A Model Theory for Paraconsistent Logic Programming

Carlos Viegas Damásio and Luís Moniz Pereira[⋆]

CRIA, Uninova and DCS, U. Nova de Lisboa
2825 Monte da Caparica
Portugal
{cd|lmp}@fct.unl.pt

Abstract. We provide a nine-valued logic to characterize the models of logic programs under a paraconsistent well-founded semantics with explicit negation $\mathit{WFSX}_{\mathrm{p}}$. We define a truth-functional logic, $\mathcal{NINE}$, based on the bilattice construction of Ginsberg and Fitting. The models identified by $\mathit{WFSX}_{\mathrm{p}}$ are models of logic $\mathcal{NINE}$. We conclude with a discussion on the conditions to obtain an isomorphism between the two definitions, and thereby characterizing $\mathit{WFSX}_{\mathrm{p}}$ model-theoretically.

1 Introduction

One of the main issues in logic programming is the definition of semantics for negation(s). Quite recently, a second form of negation besides the older default negation, was proposed by several authors [12, 8, 11, 21, 13]) providing a mechanism for explicitly declaring the falsity of literals, which was not available before. The importance of extending LP with a second kind of negation $\neg$, has been stressed for use in deductive databases, knowledge representation, and non-monotonic reasoning. Different semantics for extended LPs with $\neg$-negation have appeared (e.g. [8, 18, 13, 22]). The specific generalization for extended programs of well-founded semantics [7], *WFSX*, defined in [13, 2] using "explicit negation", is taken as the base semantics in this paper.

The introduction of explicit negation requires being able to reason with, or at least detect, contradictory knowledge. Indeed, information is not only normally incomplete but contradictory as well. As remarked by [22] there are three main ways of dealing with inconsistent information:

Explosive approach: If the program is contradictory then every formula is derived from it. This corresponds to the usual approach in mathematical logic, and of several semantics for extended logic programs [18, 8, 13, 2].

Belief revision approach: The program is revised in order to regain consistency. This is the view adopted by some authors in the LP community [17, 14, 10, 1, 2]. It does not necessarily require an explicit paraconsistent semantics: the procedural revision operators suffice.

[⋆] We thank Esprit BR project Compulog 2 (no. 6810), and JNICT for their support.

Paraconsistent approach: Accept contradictory information and perform reasoning tasks that take it into account. This is the approach of [4, 19, 22], and the one we will follow in this paper.

The first approach is rather naïve and only makes sense when dealing with mathematical objects. For instance, if we have a large knowledge base being mantained or updated by different agents, it is natural to encounter inconsistencies in the database. Most of the time, this inconsistency is local to some part of the knowledge base and it shouldn't affect other, independent, information. If we adopt the explosive approach, and a single contradiction is found then we must discard the entire knowledge base. This is uneconomical.

Sometimes the contradictory information can be due to a specification error, and we'd like to fix it through debugging. In other situations the information provided is in itself contradictory. In the former, we can use belief revision techniques. In the latter, a paraconsistent deductive mechanism is necessary. Notice however that to perform belief revision we need in any case to detect the inconsistencies and the reasons supporting them. Thus, paraconsistent reasoning is an, at least implicit, intermediate step to attain belief revision.

Since we want to assign meaning to every program we will make use of the paraconsistent version of *WFSX*, $\mathit{WFSX}_{\mathrm{p}}$ which can be found in [1, 2]. The semantics complies with two basic principles: coherence and also a form of introspection. The "coherence principle" of [13, 2] relates the two forms of negation, default and explicit: it stipulates that the latter entails the former, i.e if L $(\neg L)$ is entailed then so is $\mathit{not}\,\neg L$ $(\mathit{not}\,L)$. In other words, coherence requires that if I'm convinced of the truth of a proposition then I must believe (i.e. be weakly convinced about) the truth of the proposition.

The introspection mechanism too provides the derivation of new weak convictions. To express it we need the notion of "doubt": I doubt the truth of L iff I have weak conviction for the falsity of L; I have conviction in the truth of L iff I doubt the weak conviction in the falsity of L. Now we can state the principle of introspective doubt: if I doubt all the bodies of rules for L then I'm weakly convinced of the falsity of L. The joint application of these principles is brought out in example 1.

