. By Property 11 and Corollary 27, $[\mathcal{P} + E]$ is consistent. By Corollary 29, $[\mathcal{P}] \subseteq [\mathcal{P} + E]$. Let a member φ of $[\mathcal{P} + E]$ be given. Choose $X \in (\mathcal{P} + E)[\varphi]$ such that $X \subseteq [\mathcal{P} + E]$. By Property 28, chose a member Y of $\mathcal{P}[E, \varphi]$ such that X is is obtained from Y by deleting some members of E. For all $\psi \in E$ that occur in Y, choose a member Y_ψ of $\mathcal{P}[\psi]$ such that $Y_\psi \subseteq E$. Define Y' to be the union of Y with all sets of the form Y_ψ, where ψ is a member of E that occurs in Y. It is easy to verify that $Y' \in \mathcal{P}[\varphi]$. Clearly, $[\mathcal{P} + E] \cup E$, which is equal to $[\mathcal{P} + E]$ by Property 34, includes Y'. Hence $[\mathcal{P} + E]$ is a supporting extension for $\mathcal{P}$. □

7.3 Supporting chains

Let a logic program $\mathcal{P}$ be given. One way to define extensions E for $\mathcal{P}$ that have the property that $E \subseteq [\mathcal{P} + E]$ would be to iterate Definition 30:

- start with a supporting extension E_0 for $\mathcal{P}$;
- propose a supporting extension E_1 for $\mathcal{P} + E_0$;
- propose a supporting extension E_2 for $(\mathcal{P} + E_0) + E_1$;
- propose a supporting extension E_3 for $\big((\mathcal{P} + E_0) + E_1\big) + E_2$;
- Etc.

But having in mind Rondogiannis and Wadge's treatment of the well-founded semantics for classical logic programs, we investigate the properties of another construction, given in Definition 36, that can be described as follows:

- start with a set E_0 of closed literals such that $[\mathcal{P}] \cup E_0$ is a supporting extension for $\mathcal{P}$;
- propose a set E_1 of closed literals such that $[\mathcal{P} + E_0] \cup E_1$ is a supporting extension for $\mathcal{P}$;
- propose a set E_2 of closed literals such that $[\mathcal{P}+(E_0\cup E_1)]\cup E_2$ is a supporting extension for $\mathcal{P}$;
- propose a set E_3 of closed literals such that $[\mathcal{P} + (E_0 \cup E_1 \cup E_2)] \cup E_3$ is a supporting extension for $\mathcal{P}$;
- Etc.

Definition 36. Let a logic program $\mathcal{P}$ be given.

Given an ordinal α, a *supporting α-chain for $\mathcal{P}$* is a sequence $(E_\beta)_{\beta<\alpha}$ of sets of closed literals such that $\bigcup_{\beta<\alpha} E_\beta$ is consistent and for all ordinals β strictly smaller than α, $[\mathcal{P} + \bigcup_{\gamma<\beta} E_\gamma] \cup E_\beta$ is a supporting extension for $\mathcal{P}$.

A *supporting chain for $\mathcal{P}$* is a sequence $(E_\alpha)_{\alpha\in\mathrm{Ord}}$ of sets of closed literals such that for all ordinals α, $(E_\beta)_{\beta<\alpha}$ is a supporting α-chain for $\mathcal{P}$.

Let a logic program $\mathcal{P}$ and a supporting chain $(E_\alpha)_{\alpha\in\mathrm{Ord}}$ for $\mathcal{P}$ be given. We obviously have in mind the transformation of $\mathcal{P}$ to $\mathcal{P} + \bigcup_{\alpha\in\mathrm{Ord}} E_\alpha$. The first thing we should verify is that $\bigcup_{\alpha\in\mathrm{Ord}} E_\alpha$ is contained in $[\mathcal{P} + \bigcup_{\alpha\in\mathrm{Ord}} E_\alpha]$. This is indeed the case, as expressed in Corollary 42, immediately derived from Proposition 41. A number of preliminary results facilitate the proof of Proposition 41.

Lemma 37. *Let $\mathcal{P}$ be a logic program that is locally consistent in $\mathcal{W}$. Let E and F be two sets of closed literals such that $E \subseteq [\mathcal{P}+E]$ and $[\mathcal{P}+E]\cup F$ is consistent. Then $[\mathcal{P}+[\mathcal{P}+E]\cup F]$ is a subset of $[\mathcal{P}+E\cup F]$.*

Proof. For all closed literals φ and members X of $(\mathcal{P}+[\mathcal{P}+E]\cup F)[\varphi]$, we define a member $g(\varphi,X)$ of $(\mathcal{P}+E\cup F)[\varphi]$ as follows. Let a closed literal φ and a member X of $(\mathcal{P}+[\mathcal{P}+E]\cup F)[\varphi]$ be given. By Property 28, chose a member Y of $\mathcal{P}[[\mathcal{P}+E]\cup F,\varphi]$ such that X is obtained from Y by deleting some members of $[\mathcal{P}+E]\cup F$. For all $\psi\in[\mathcal{P}+E]$ that occur in Y, choose a member X_ψ of $(\mathcal{P}+E)[\psi]$ such that $X_\psi\subseteq[\mathcal{P}+E]$. Define $g(\varphi,X)$ to be the union of Y with all sets of the form X_ψ, where ψ is a member of $[\mathcal{P}+E]$ that occurs in Y. Since $E\subseteq[\mathcal{P}+E]$ and $[\mathcal{P}+E]\cup F$ is consistent, it is easily verified that $g(\varphi,X)$ belongs to $(\mathcal{P}+E\cup F)[\varphi]$. Also, the fact that $E\subseteq[\mathcal{P}+E]$ and $[\mathcal{P}+E]\cup F$ is consistent implies easily that $[\mathcal{P}+E]\subseteq[\mathcal{P}+E\cup F]$. Now let ordinal λ be such that $[\mathcal{P}+E]\subseteq[\mathcal{P}+E]^\lambda$. A simple inductive argument then allows one to show that for all closed literals φ, members X of $(\mathcal{P}+E\cup F)[\varphi]$ and ordinals α, if $X\subseteq\bigcup_{\beta<\alpha}[\mathcal{P}+[\mathcal{P}+E]\cup F]^\beta$ then $g(\varphi,X)$ is a subset of $\bigcup_{\beta<\alpha}[\mathcal{P}+E\cup F]^{\lambda+\beta}$. We conclude that $[\mathcal{P}+[\mathcal{P}+E]\cup F]$ is included in $[\mathcal{P}+E\cup F]$. □

Property 38. *Let a logic program $\mathcal{P}$ and two consistent sets E and F of closed literals be such that $E\subseteq F$. Let a sentence φ over $\mathcal{V}\cup\{=\}$ be given. For all members X of $(\mathcal{P}+E)[\varphi]$, if $X\cup F$ is consistent then $X\in(\mathcal{P}+F)[\varphi]$.*

Corollary 39. *Let a logic program $\mathcal{P}$, two sets E and F of closed literals with $E\subseteq F$, and an ordinal α be given.*

- *If $[\mathcal{P}+E]^\alpha\cup F$ is consistent then $[\mathcal{P}+E]^\alpha\subseteq[\mathcal{P}+F]^\alpha$.*
- *If $[\mathcal{P}+E]_\alpha\cup F$ is consistent then $[\mathcal{P}+E]_\alpha\subseteq[\mathcal{P}+F]_\alpha$.*

Corollary 40. *Let a logic program $\mathcal{P}$, an ordinal α, and a supporting α-chain $(E_\beta)_{\beta<\alpha}$ for $\mathcal{P}$ be given. Then for all $\beta<\alpha$, $[\mathcal{P}+\bigcup_{\gamma<\beta}E_\gamma]\subseteq[\mathcal{P}+\bigcup_{\gamma\leq\beta}E_\gamma]$.*

Proposition 41. *Let a logic program $\mathcal{P}$, an ordinal α, and a supporting α-chain $(E_\beta)_{\beta<\alpha}$ for $\mathcal{P}$ be given. Then the following holds.*

- *$\bigcup_{\beta<\alpha}E_\beta\subseteq[\mathcal{P}+\bigcup_{\beta<\alpha}E_\beta]$.*
- *For all ordinals β_1 and β_2 with $\beta_1\leq\beta_2\leq\alpha$, $[\mathcal{P}+\bigcup_{\gamma<\beta_1}E_\gamma]$ is a subset of $[\mathcal{P}+\bigcup_{\gamma<\beta_2}E_\gamma]$.*

Proof. Proof is by induction. Assume that for all $\beta<\alpha$, $\bigcup_{\gamma<\beta}E_\gamma$ is included in $[\mathcal{P}+\bigcup_{\gamma<\beta}E_\gamma]$ and for all ordinals β_1 and β_2 with $\beta_1\leq\beta_2\leq\beta$, $[\mathcal{P}+\bigcup_{\gamma<\beta_1}E_\gamma]$ is included in $[\mathcal{P}+\bigcup_{\gamma<\beta_2}E_\gamma]$.

Assume that α is a limit ordinal. It follows from the inductive hypothesis that for all ordinals δ and β with $\delta < \beta < \alpha$, $E_\delta \subseteq [\mathcal{P} + \bigcup_{\gamma<\beta} E_\gamma]$. Together with the inductive hypothesis again, this implies that for all ordinals β strictly smaller than α, $\bigcup_{\gamma<\alpha} E_\gamma \cup [\mathcal{P} + \bigcup_{\gamma<\beta} E_\gamma]$ is consistent. We then infer from Corollary 39 that for all $\beta < \alpha$, $[\mathcal{P} + \bigcup_{\gamma<\beta} E_\gamma] \subseteq [\mathcal{P} + \bigcup_{\gamma<\alpha} E_\gamma]$. Using the induction hypothesis again, it follows that $\bigcup_{\gamma<\alpha} E_\gamma$ is also included in $[\mathcal{P} + \bigcup_{\gamma<\alpha} E_\gamma]$.

Suppose that α is of the form $\delta+1$. By inductive hypothesis, $\bigcup_{\gamma<\delta} E_\gamma$ is a subset of $[\mathcal{P} + \bigcup_{\gamma<\delta} E_\gamma]$. Moreover, $[\mathcal{P} + \bigcup_{\gamma<\delta} E_\gamma] \cup E_\delta$, being a supporting extension for $\mathcal{P}$, is consistent, and it follows from Property 34 that $E_\delta \subseteq [\mathcal{P} + [\mathcal{P} + \bigcup_{\gamma<\delta} E_\gamma] \cup E_\delta]$. Lemma 37 then implies that E_δ is a subset of $[\mathcal{P} + \bigcup_{\gamma\leq\delta} E_\gamma]$. We conclude with the inductive hypothesis and Corollary 40. □

Corollary 42. *For all logic programs $\mathcal{P}$ and supporting chains $(E_\alpha)_{\alpha\in Ord}$ for $\mathcal{P}$, $\bigcup_{\alpha\in Ord} E_\alpha$ is included in $[\mathcal{P} + \bigcup_{\alpha\in Ord} E_\alpha]$.*

Corollary 43. *For all logic programs $\mathcal{P}$ and supporting chains $(E_\alpha)_{\alpha\in Ord}$ for $\mathcal{P}$,*

$$[\mathcal{P} + \bigcup_{\alpha\in Ord} E_\alpha] = \bigcup_{\alpha\in Ord} [\mathcal{P} + \bigcup_{\beta<\alpha} E_\beta].$$

Proposition 35 has a counterpart for supporting chains, expressed as Propositions 47 and 48. Their proofs are facilitated by a few of propositions that are interesting in their own right.

Proposition 44. *Let $\mathcal{P}$ be a logic program that is locally consistent in $\mathcal{W}$. Let an ordinal α and a supporting α-chain $(E_\beta)_{\beta<\alpha}$ for $\mathcal{P}$ be given. Then*

$$[\mathcal{P} + \bigcup_{\beta<\alpha} E_\alpha] = [\mathcal{P} + \bigcup_{\beta<\alpha} [\mathcal{P} + \bigcup_{\gamma\leq\beta} E_\gamma]].$$

Proof. It follows from Proposition 41 that $\bigcup_{\beta<\alpha} E_\beta \subseteq \bigcup_{\beta<\alpha} [\mathcal{P} + \bigcup_{\gamma\leq\beta} E_\gamma]$ and $\bigcup_{\beta<\alpha} [\mathcal{P} + \bigcup_{\gamma\leq\beta} E_\gamma]$ is a subset of $[\mathcal{P} + \bigcup_{\beta<\alpha} E_\beta]$. By Property 11 and Corollary 27, $[\mathcal{P} + \bigcup_{\beta<\alpha} E_\beta]$ is a consistent set. We can then derive from Corollary 39 that $[\mathcal{P} + \bigcup_{\beta<\alpha} E_\beta]$ is included in $[\mathcal{P} + \bigcup_{\beta<\alpha} [\mathcal{P} + \bigcup_{\gamma\leq\beta} E_\gamma]]$.

By Corollary 27, $[\mathcal{P} + \bigcup_{\gamma\leq\beta} E_\gamma]$ is consistent for all ordinals β strictly smaller than α. Together with Proposition 41, this implies that $\bigcup_{\beta<\alpha} [\mathcal{P} + \bigcup_{\gamma\leq\beta} E_\gamma]$ is consistent. Using two applications of Property 11 and Corollary 27, we then derive that $[\mathcal{P} + \bigcup_{\beta<\alpha} [\mathcal{P} + \bigcup_{\gamma\leq\beta} E_\gamma]]$ is consistent. Moreover, we have shown that the latter includes $[\mathcal{P} + \bigcup_{\beta<\alpha} E_\beta]$, so we infer from Corollary 39 that $[\mathcal{P} + \bigcup_{\beta<\alpha} [\mathcal{P} + \bigcup_{\gamma\leq\beta} E_\gamma]]$ is included in $[\mathcal{P} + [\mathcal{P} + \bigcup_{\beta<\alpha} E_\beta]]$. But Lemma 37 and Proposition 41 imply that $[\mathcal{P} + [\mathcal{P} + \bigcup_{\beta<\alpha} E_\beta]]$ is a subset of $[\mathcal{P} + \bigcup_{\beta<\alpha} E_\beta]$, completing the proof of the proposition. □

Proposition 45. *Let a logic program $\mathcal{P}$ be given. Let an ordinal α and a supporting $(\alpha+1)$-chain $(E_\beta)_{\beta\leq\alpha}$ for $\mathcal{P}$ be given. Then*

$$[\mathcal{P} + [\mathcal{P} + \bigcup_{\beta<\alpha} E_\beta] \cup E_\alpha] = [\mathcal{P} + \bigcup_{\beta\leq\alpha} E_\beta].$$

Proof. By Lemma 37, $[\mathcal{P} + [\mathcal{P} + \bigcup_{\beta<\alpha} E_\beta] \cup E_\alpha] \subseteq [\mathcal{P} + \bigcup_{\beta\leq\alpha} E_\beta]$. By Corollary 39 and Proposition 41, $[\mathcal{P} + \bigcup_{\beta\leq\alpha} E_\beta] \subseteq [\mathcal{P} + [\mathcal{P} + \bigcup_{\beta<\alpha} E_\beta] \cup E_\alpha]$. □

Corollary 46. *Let $\mathcal{P}$ be a logic program that is locally consistent in $\mathcal{W}$. Let an ordinal α and a supporting α-chain $(E_\beta)_{\beta<\alpha}$ for $\mathcal{P}$ be given. For all ordinals $\beta \geq \alpha$, set $E_\beta = \emptyset$. Then $(E_\beta)_{\beta\in Ord}$ is a supporting chain for $\mathcal{P}$.*

Proposition 47. *Let $\mathcal{P}$ be a logic program that is locally consistent in $\mathcal{W}$. Let an ordinal α and a supporting α-chain $(E_\beta)_{\beta<\alpha}$ for $\mathcal{P}$ be given. Then $[\mathcal{P} + \bigcup_{\beta<\alpha} E_\beta]$ is a supporting extension for $\mathcal{P}$.*

Proof. This follows immediately from Propositions 35 and 44. □

Proposition 48. *Let a logic program $\mathcal{P}$ and a supporting chain $(E_\alpha)_{\alpha\in Ord}$ for $\mathcal{P}$ be given. Then $[\mathcal{P} + \bigcup_{\alpha\in Ord} E_\alpha]$ is a supporting extension for $\mathcal{P}$.*

Proof. This follows immediately from Property 33, Proposition 41, and Corollary 43. □

Now we provide a strong version of a counterpart to Property 33 for supporting chains. Note that thanks to Corollary 46, Proposition 49 considers supporting chains only (rather than supporting chains and supporting α-chains for arbitrary ordinals α) at no loss of generality.

Proposition 49. *Let $\mathcal{P}$ be a logic program that is locally consistent in $\mathcal{W}$. Let a set I and a set of supporting chains for $\mathcal{P}$ of the form $\{(E^\sigma_\gamma)_{\gamma\in Ord} \mid \sigma \in I\}$ be given. Then $(\bigcup_{\sigma\in I} E^\sigma_\alpha)_{\alpha\in Ord}$ is a supporting chain for $\mathcal{P}$ iff $\bigcup_{\sigma\in I}\bigcup_{\alpha\in Ord} E^\sigma_\alpha$ is consistent.*

Proof. Only one direction of the proposition requires a proof. Assume that the set $\bigcup_{\sigma\in I}\bigcup_{\alpha\in \mathrm{Ord}} E^\sigma_\alpha$ is consistent. For all $\alpha \in \mathrm{Ord}$, set $F_\alpha = \bigcup_{\sigma\in I} E^\sigma_\alpha$. Let ordinal α be given, and assume that $(F_\beta)_{\beta<\alpha}$ is a supporting α-chain for $\mathcal{P}$. By Proposition 47, $[\mathcal{P} + \bigcup_{\beta<\alpha} F_\beta]$ is a supporting extension for $\mathcal{P}$. Moreover, we infer from Corollary 39 that for all $\sigma \in I$, $[\mathcal{P} + \bigcup_{\beta<\alpha} F_\beta] \cup E^\sigma_\alpha$ contains a superset of E^σ_α that is a supporting extension for $\mathcal{P}$. If $[\mathcal{P} + \bigcup_{\beta<\alpha} F_\beta] \cup F_\alpha$ is consistent then we can conclude with Property 33 that $[\mathcal{P} + \bigcup_{\beta<\alpha} F_\beta] \cup F_\alpha$ is a supporting extension for $\mathcal{P}$. So let us show that $[\mathcal{P} + \bigcup_{\beta<\alpha} F_\beta] \cup F_\alpha$ is consistent. By Corollary 39 and Proposition 41, E^σ_α is

included in $[\mathcal{P} + \bigcup_{\gamma \leq \alpha} F_\gamma]$ for all members σ of I. We infer that F_α is a subset of $[\mathcal{P} + \bigcup_{\gamma \leq \alpha} F_\gamma]$. By Corollary 39 again, $[\mathcal{P} + \bigcup_{\beta < \alpha} F_\beta]$ is a subset of $[\mathcal{P} + \bigcup_{\gamma \leq \alpha} F_\gamma]$. Hence $[\mathcal{P} + \bigcup_{\beta < \alpha} F_\beta] \cup F_\alpha$ is consistent, as wanted. □

As an application of Proposition 49, we can take the view of classical Logic programming, be biased towards negated atoms, and get the following corollary, which shows that for all logic programs $\mathcal{P}$, there exists a $\subseteq$-maximal extension for $\mathcal{P}$ consisting of negated closed atoms only, that can be written as a supporting chain for $\mathcal{P}$.