Example 1. Let P be the extended logic program containing the five rules $\{a; \neg a; b \leftarrow a; c \leftarrow \mathit{not}\,b; d \leftarrow \mathit{not}\,d\}$. It is clear that the model of this program must entail a, $\neg a$ and b; therefore I'm convinced of the truth of a and b (a and b are true) and convinced of the falsity of a ($\neg a$ is true). By applying coherence we should also have $\mathit{not}\,a$, $\mathit{not}\,\neg a$ and $\mathit{not}\,\neg b$, i.e. I'm weakly convinced of the falsity of a ($\mathit{not}\,a$ is true) and weakly convinced of the truth of a and b ($\mathit{not}\,\neg a$ and $\mathit{not}\,\neg b$ are true). By doubt introspection, I believe in the falsity of b ($\mathit{not}\,b$ is true) if I have doubts about the truth of a; and indeed I'm reserved about the truth of a because I believe in the falsity of a, i.e. $\mathit{not}\,a$ is true. Therefore $\mathit{not}\,b$, and thus c, should belong to the model. By the same introspective doubt, I'm weakly convinced of the falsity of c ($\mathit{not}\,c$ is true) if I doubt my belief in the falsity of b, i.e. if I'm convinced of the truth of b; this is so (b is true), therefore $\mathit{not}\,c$

should belong to the model of the program. Regarding literal d, I'm convinced of the truth of d iff I doubt the truth of d, therefore I remain agnostic about my conviction, i.e. d and $\mathit{not}\, d$ are not entailed. On the other hand, I'm weakly convinced of the truth d since I have no rule for $\neg d$.

Therefore we equate conviction in the truth (resp. falsity) of a proposition with L (resp. $\neg L$), and belief with "$\mathit{not}\, \neg$", i.e. belief in the truth (resp. falsity) of a proposition with $\mathit{not}\, \neg L$ (resp. $\mathit{not}\, \neg\neg L = \mathit{not}\, L$).

We assume the reader acquainted with the terminology of extended logic programs, in particular that atoms or explicitly negated atoms are called objective literals, and that default literals are the default negated objective literals. For details see [8, 13, 2]. All definitions and results in this paper are valid only for the ground instances of programs.

We refer the reader to [1] for the alternating fixpoint definition of $\mathit{WFSX}_{\mathrm{p}}$. Different, but equivalent, definitions of $\mathit{WFSX}_{\mathrm{p}}$ also appeared in [16, 15, 2]. In these works the paraconsistent version of WFSX is used only to detect contradictions to be removed. This is not the stance taken here. We wish to analyse the intrinsic properties of the semantics to shed new light on the subject, and also to turn the connectives into truth-functional ones. We provide a nine-valued logic to characterize the models of logic programs explicit negation under $\mathit{WFSX}_{\mathrm{p}}$, based on the bilattice construction of Ginsberg and Fitting. The models identified by $\mathit{WFSX}_{\mathrm{p}}$ are models of logic $\mathcal{NINE}$. We conclude with a discussion on the conditions to obtain an isomorphism between both definitions, thereby characterizing $\mathit{WFSX}_{\mathrm{p}}$ model-theoretically.

2 Model theory for $\mathit{WFSX}_{\mathrm{p}}$

One of the main criticsms to the original definition of well-founded semantics with explicit negation regards its model theory. In [13, 2] the definition of the logical implication operation is not truth functional and literals a and $\neg a$ are viewed almost as separate entities: though $\neg a$ entails a false nothing else follows for other truth values. Furthermore, in some cases the head of a rule is assigned the truth value $\mathbf{f}$ the body $\mathbf{u}$, but the rule is satisfied ($\mathbf{f} \leftarrow \mathbf{u}$). This is due, in such cases, to the coherence principle, and it might be considered awkward in a three-valued logic. In other situations, with the same assignment of values to head and body a rule, and where coherence does not intervene, the rule is unsatisfiable, showing again the non-truth functional character of the model definition used. These problems are illustrated with example 2.