Corollary 50. *For all logic programs $\mathcal{P}$ that are locally consistent in $\mathcal{W}$, there exists a $\subseteq$-maximal set E of negated closed atoms such that there exists a supporting chain $(E_\alpha)_{\alpha \in Ord}$ for $\mathcal{P}$ with $\bigcup_{\alpha \in Ord} E_\alpha = E$.*

Let a logic program $\mathcal{P}$ and a supporting chain $(E_\alpha)_{\alpha \in \mathrm{Ord}}$ for $\mathcal{P}$ be given. With $\mathcal{P} + \bigcup_{\alpha \in \mathrm{Ord}} E_\alpha$, we transform $\mathcal{P}$ by incorporating the members of $\bigcup_{\alpha \in \mathrm{Ord}} E_\alpha$ in the bodies of $\mathcal{P}$'s rules at once, irrespective of which sets E_β, $\beta \in \mathrm{Ord}$, a given member of $\bigcup_{\alpha \in \mathrm{Ord}} E_\alpha$ belongs to. We could try and state a counterpart to Property 34, and given a member φ of $\bigcup_{\alpha \in \mathrm{Ord}} E_\alpha$, see the relationship between the ordinals β such that $\varphi \in E_\beta$, and the ordinals β such that $\varphi \in [\mathcal{P} + \bigcup_{\alpha \in \mathrm{Ord}} E_\alpha]^\beta$ or $\varphi \in [\mathcal{P} + \bigcup_{\alpha \in \mathrm{Ord}} E_\alpha]_\beta$. But such a relationship would shed no light on Rondogiannis and Wadge's treatment of the well-founded semantics for classical logic programs. In the case where $\mathcal{P}$ is symmetric, the well-founded model of $\mathcal{P}$ can be obtained from $\mathcal{P}$ by transforming $\mathcal{P}$ into $\mathcal{P} + E$, where E is the set defined in Corollary 50. This relationship is welcome, but insufficient: we also want to establish a relationship between an ordinal β that, in Rondogiannis and Wadge's setting, is interpreted as a degree of infinitesimal truth value, and one of the uses of β in our setting. For this purpose, we need to define a particular supporting chain for $\mathcal{P}$, the union of whose members is E. Definition 51 gives the details.

Definition 51. Let a logic program $\mathcal{P}$ be given. Inductively define for all ordinals α two sets E^+_α and E^-_α of literals as follows. For all $\alpha \in \mathrm{Ord}$, define E^+_α as the set of atoms in $[\mathcal{P} + \bigcup_{\beta < \alpha} E^-_\beta]_0$ and not in $\bigcup_{\beta < \alpha} E^+_\beta$. For all $\alpha \in \mathrm{Ord}$, define E^-_α as the $\subseteq$-maximal set of negated atoms not in $\bigcup_{\beta < \alpha} E^-_\beta$ such that $\bigcup_{\beta < \alpha}(E^+_\beta \cup E^-_\beta) \cup E^-_\alpha$ is a supporting extension for $\mathcal{P}$. We call $\left((E^+_\alpha, E^-_\alpha)\right)_{\alpha \in \mathrm{Ord}}$ the *RW-sequence for* $\mathcal{P}$.[4]

Proposition 52 proves that Definition 51 is appropriate to turn the set E described in Corollary 50 into a supporting chain.

[4] "RW" in "RW-sequence" stands for "Rondogiannis and Wadge."

Proposition 52. *Let $\mathcal{P}$ be a logic program that is locally consistent in $\mathcal{W}$. Let $\left((E^+_\alpha, E^-_\alpha)\right)_{\alpha \in Ord}$ be the RW-sequence for $\mathcal{P}$. Then:*

- *$(E^-_\alpha)_{\alpha \in Ord}$ is a supporting chain for $\mathcal{P}$;*
- *$\bigcup_{\alpha \in Ord}(E^+_\alpha \cup E^-_\alpha) = [\mathcal{P} + \bigcup_{\alpha \in Ord} E^-_\alpha]$;*
- *$\bigcup_{\alpha \in Ord} E^-_\alpha$ is the $\subseteq$-maximal set of closed negated atoms such that there exists a supporting chain $(F_\alpha)_{\alpha \in Ord}$ for $\mathcal{P}$ with $\bigcup_{\alpha \in Ord} F_\alpha = \bigcup_{\alpha \in Ord} E^-_\alpha$.*

Proof. For all ordinals α, define F_α as the $\subseteq$-maximal set of negated closed atoms such that $[\mathcal{P} + \bigcup_{\beta<\alpha} F_\beta] \cup F_\alpha$ is a supporting extension for $\mathcal{P}$. It is easy to derive from Proposition 49 that $\bigcup_{\alpha \in \mathrm{Ord}} F_\alpha$ is the $\subseteq$-maximal set of closed negated atoms such that there exists a supporting chain $(G_\alpha)_{\alpha \in \mathrm{Ord}}$ for $\mathcal{P}$ with $\bigcup_{\alpha \in \mathrm{Ord}} G_\alpha$ being equal to $\bigcup_{\alpha \in \mathrm{Ord}} F_\alpha$.

We first show that for all ordinals α,

- $E^-_\alpha \subseteq F_\alpha$;
- $E^+_\alpha \subseteq [\mathcal{P} + \bigcup_{\beta \leq \alpha} F_\beta]$.

Proof is by induction. Let an ordinal α be given, and assume that for all ordinals β strictly smaller than α, $E^-_\beta \subseteq F_\beta$ and $E^+_\beta \subseteq [\mathcal{P} + \bigcup_{\gamma \leq \beta} F_\gamma]$. It follows from Corollary 39 and Proposition 41 that $\bigcup_{\beta<\alpha}(E^+_\beta \cup E^-_\beta) \subseteq [\mathcal{P} + \bigcup_{\beta<\alpha} F_\beta]$. So by the definition of E^-_α, $[\mathcal{P} + \bigcup_{\beta<\alpha} F_\beta] \cup E^-_\alpha$ contains a superset of E^-_α that is a supporting extension for $\mathcal{P}$. Hence by the definition of F_α, $E^-_\alpha \subseteq F_\alpha$. This together with Corollary 39 and the inductive hypothesis implies immediately that E^+_α is included in $[\mathcal{P} + \bigcup_{\beta \leq \alpha} F_\beta]$.

We now show that $(E^-_\alpha)_{\alpha \in \mathrm{Ord}}$ is a supporting chain for $\mathcal{P}$. Proof is by induction. Let an ordinal α be given, and assume that for all $\beta < \alpha$, $(E^-_\gamma)_{\gamma<\beta}$ is a supporting β-chain for $\mathcal{P}$ and $\bigcup_{\gamma<\beta} E^+_\gamma \subseteq [\mathcal{P} + \bigcup_{\gamma<\beta} E^-_\gamma]$. It follows easily from Corollary 39 and the inductive hypothesis that $\bigcup_{\beta<\alpha} E^+_\beta \subseteq [\mathcal{P} + \bigcup_{\beta<\alpha} E^-_\beta]$. If α is a limit ordinal then $(E^-_\beta)_{\beta<\alpha}$ is trivially a supporting α-chain for $\mathcal{P}$. Suppose that α is of the form $\delta + 1$. By Proposition 47, $[\mathcal{P} + \bigcup_{\beta<\delta} E^-_\beta]$ is a supporting extension for $\mathcal{P}$. Using the inductive hypothesis, Corollary 39, Proposition 41, and the fact that $\bigcup_{\beta<\delta} E^+_\beta \subseteq [\mathcal{P} + \bigcup_{\beta<\delta} E^-_\beta]$, we infer that $[\mathcal{P} + \bigcup_{\beta<\delta} E^-_\beta] \cup E^-_\delta$ contains a superset of E^-_δ that is a supporting extension for $\mathcal{P}$. Moreover, we have established that for all $\beta \leq \delta$, $E^-_\beta \subseteq F^-_\delta$, which together with Corollary 39, allows one to infer that $[\mathcal{P} + \bigcup_{\beta<\delta} E^-_\beta] \cup E^-_\delta$ is consistent. We conclude with Property 33 that $[\mathcal{P} + \bigcup_{\beta<\delta} E^-_\beta] \cup E^-_\delta$ is a supporting extension for $\mathcal{P}$. So we have shown that $(E^-_\beta)_{\beta<\alpha}$ is a supporting α-chain for $\mathcal{P}$.

Let an ordinal δ be such that $\bigcup_{\alpha\in\mathrm{Ord}} E^-_\alpha = \bigcup_{\alpha<\delta} E^-_\alpha$. It is then easy to verify by induction that for all ordinals β, the set of atoms in $[\mathcal{P}+\bigcup_{\alpha\in\mathrm{Ord}} E^-_\alpha]_\beta$ is included in $E^+_{\delta+\beta}$ and the set of negated atoms in $[\mathcal{P}+\bigcup_{\alpha\in\mathrm{Ord}} E^-_\alpha]_\beta$ is included in $E^-_{\delta+\beta}$. Hence $\bigcup_{\alpha\in\mathrm{Ord}}(E^+_\alpha \cup E^-_\alpha) = [\mathcal{P}+\bigcup_{\alpha\in\mathrm{Ord}} E^-_\alpha]$.

Let an ordinal α be given, and assume that for all $\beta<\alpha$, $F_\beta \subseteq \bigcup_{\gamma\in\mathrm{Ord}} E^-_\gamma$. To complete the proof of the proposition, it suffices to show that $F_\alpha \subseteq \bigcup_{\gamma\in\mathrm{Ord}} E^-_\gamma$. Let $\beta<\alpha$ be given. Let an ordinal δ_β be such that $F_\beta \subseteq \bigcup_{\gamma<\delta_\beta} E^-_\gamma$. For all ordinals δ, let $G^+_{\beta,\delta}$ denote the set of atoms in $[\mathcal{P}+\bigcup_{\gamma\leq\beta} F_\gamma]_\delta$, and let $G^-_{\beta,\delta}$ denote the set of negated atoms in $[\mathcal{P}+\bigcup_{\gamma\leq\beta} F_\gamma]_\delta$. It is easy to verify that for all ordinals δ, $G^-_{\beta,\delta} \subseteq \bigcup_{\gamma<\delta_\beta+\delta} E^-_\delta$ and $G^+_{\beta,\delta} \subseteq \bigcup_{\gamma<\delta_\beta+\delta} E^+_\delta$. Hence we can choose an ordinal δ with $[\mathcal{P}+\bigcup_{\beta<\alpha} F_\beta] \subseteq \bigcup_{\beta<\delta}(E^+_\beta \cup E^-_\beta)$. Since $[\mathcal{P}+\bigcup_{\beta<\alpha} F_\beta] \cup F_\alpha$ is a supporting extension for $\mathcal{P}$, we conclude that $F_\alpha \subseteq \bigcup_{\gamma\leq\delta} E^-_\gamma$, completing the proof of the proposition. □

8 Relationships to classical semantics

8.1 Kripke-Kleene semantics

Kripke-Kleene semantics is usually presented in a 3-valued logical setting. We use partial functions, with undefinedness implicitly representing the third truth value besides `true` and `false`. Definition 53 formalizes Kripke-Kleene models on the basis of logic programs, but note how the definition only involves the positive rules, so as to be faithful to the definition in the classical setting.

Definition 53. Given a logic program $\mathcal{P} = \left((\varphi^+_p, \varphi^-_p)\right)_{p\in\mathrm{Prd}(\mathcal{V})}$, a *KK-model of* $\mathcal{P}$ is a partial mapping from the set of closed members of $\mathfrak{L}_{\omega_1\omega}(\mathcal{V}\cup\{=\})$ into $\{\mathtt{true}, \mathtt{false}\}$ that satisfies the following conditions.[5]

- For all sets X of closed members of $\mathfrak{L}_{\omega_1\omega}(\mathcal{V}\cup\{=\})$,
  - $M(\bigvee X) = \mathtt{true}$ iff $M(\varphi) = \mathtt{true}$ for some $\varphi\in X$;
  - $M(\bigwedge X) = \mathtt{true}$ iff $M(\varphi) = \mathtt{true}$ for all $\varphi\in X$.
- For all $\varphi\in\mathfrak{L}_{\omega_1\omega}(\mathcal{V}\cup\{=\})$ and variables x with $\mathrm{fv}(\varphi)=\{x\}$,
  - $M(\exists x\varphi) = \mathtt{true}$ iff $M(\varphi[t/x]) = \mathtt{true}$ for some closed term t;
  - $M(\forall x\varphi) = \mathtt{true}$ iff $M(\varphi[t/x]) = \mathtt{true}$ for all closed terms t.

[5] "KK" in "KK-model" stands for "Kripke-Kleene."

- For all $n \in \mathbb{N}$, $p \in \mathrm{Prd}(\mathcal{V}, n)$, and closed terms $t_1, \ldots, t_n$, $M(p(t_1, \ldots, t_n))$ is defined iff $M(\varphi_p^+[t_1/v_1, \ldots, t_n/v_n])$ is defined, and is equal to `true` iff $M(\varphi_p^+[t_1/v_1, \ldots, t_n/v_n])$ is equal to `true`.
- For all closed terms t_1, t_2, $M(t_1 = t_2) = \texttt{true}$ if t_1 and t_2 are identical, and $M(t_1 \neq t_2) = \texttt{true}$ iff t_1 and t_2 are distinct.
- For all closed $\varphi \in \mathcal{L}_{\omega_1\omega}(\mathcal{V} \cup \{=\})$, $M(\varphi)$ is defined iff $M(\sim\varphi)$ is defined; moreover, if both are defined then $M(\varphi) = \texttt{false}$ iff $M(\sim\varphi) = \texttt{true}$.

Kripke-Kleene semantics focuses on minimal KK-models, defined next.

Definition 54. Given a logic program $\mathcal{P}$, a *minimal KK-model of* $\mathcal{P}$ is a KK-model M of $\mathcal{P}$ such that for all closed atoms φ over $\mathcal{V}$, the following holds.

- $M(\varphi) = \texttt{true}$ iff $N(\varphi) = \texttt{true}$ for all KK-models N of $\mathcal{P}$.
- $M(\varphi) = \texttt{false}$ iff $N(\varphi) = \texttt{false}$ for all KK-models N of $\mathcal{P}$.

Proposition 55 states the relationship between Kripke-Kleene's account of the classical setting—concerned only with symmetric logic programs—and our framework. Essentially, the proposition expresses that Kripke-Kleene semantics expresses adequately the way a symmetric logic program generates literals.

Proposition 55. *Let a symmetric logic program $\mathcal{P}$ be given. Let M be the unique minimal KK-model of $\mathcal{P}$. Then for all closed atoms φ,*

- $M(\varphi) = \texttt{true}$ *iff* $\varphi \in [\mathcal{P}]$*;*
- $M(\varphi) = \texttt{false}$ *iff* $\neg\varphi \in [\mathcal{P}]$.

Proof. The fact that $\mathcal{P}$ is symmetric implies immediately that for all KK-models N of $\mathcal{P}$, members n of $\mathbb{N}$, members p of $\mathrm{Prd}(\mathcal{V}, n)$, and closed terms $t_1, \ldots, t_n$, $N(\neg p(t_1, \ldots, t_n))$ is defined iff $N(\varphi_p^-[t_1/v_1, \ldots, t_n/v_n])$ is defined, and is equal to `true` iff $N(\varphi_p^-[t_1/v_1, \ldots, t_n/v_n])$ is equal to `true`. Let N be a KK-model of $\mathcal{P}$. It is then easy to verify by induction that for all ordinals α and members φ of $[\mathcal{P}]^\alpha$, $N(\varphi) = \texttt{true}$. Let N be the (unique) partial function from the set of closed members of $\mathcal{L}_{\omega_1\omega}(\mathcal{V} \cup \{=\})$ into $\{\texttt{true}, \texttt{false}\}$ such that for all closed $\varphi \in \mathcal{L}_{\omega_1\omega}(\mathcal{V} \cup \{=\})$, $N(\varphi) = \texttt{true}$ if $[\mathcal{P}] \vDash_{\mathcal{W}} \varphi$, $N(\varphi) = \texttt{false}$ if $[\mathcal{P}] \vDash_{\mathcal{W}} \sim\varphi$, and $N(\varphi)$ is undefined otherwise. By Property 32, N is a KK-model of $\mathcal{P}$, and it follows from the previous observation that $M = N$. □

8.2 Well-founded semantics

In this section, we turn to Rondogiannis and Wadge's treatment of the well-founded semantics for classical logic programs. The definition of Rondogiannis-Wadge models is made easier and more natural by first extending the vocabulary $\mathcal{V}$ with a set of new predicate symbols, one for each predicate symbol in $\mathcal{V}$.

Notation 56. We introduce for all $n \in \mathbb{N}$ and $p \in \mathrm{Prd}(\mathcal{V}, n)$ an n-ary predicate symbol, denoted $\widetilde{p}$, distinct from all symbols in $\mathcal{V}$, and such that for all distinct members p and q of $\mathrm{Prd}(\mathcal{V})$, $\widetilde{p}$ and $\widetilde{q}$ are distinct.

We set $\widetilde{\mathcal{V}} = \mathcal{V} \cup \{\widetilde{p} \mid p \in \mathrm{Prd}(\mathcal{V})\}$.

Notation 56 allows one to naturally transform a formula φ over $\mathcal{V} \cup \{=\}$ to a positive formula $\oplus_{\sim}\varphi$ over $\widetilde{\mathcal{V}} \cup \{=\}$, or to a negative formula $\ominus_{\sim}\varphi$ over $\widetilde{\mathcal{V}} \cup \{=\}$. This is made precise in the next definition.

Definition 57. Let a formula φ over $\mathcal{V} \cup \{=\}$ be given.

We denote by $\oplus_{\sim}\varphi$ the formula over $\widetilde{\mathcal{V}} \cup \{=\}$ obtained from φ by replacing all occurrences of negated atoms, say $\neg p(t_1, \ldots, t_n)$, with $\widetilde{p}(t_1, \ldots, t_n)$.

We denote by $\ominus_{\sim}\varphi$ the formula over $\widetilde{\mathcal{V}} \cup \{=\}$ obtained from φ by replacing all occurrences of nonnegated atoms, say $p(t_1, \ldots, t_n)$, with $\neg\widetilde{p}(t_1, \ldots, t_n)$.

Given a logic program $\mathcal{P}$ over $\mathcal{V}$, Definition 57 allows one to naturally transform $\mathcal{P}$ into a logic program over $\widetilde{\mathcal{V}}$, where the bodies of the positive rules are positive formulas over $\widetilde{\mathcal{V}} \cup \{=\}$, and the bodies of the negative rules are negative formulas over $\widetilde{\mathcal{V}} \cup \{=\}$. But of course, the link between the predicate symbols in $\mathcal{V}$ and the extra predicate symbols in $\widetilde{\mathcal{V}} \setminus \mathcal{V}$ should be made, with for all $n \in \mathbb{N}$ and $p \in \mathrm{Prd}(\mathcal{V}, n)$, an extra pair of rules $\widetilde{p}(v_1, \ldots, v_n) \leftarrow \neg p(v_1, \ldots, v_n)$ and $\neg\widetilde{p}(v_1, \ldots, v_n) \leftarrow p(v_1, \ldots, v_n)$.

We are now ready to define the models underlying Rondogiannis-Wadge semantics. We use partial functions, with undefinedness implicitly representing the "middle" truth value besides $\mathtt{true}_\alpha$ and and $\mathtt{false}_\alpha$, $\alpha \in \mathrm{Ord}$. Definitions 58 and 59 formalize Rondogiannis-Wadge models on the basis of logic programs, but note how the definition only involves the positive rules, so as to be faithful to the definition in their setting. Definition 58 conveniently uses the extended vocabulary, and Definition 59 gets back to the original vocabulary, so as to be in accordance with the original notion.

Definition 58. Let a logic program $\mathcal{P} = \left((\varphi_p^+, \varphi_p^-)\right)_{p \in \mathrm{Prd}(\mathcal{V})}$ be given. An *extended RW-model of* $\mathcal{P}$ is defined as a partial mapping from the set of closed members of $\mathcal{L}_{\omega_1\omega}(\widetilde{\mathcal{V}} \cup \{=\})$ into $\bigcup_{\alpha \in \mathrm{Ord}} \{\mathtt{true}_\alpha, \mathtt{false}_\alpha\}$ that satisfies the following conditions.

- For all sets X of closed members of $\mathcal{L}_{\omega_1\omega}(\widetilde{\mathcal{V}} \cup \{=\})$ and ordinals α,

- $M(\bigvee X) = \mathtt{true}_\alpha$ iff $M(\varphi) = \mathtt{true}_\alpha$ for some $\varphi \in X$, and there exists no $\psi \in X$ and $\beta < \alpha$ such that $M(\psi) = \mathtt{true}_\beta$;
- $M(\bigwedge X) = \mathtt{true}_\alpha$ iff α is the least ordinal such that for all $\psi \in X$, $M(\psi) = \mathtt{true}_\beta$ for some $\beta \leq \alpha$.

- For all $\varphi \in \mathfrak{L}_{\omega_1\omega}(\widetilde{\mathcal{V}} \cup \{=\})$, variables x with $\mathrm{fv}(\varphi) = \{x\}$, and ordinals α,
  - $M(\exists x\varphi) = \mathtt{true}_\alpha$ iff $M(\varphi[t/x]) = \mathtt{true}_\alpha$ for some closed term t, and there exists no closed term t and $\beta < \alpha$ such that $M(\varphi[t/x]) = \mathtt{true}_\beta$;
  - $M(\forall x\varphi) = \mathtt{true}_\alpha$ iff α is the least ordinal such that for all closed terms t, $M(\varphi[t/x]) = \mathtt{true}_\beta$ for some $\beta \leq \alpha$.

- For all $n \in \mathbb{N}$, $p \in \mathrm{Prd}(\mathcal{V}, n)$, closed terms $t_1, \ldots, t_n$, and ordinals α,
  - $M(p(t_1, \ldots, t_n))$ is defined iff $M(\oplus_\sim \varphi_p^+[t_1/v_1, \ldots, t_n/v_n])$ is defined, and equal to $\mathtt{true}_\alpha$ iff $M(\oplus_\sim \varphi_p^+[t_1/v_1, \ldots, t_n/v_n]) = \mathtt{true}_\alpha$;
  - $M(\widetilde{p}(t_1, \ldots, t_n))$ is defined iff $M(p(t_1, \ldots, t_n))$ is defined, and equal to $\mathtt{true}_{\alpha+1}$ iff $M(p(t_1, \ldots, t_n)) = \mathtt{false}_\alpha$.

- For all closed terms t_1 and t_2, $M(t_1 = t_2) = \mathtt{true}_0$ if t_1 and t_2 are identical, and $M(t_1 \neq t_2) = \mathtt{true}_0$ iff t_1 and t_2 are distinct.

- For all closed $\varphi \in \mathfrak{L}_{\omega_1\omega}(\widetilde{\mathcal{V}} \cup \{=\})$, $M(\varphi)$ is defined iff $M(\sim\varphi)$ is defined; moreover, if both are defined then for all ordinals α, $M(\varphi) = \mathtt{false}_\alpha$ iff $M(\sim\varphi) = \mathtt{true}_\alpha$.

Definition 59. Given a logic program $\mathcal{P}$, an *RW-model of* $\mathcal{P}$ is defined as the restriction to the set of closed members of $\mathfrak{L}_{\omega_1\omega}(\mathcal{V} \cup \{=\})$ of an extended RW-model of $\mathcal{P}$.

Rondogiannis-Wadge semantics focuses on minimal RW-models, defined next.

Definition 60. Let a logic program $\mathcal{P}$ be given. A *minimal RW-model of* $\mathcal{P}$ is defined as an RW-model M of $\mathcal{P}$ such that for all ordinals α and closed atoms φ, the following holds.

- $M(\varphi) = \mathtt{true}_\alpha$ iff all RW-models N of $\mathcal{P}$ have the following property. Suppose that for all $\beta < \alpha$ and closed atoms ψ, if $M(\psi)$ is equal to either $\mathtt{true}_\beta$ or $\mathtt{false}_\beta$ then $N(\psi) = M(\psi)$. Then $N(\varphi) = \mathtt{true}_\alpha$.

- $M(\varphi) = \mathtt{false}_\alpha$ iff there exists an RW-model N of $\mathcal{P}$ such that
  - for all ordinals $\beta < \alpha$ and closed atoms ψ, if $M(\psi)$ is equal to either $\mathtt{true}_\beta$ or $\mathtt{false}_\beta$ then $N(\psi) = M(\psi)$;

– $N(\varphi) = \mathtt{false}_\alpha$.

Proposition 61 states the relationship between the Rondogiannis-Wadge' s account of the classical setting—concerned only with what we have defined as symmetric logic programs—and our framework. Essentially, the proposition expresses that Rondogiannis-Wadge semantics expresses adequately the way a symmetric logic program *whose rules are modified so as to force the generation of a large set of negated atoms, in one amongst many manners of forcing the generation of sets of negated atoms*, generates literals. Presented with a pair of rules of the form $p \leftarrow p$ and $\neg p \leftarrow \neg p$ (be the latter implicit), we certainly would not claim that you should irrepressibly want to make $\neg p$ false in all intended models of a logic program that contains those rules. But if you want to *modify* the rules and play instead with $p \leftarrow \bigwedge \emptyset$ and $\neg p \leftarrow \bigvee \emptyset$, then you can go for it (just make that clear, *i.e.*, explicit). You could also decide to modify the rules and play with $p \leftarrow \bigvee \emptyset$ and $\neg p \leftarrow \bigwedge \emptyset$, because you would like to generate more literals than you have first been allowed to, and either have no bias towards negation, or have a bias towards nonnegation.

Proposition 61. *Let a symmetric logic program $\mathcal{P}$ be given. Let M be the unique minimal RW-model of $\mathcal{P}$. Let $\left((E_\alpha^+, E_\alpha^-)\right)_{\alpha \in Ord}$ be the RW-sequence for $\mathcal{P}$. Then for all ordinals α and closed atoms φ,*

- $M(\varphi) = \mathtt{true}_\alpha$ *iff* $\varphi \in E_\alpha^+$;
- $M(\varphi) = \mathtt{false}_\alpha$ *iff* $\varphi \in E_\alpha^-$.

Proof. The fact that $\mathcal{P}$ is symmetric implies immediately that for all extended RW-models N of $\mathcal{P}$, $n \in \mathbb{N}$, $p \in \mathrm{Prd}(\mathcal{V}, n)$, closed terms $t_1, \ldots, t_n$, and $\alpha \in \mathrm{Ord}$,

- $N(\neg p(t_1, \ldots, t_n))$ is defined iff $N(\ominus_\sim \varphi_p^-[t_1/v_1, \ldots, t_n/v_n])$ is defined, and is equal to $\mathtt{true}_\alpha$ iff $N(\ominus_\sim \varphi_p^-[t_1/v_1, \ldots, t_n/v_n]) = \mathtt{true}_\alpha$;
- $N(\neg \widetilde{p}(t_1, \ldots, t_n))$ is defined iff $N(p(t_1, \ldots, t_n))$ is defined, and is equal to $\mathtt{true}_{\alpha+1}$ iff $N(p(t_1, \ldots, t_n)) = \mathtt{false}_\alpha$.

Set

$$X = \Big\{ \forall \big(\neg p(v_1, \ldots, v_n) \leftrightarrow \widetilde{p}(v_1, \ldots, v_n) \big) \mid n \in \mathbb{N},\ p \in \mathrm{Prd}(\mathcal{V}, n) \Big\}.$$

Let N be the partial function from the set of closed members of $\mathcal{L}_{\omega_1\omega}(\widetilde{\mathcal{V}} \cup \{=\})$ into $\bigcup_{\alpha \in \mathrm{Ord}} \{\mathtt{true}_\alpha, \mathtt{false}_\alpha\}$ such that for all closed members φ of $\mathcal{L}_{\omega_1\omega}(\widetilde{\mathcal{V}} \cup \{=\})$ and ordinals α,

- $N(\varphi) = \mathtt{true}_\alpha$ if α is the least ordinal such that φ is a logical consequence of $X \cup \bigcup_{\beta \leq \alpha} (E_\beta^+ \cup E_\beta^-)$ in $\mathcal{W}$;

- $N(\varphi) = \texttt{false}_\alpha$ if α is the least ordinal such that $\sim\varphi$ is a logical consequence of $X \cup \bigcup_{\beta\leq\alpha}(E^+_\beta \cup E^-_\beta)$ in $\mathcal{W}$;
- $N(\varphi)$ is undefined otherwise.

It is easy to verify by induction that N is the extended RW-model of $\mathcal{P}$ whose restriction to $\mathcal{L}_{\omega_1\omega}(\mathcal{V} \cup \{=\})$ is M. □

Note that Proposition 61 expresses how Rondogiannis-Wadge semantics exploits a particular sequence of *successive* modifications of $\mathcal{P}$. We could incorporate the members of $\bigcup_{\alpha\in\text{Ord}} E^-_\alpha$ in the bodies of $\mathcal{P}$'s rules at once, in other words, consider $\mathcal{P} + \bigcup_{\alpha\in\text{Ord}} E^-_\alpha$. That would also relate our framework to the well-founded semantics, with $[\mathcal{P} + \bigcup_{\alpha\in\text{Ord}} E^-_\alpha]$ being precisely the set of closed literals true in the 3-valued well-founded model of $\mathcal{P}$—this is exactly what the second clause in Proposition 52 tells us, on the basis of Proposition 61 and the relationship between the minimal RW-model of $\mathcal{P}$ and the 3-valued well-founded model of $\mathcal{P}$. But there would be no relationship between the members of the sets $[\mathcal{P} + \bigcup_{\alpha\in\text{Ord}} E^-_\alpha]_\beta$, $\beta \in$ Ord, and their truth values in the minimal RW-model of $\mathcal{P}$.

8.3 Stable model semantics

Note that a complete set E of closed literals can trivially be identified with the standard structure defined as the set of atoms in E. Hence given a theory T and a complete set E of literals, if $E \cup T$ is consistent in $\mathcal{W}$ (or equivalently, if T is a logical consequence of E in $\mathcal{W}$) then E can be viewed as a (standard) model of T. Let $\mathcal{P}$ be a logic program whose classical logical form, say T, is consistent in $\mathcal{W}$, and let E be a complete set of closed literals that is a model of T. By Property 34, $[\mathcal{P} + E] = E$. A natural question to ask is: what are the subsets F of E such that $[\mathcal{P} + F] = E$? Tough such a set F might not be a supporting extension for $\mathcal{P}$, it is still preserved when transforming $\mathcal{P}$ to $\mathcal{P} + F$ (in the sense that it is included in $[\mathcal{P} + F]$), and it can be seen as a generator of a complete supporting extension for $\mathcal{P}$, namely $[\mathcal{P} + E]$. We will see that the stable model semantics corresponds to the particular case when we can take for F precisely the set of negated atoms in E. (See [11] for another approach that is also based on the idea that the stable model semantics corresponds to adding negative information to the Kripke-Kleene semantics.) But let us beforehand define the more general notion that we have introduced.

Definition 62. Given a logic program $\mathcal{P}$, a *completing extension for* $\mathcal{P}$ is defined as any consistent set E of closed literals such that $[\mathcal{P} + E]$ is a complete superset of E.

We first check that a completing extension of a logic program $\mathcal{P}$ is a model of the classical logical form of $\mathcal{P}$. This is expressed in Proposition 65, and proved with the help of Proposition 63, which is interesting in its own right, and Property 64.

Proposition 63. *For all logic programs $\mathcal{P}$, if $[\mathcal{P}]$ is complete then the set of atoms in $[\mathcal{P}]$ is a model of* $\mathrm{CLF}(\mathcal{P})$.

Proof. Let a logic program $\mathcal{P} = \left((\varphi_p^+, \varphi_p^-)\right)_{p \in \mathrm{Prd}(\mathcal{V})}$ be given. Assume that $[\mathcal{P}]$ is complete. Let $n \in \mathbb{N}$, a member p of $\mathrm{Prd}(\mathcal{V}, n)$, and closed terms $t_1, \ldots, t_n$ be given. If $p(t_1, \ldots, t_n)$ belongs to $[\mathcal{P}]$ then $[\mathcal{P}] \models_{\mathcal{W}} \varphi_p^+[t_1/v_1, \ldots, t_n/v_n]$, impling that $[\mathcal{P}] \models_{\mathcal{W}} {\sim}\varphi_p^-[t_1/v_1, \ldots, t_n/v_n]$. Similarly, if $\neg p(t_1, \ldots, t_n)$ belongs to $[\mathcal{P}]$ then $[\mathcal{P}] \models_{\mathcal{W}} \varphi_p^-[t_1/v_1, \ldots, t_n/v_n]$, impling that $[\mathcal{P}] \models_{\mathcal{W}} {\sim}\varphi_p^+[t_1/v_1, \ldots, t_n/v_n]$. So we have verified that

- $[\mathcal{P}] \models p(t_1, \ldots, t_n)$ iff $[\mathcal{P}] \models_{\mathcal{W}} \varphi_p^+[t_1/v_1, \ldots, t_n/v_n]$, and
- $[\mathcal{P}] \models \neg p(t_1, \ldots, t_n)$ iff $[\mathcal{P}] \models_{\mathcal{W}} \varphi_p^-[t_1/v_1, \ldots, t_n/v_n]$,

which completes the proof of the proposition. □

Property 64. *For all complete sets E of closed literals and sentences φ over $\mathcal{V} \cup \{=\}$, E logically implies $\varphi \leftrightarrow \circledcirc_E \varphi$ in $\mathcal{W}$.*

Proposition 65. *For all logic programs $\mathcal{P}$ and completing extensions E for $\mathcal{P}$, the set of atoms in $[\mathcal{P} + E]$ is a model of* $\mathrm{CLF}(\mathcal{P})$.

Proof. Let a logic program $\mathcal{P}$ and a completing extension E for $\mathcal{P}$ be given. By Property 64, $\mathrm{CLF}(\mathcal{P}) \cup [\mathcal{P} + E]$ is logically equivalent to $\mathrm{CLF}(\mathcal{P} + E) \cup [\mathcal{P} + E]$ in $\mathcal{W}$. By Corollary 39, the assumptions that E is included in $[\mathcal{P} + E]$ and $[\mathcal{P} + E]$ is complete yield $[\mathcal{P} + [\mathcal{P} + E]] = [\mathcal{P} + E]$. Together with Property 10, this implies that $\mathrm{CLF}(\mathcal{P} + E) \cup [\mathcal{P} + E]$ is logically equivalent to $\mathrm{CLF}(\mathcal{P} + E)$ in $\mathcal{W}$. Hence $\mathrm{CLF}(\mathcal{P}) \cup [\mathcal{P} + E]$ is logically equivalent to $\mathrm{CLF}(\mathcal{P} + E)$ in $\mathcal{W}$. Moreover, using again the fact that $[\mathcal{P} + [\mathcal{P} + E]]$ is equal to the complete set $[\mathcal{P} + E]$, we infer from Proposition 63 that $\mathrm{CLF}(\mathcal{P} + E)$ is consistent in $\mathcal{W}$. Hence $\mathrm{CLF}(\mathcal{P}) \cup [\mathcal{P} + E]$ is consistent in $\mathcal{W}$, and we conclude that $[\mathcal{P} + E] \models_{\mathcal{W}} \mathrm{CLF}(\mathcal{P})$. □

Corollary 66. *For all logic programs $\mathcal{P}$ and for all completing extensions E for $\mathcal{P}$, $[\mathcal{P} + E]$ is a supporting extension for $\mathcal{P}$.*

We now show that stable models are particular kinds of completing extensions. Definition 67 describes a stable model using the notation of our framework rather than the Lloyd-Topor transformation [10].

Definition 67. Given a logic program $\mathcal{P} = \left((\varphi_p^+, \varphi_p^-)\right)_{p \in \mathrm{Prd}(\mathcal{V})}$ and a complete set E of closed literals, we say that E is *stable for* $\mathcal{P}$ just in case for all closed atoms φ, $\varphi \in E$ iff

$$\left\{\forall(\oplus_E \varphi_p^+ \rightarrow p(v_1, \ldots, v_n)) \mid n \in \mathbb{N},\ p \in \mathrm{Prd}(\mathcal{V}, n)\right\} \vDash_{\mathcal{W}} \varphi.^6$$

Note that Definition 67 does not assume that $\mathcal{P}$ is symmetric; actually it only depends on the positive part of $\mathcal{P}$. This suggests that if E is stable for $\mathcal{P}$ then, denoting by E^+ the set of atoms in E and denoting by E^- the set of negated atoms in E, we should check whether $[\mathcal{P} + E^-]$ is equal to E^+. This is indeed the case, as expressed in Proposition 71. But we are more interested in Proposition 72 as it yields, thanks to Corollary 73, a relationship between stable models and a notion—that of completing extension—which does not make any distinction between atoms and negated atoms, or between positive rules and negative rules. The proof of Proposition 71 is reduced to three easy properties, given next.

Property 68. *Let a logic program $\mathcal{P} = \left((\varphi_p^+, \varphi_p^-)\right)_{p \in \mathrm{Prd}(\mathcal{V})}$ and a complete set E of closed literals be given. Let E^- be the set of negated atoms in E. Then E is stable for $\mathcal{P}$ iff*

$$E^- \cup \left\{\forall(\oplus_E \varphi_p^+ \rightarrow p(v_1, \ldots, v_n)) \mid n \in \mathbb{N},\ p \in \mathrm{Prd}(\mathcal{V}, n)\right\}$$

is logically equivalent to E in $\mathcal{W}$.

Property 69. *Let a sentence φ over $\mathcal{V} \cup \{=\}$ be given.*

- *For all sets E of negated closed atoms, $\{\oplus_E \varphi\} \vDash_{\mathcal{W}} \circledcirc_E \varphi$.*
- *For all sets E of closed atoms, $\{\ominus_E \varphi\} \vDash_{\mathcal{W}} \circledcirc_E \varphi$.*

Property 70. *Let a sentence φ over $\mathcal{V} \cup \{=\}$ be given.*

- *For all sets E of negated closed atoms, $E \cup \{\circledcirc_E \varphi\} \vDash_{\mathcal{W}} \oplus_E \varphi$.*
- *For all sets E of closed atoms, $E \cup \{\circledcirc_E \varphi\} \vDash_{\mathcal{W}} \ominus_E \varphi$.*

Proposition 71. *Let a logic program $\mathcal{P} = \left((\varphi_p^+, \varphi_p^-)\right)_{p \in \mathrm{Prd}(\mathcal{V})}$ and a complete set E of closed literals be given. Let E^- be the set of negated atoms in E, and let E^+ be the set of atoms in E. Let $\mathcal{P}^+$ be the logic program $\left((\varphi_p^+, \bigvee \emptyset)\right)_{p \in \mathrm{Prd}(\mathcal{V})}$. Then E is stable for $\mathcal{P}$ iff $[\mathcal{P}^+ + E^-] = E^+$.*

[6] Given a sentence φ over $\mathcal{V} \cup \{=\}$ and a set E of closed literals, if E^- is the set of negated atoms in E then $\oplus_E \varphi$ is obviously identical to $\oplus_{E^-} \varphi$.

Proof. The proposition follows immediately from Properties 68, 69 and 70. □

Proposition 72. *Let a logic program $\mathcal{P}$ and a complete set E of closed literals be given. Let E^- be the set of negated atoms in E. Then E is stable for $\mathcal{P}$ iff $[\mathcal{P} + E^-] = E$.*

Proof. Let $\mathcal{P}^+$ be the logic program $\left((\varphi_p^+, \bigvee \emptyset)\right)_{p \in \mathrm{Prd}(\mathcal{V})}$. By Proposition 71, it suffices to show that $[\mathcal{P}^+ + E^-] = E^+$ iff $[\mathcal{P} + E^-] = E$. It is easy to verify that if $[\mathcal{P} + E^-] = E$ then $[\mathcal{P}^+ + E^-] = E^+$. Conversely, assume that $[\mathcal{P}^+ + E^-] = E^+$. Let an ordinal α be such that $[\mathcal{P}^+ + E^-]_\alpha = E^+$. Then trivially, $E^+ \subseteq [\mathcal{P} + E^-]_\alpha$, which implies immediately that $[\mathcal{P} + E^-]_{\alpha+1} = E$. □

Corollary 73. *Let a logic program $\mathcal{P}$ and a complete set E of closed literals be given. Let E^- be the set of negated atoms in E. If E is stable for $\mathcal{P}$ then E^- is a completing extension for $\mathcal{P}$.*

We end this section with an example that illustrates the various notions of extensions we have been dealing with. For simplicity, we consider a symmetric logic program, and the bodies of the positive rules of this program have no occurrence of negation. But these constraints play no special role at all.

Example 74. Suppose that $\mathcal{V}$ consists of 3 nullary predicate symbols p_1, p_2 and p_3. Let $\mathcal{P}$ be the symmetric logic program given by the following formulas.

$$\varphi_{p_1}^+ \text{ is } p_1 \wedge p_2 \qquad \varphi_{p_2}^+ \text{ is } p_2 \vee p_3 \qquad \varphi_{p_3}^+ \text{ is } p_3$$

- The set of standard models of $\mathcal{P}$ consists of the following sets.

$$\{p_1, p_2, p_3\} \qquad \{p_1, p_2\} \qquad \{p_2, p_3\} \qquad \{p_2\} \qquad \emptyset$$

- The set of completing extensions for $\mathcal{P}$ consists of the following sets.

$$\begin{array}{llll} \{p_1, p_2, p_3\} & \{p_1, p_3\} & \{p_1, p_2, \neg p_3\} & \{\neg p_1, p_2, p_3\} \\ \{\neg p_1, p_3\} & \{\neg p_1, p_2, \neg p_3\} & \{\neg p_1, \neg p_2, \neg p_3\} & \{\neg p_2, \neg p_3\} \end{array}$$

- There is a unique complete set of closed literals that is stable for $\mathcal{P}$, namely $\{\neg p_1, \neg p_2, \neg p_3\}$.

- The sets of closed literals that are supporting extensions for $\mathcal{P}$ are all the completing extensions for $\mathcal{P}$ except $\{p_1, p_3\}$.

9 Inferences in the style of Logic programming

Let $\mathcal{P}$ be the logic program described in the following example.

Example 75. Suppose that $\mathcal{V}$ consists of three nullary predicates p, q and r. Let $\mathcal{P}$ be the logic program represented by the following formulas.

$$\varphi_p^+ \equiv \neg p \qquad \varphi_q^+ \equiv q \qquad \varphi_r^+ \equiv \bigvee \emptyset$$
$$\varphi_p^- \equiv \neg q \wedge \neg r \qquad \varphi_q^- \equiv \neg q \qquad \varphi_r^- \equiv r$$

Then $[\mathcal{P}] = \emptyset$ and $\mathrm{CLF}(\mathcal{P})$ consists of the following sentences.

$$\bigvee\{p\} \qquad \bigvee\{\bigvee\{q, r\}, \neg p\} \qquad \bigvee\{\neg q, q\} \qquad \bigvee\{\bigwedge \emptyset, r\} \qquad \bigvee\{\neg r\}$$

It is clear that p, q and $\neg r$ could be generated from $\mathrm{CLF}(\mathcal{P})$ by applying the second of the following principles:

- Given a set X of sentences, if $\bigwedge X$ is assumed or has been derived then all members of X can be derived.
- Given a set X of sentences and a member φ of X, if $\bigvee X$ is assumed or has been derived, and if all members of X except φ either have been assumed to be false or have been refuted (*i.e.*, if for all $\psi \in X$ distinct from φ, $\sim\psi$ has been either assumed or derived), then φ can be derived.
- Given a sentence of the form $\forall x\varphi$, if $\forall x\varphi$ is assumed or has been derived then for all closed terms t, $\varphi[t/x]$ can be derived.
- Given a sentence of the form $\exists x\varphi$ and a closed term t, if $\exists x\varphi$ is assumed or has been derived, and if for all closed terms t' distinct from t, $\varphi[t'/x]$ either has been assumed to be false or has been refuted (*i.e.*, $\sim\varphi[t'/x]$ has been either assumed or derived), then $\varphi[t/x]$ can be derived.

The second principle above is the key principle behind the machinery of Logic programming, but it is there restricted to sentences of the form $\bigvee\{\varphi, \psi\}$, with ψ being the closed literal obtained from taking a closed instance of the head of a rule, and φ being the corresponding closed instance of the body of that rule. One part of the restriction is that disjunctions are over sets of cardinality two or one (two in the interesting cases). But more importantly, the restriction is that one formula corresponds to the head and the other formula corresponds to the body, and that this distinction matters: an order is imposed, which is lost when disjunctions operate over sets. Whether such a restriction is really essential is disputable. The principles above appear as more fundamental. This suggests a kind

of inference from arbitrary sets of sentences "in the style of Logic programming." It is captured by the next definition, using the forcing relation that has been described in Definition 12.

Definition 76. Given a set T of sentences, we denote by $[T]$ the $\subseteq$-smallest set of closed literals and by $\widehat{T}$ the $\subseteq$-smallest set of sentences that satisfy the following conditions.

- T is a subset of $\widehat{T}$.
- For all literals φ, if $\varphi \in \widehat{T}$ then $\varphi \in [T]$.
- For all members of $\widehat{T}$ of the form $\bigwedge X$, X is included in $\widehat{T}$.
- For all members of $\widehat{T}$ of the form $\bigvee X$ and for all members φ of X, φ belongs to $\widehat{T}$ whenever $[T] \Vdash_{\mathcal{W}} \sim \bigvee X \setminus \{\varphi\}$.
- For all members of $\widehat{T}$ of the form $\forall x \varphi$, $\varphi[t/x]$ belongs to $\widehat{T}$ for all closed terms t.
- For all members of $\widehat{T}$ of the form $\exists x \varphi$ and for all closed terms t, $\varphi[t/x]$ belongs to $\widehat{T}$ whenever $[T] \Vdash_{\mathcal{W}} \forall x (\sim \varphi \vee x = t)$.

Example 77. If $\mathcal{P}$ is the logic program described in Example 75 then $[\mathrm{CLF}(\mathcal{P})]$ is equal to $\{p, q, \neg r\}$.

Examples 75 and 77 illustrates the next property, whose verification is immediate.

Property 78. *For all logic programs $\mathcal{P}$, $[\mathcal{P}] \subseteq [\mathrm{CLF}(\mathcal{P})]$.*

Logic programs are still general enough to perform the inferences in the style of Logic programming from arbitrary sets of sentences that have been described, as expressed in the last definition and property of this paper.

Definition 79. Let T be a countable set of sentences. Let C be the $\subseteq$-minimal set of pairs of formulas over $\mathcal{V} \cup \{=\}$ that satisfies the following conditions.

- $T \times \{\bigwedge \emptyset\} \subseteq C$.
- For all members of C of the form $(\bigwedge X, \psi)$ and for all $\varphi \in X$, $(\varphi, \psi) \in C$.
- For all members of C of the form $(\forall x \varphi, \psi)$ and for all closed terms t, C contains $(\varphi[t/x], \psi)$.

- For all members of C of the form $(\bigvee X, \bigwedge Y)$ and for all $\varphi \in X$, C contains $(\varphi, \bigwedge Y \cup \{\sim\chi \mid \chi \in X \setminus \{\varphi\}\})$.
- For all members of C of the form $(\exists x\varphi, Y)$ and for all closed terms t, C contains $(\varphi[t/x], \bigwedge Y \cup \{\forall x(\sim\varphi \vee x = t)\})$.

For all $n \in \mathbb{N}$ and $p \in \mathrm{Prd}(\mathcal{V}, n)$,

- let φ_p^+ denote the disjunction of the set of all formulas over $\mathcal{V} \cup \{=\}$ of the form $\bigwedge\{t_i = v_i \mid i \leq i \leq n\} \cup X$ such that C contains $(p(t_1, \ldots, t_n), \bigwedge X)$;
- let φ_p^- denote the disjunction of the set of all formulas over $\mathcal{V} \cup \{=\}$ of the form $\bigwedge\{t_i = v_i \mid i \leq i \leq n\} \cup X$ such that C contains $(\neg p(t_1, \ldots, t_n), \bigwedge X)$.

Then $((\varphi_p^+, \varphi_p^-))_{p \in \mathrm{Prd}(\mathcal{V})}$ is a logic program; we call it the *logic program generated by* T, and denote it by $\mathrm{LP}(T)$.

Example 80. Let $\mathcal{P}$ be the logic program described in Example 75. Then the logic program $\mathrm{LP}(\mathrm{CLF}(T))$ is represented by the following formulas.

$$\begin{array}{ll} \varphi_p^+ \equiv \bigvee\{\bigwedge\emptyset\} & \varphi_p^- \equiv \bigvee\{\bigwedge\{\bigwedge\{\neg q, \neg r\}\}\} \\ \varphi_q^+ \equiv \bigvee\{\bigwedge\{p, \neg r\}, \bigwedge\{q\}\} & \varphi_q^- \equiv \bigvee\{\bigwedge\{\neg q\}\} \\ \varphi_r^+ \equiv \bigvee\{\bigwedge\{p, \neg q\}, \bigwedge\{\bigvee\emptyset\}\} & \varphi_r^- \equiv \bigvee\{\bigwedge\emptyset\} \end{array}$$

Property 81. *For all countable sets T of sentences,* $[T] = [\mathrm{LP}(T)]$.

10 Conclusion

The notion of logic program has not only be generalized; it has been redefined. We have carefully not defined a logic program as a set of logical formulas. We have chosen to model the behavior of a set of rules that can fire transfinitely often, hence to provide an operational semantics, which does not require to represent rules as logical formulas. Since the classical logical form of a logic program $\mathcal{P}$ does not adequately represent the behavior of $\mathcal{P}$, that classical logical form does not provide a sound basis for a declarative semantics. This is not because the logical symbols, including negation, should receive a different meaning than they do in the classical setting of first-order logic. An intuitionistic interpretation of the logical symbols would not provide an adequate semantics either, though it has been applied to particular classes of logic programs [4]. Still it is perfectly possible to provide a semantics where the logical symbols keep their classical meaning, and where the behavior of a logic program can be described in terms of a classical

notion of logical consequence: we can show that given a logic program $\mathcal{P}$, a closed literal φ and ordinals α, β, some statement, written $\odot_\alpha \varphi$, is such that $T \vDash \odot_\alpha \varphi$ iff $\varphi \in [\mathcal{P}]_\beta$ and $\alpha > \beta$, where T is some set of sentences that is obtained from $\mathcal{P}$ as easily as the classical logical form, and where $\vDash$ is a classical notion of logical consequence applied to interpretations—called aggregative multistructures—that generalize standard structures. So there is a straightforward translation from the language for the operational semantics to the language for the declarative semantics, and conversely, but such a translation is needed; both languages cannot be the same. Why should they? Certainly, a simple isomorphism between two languages and their associated semantics is as satisfactory as two different semantics for the same language. Another paper will be devoted to this topic.

References

[1] José Júlio Alferes, Luís Moniz Pereira, and Teodor C. Przymusinski, *'Classical' Negation in Nonmonotonic Reasoning and Logic Programming*, Journal of Automated Reasoning **20** (1998), no. 1, 107–142. Logics for artificial intelligence (Évora, 1996).

[2] Krzysztof R. Apt and Roland N. Bol, *Logic Programming and Negation: A Survey*, Journal of Logic Programming **19/20** (1994), 9–71.

[3] Ofer Arieli and Arnon Avron, *The Value of the Four Values*, Artificial Intelligence **102** (1998), no. 1, 97–141.

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

[5] Keith L. Clark, *Negation as failure*, Logic and databases. Proceedings of the Symposium held at the Centre d'Études et de Recherches de l'École Nationale Supérieure de l'Aéronautique et de l'Espace de Toulouse (C.E.R.T.), Toulouse, November 16–18, 1977, 1978, pp. viii+458.

[6] Marc Denecker, Maurice Bruynooghe, and Victor Marek, *Logic Programming Revisited: Logic Programs as Inductive Definitions*, ACM Transactions on Computational Logic **2** (2001), no. 4, 623–654.

[7] Melvin Fitting, *Bilattices and the Semantics of Logic Programming*, Journal of Logic Programming **11** (1991), no. 2, 91–116.

[8] Michael Gelfond and Vladimir Lifschitz, *The Stable Model Semantics for Logic Programming*, Proceedings of the Fifth International Conference on Logic Programming, 1988, pp. 1070–1080.

[9] Vladimir Lifschitz, *Answer Set Programming and Plan Generation*, Artificial Intelligence **138** (2002), no. 1-2, 39–54.

[10] John W. Lloyd, *Foundations of logic programming; 2nd extended ed.*, Springer-Verlag New York, Inc., New York, NY, USA, 1987.

[11] Yann Loyer and Umberto Straccia, *Epistemic Foundation of Stable Model Semantics*, Theory and Practice of Logic Programming **6** (2006), no. 4, 355–393.

[12] Halina Przymusinska and Teodor C. Przymusinski, *Weakly Stratified Logic Programs*, Fundamenta Informaticae **XIII** (1990), 51–65.

[13] Teodor C. Przymusinski, *Every logic program has a natural stratification and an iterated least fixed point model*, PODS '89: Proceedings of the eighth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, 1989, pp. 11–21.

[14] Panos Rondogiannis and William W. Wadge, *Minimum Model Semantics for Logic Programs with Negation-as-Failure*, ACM Transactions on Computational Logic **5** (2005), no. 2, 441–467.

[15] Raymond M. Smullyan, *First-order logic*, Ergebnisse der Mathematik und ihrer Grenzgebiete, Band 43, Springer-Verlag New York, Inc., New York, 1968.

[16] Maarten H. Van Emden and Robert A. Kowalski, *The Semantics of Predicate Logic as a Programming Language*, Journal of the Association for Computing Machinery **23** (1976), no. 4, 733–742.

[17] Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf, *The Well-Founded Semantics for General Logic Programs*, Journal of the Association for Computing Machinery **38** (1991), no. 3, 619–649.