Example 2. Let P be the logic program containing the following rules: $\{a \leftarrow b;\ b \leftarrow \mathit{not}\, b; \neg a\}$. The $WFM(P)$ is $\{\neg a, \mathit{not}\, a, \mathit{not}\, \neg b\}$. According to the notions of interpretation and model of [13, 2], literal b is assigned truth value undefined, a is assigned false, and $\neg a$ true. The first rule of the program is satisfied by this interpretation, even with false head and undefined body. The intuition being that $\neg a$ overrides the undefinedness of a obtained on the basis of the first rule.

The basic mechanisms we want to capture and formalize are illustrated in examples 1 and 2. Literals are entailed with three different degrees of epistemic entrenchment: conviction, weak conviction and undecidedness. Convictions are represented by objective literals and the rule implication sign in extended logic programming propagates them. Weak convictions are drawn by default or result from the mandatory application of the coherence principle.

A by now standard way of generating new logics for reasoning with missing or conflicting information is via Ginsberg's bilattices [9], who generalizes the ideas behind Belnap's four valued logic [3]. We follow Fitting's proposals [6] on how to construct a bilattice.

Definition 1. [6] Given two complete lattices $C = \langle \mathbf{C}, \leq_1 \rangle$ and $D = \langle \mathbf{D}, \leq_2 \rangle$ the structure $\mathcal{B}(C, D) = \langle \mathbf{C} \times \mathbf{D}, \leq_k, \leq_t \rangle$ is a bilattice, with $\langle c_1, d_1 \rangle \leq_k \langle c_2, d_2 \rangle$ iff $c_1 \leq_1 c_2 \wedge d_1 \leq_2 d_2$, and $\langle c_1, d_1 \rangle \leq_t \langle c_2, d_2 \rangle$ iff $c_1 \leq_1 c_2 \wedge d_2 \leq_2 d_1$. To each ordering are associated join and meet operations as usual. Conjunction and disjunction are, respectively, meet and join in the truth-ordering.

The intuition here is that C (D) provides the arguments for (against) believing in the truth of a statement. If $\langle c_1, d_1 \rangle \leq_k \langle c_2, d_2 \rangle$, then situation 2 has more information than 1, i.e. knowledge is increased. If $\langle c_1, d_1 \rangle \leq_t \langle c_2, d_2 \rangle$ then we have more reasons to believe in situation 2 than in 1, because the reasons for believing the statement either increased or the reasons against it are weaker, i.e. $\langle c_2, d_2 \rangle$ is truer than $\langle c_1, d_1 \rangle$.

Next we define our truth-space using the bilattice construction. We have identified the reasons for believing, disbelieving, or abstaining in the truth of a proposition: conviction, weak conviction and undecidedness. Using these degrees we can form two complete lattices, one with the reasons for the truth of a proposition and other with the reasons against it (or for its falsity): *For* $= (\{\} < \{not\, \neg L\} < \{L, not\, \neg L\})$ and *Against* $= (\{\} < \{not\, L\} < \{\neg L, not\, L\})$. The set $\{L\}$ $(\{\neg L\})$ is not an element of lattice *For* (*Against*) because of the coherence principle: If I am convinced of the truth of L $(\neg L)$ then I must believe in L $(\neg L)$, i.e. $not\, \neg L$ $(not\, L)$ should hold. In fact, we can abstract from the lattices *For* and *Against* by defining their isomorphic lattice of opinion. The bilattice $\mathcal{NINE}$ is then constructed from the opinion lattice.

Definition 2. The opinion lattice is the complete lattice $O =< \{none, weak, strong\}, (none < weak < strong) >$. Bilattice $\mathcal{NINE}$ is defined as $\mathcal{B}(O, O)$. To each one of the nine pairs $\langle o_1, o_2 \rangle$ of $\mathcal{NINE}$ we assign it a value as follows:

Pair	Value	Pair	Value	Pair	Value
$\langle none, none \rangle$	$\perp$	$\langle none, strong \rangle$	**f**	$\langle strong, weak \rangle$	**II**
$\langle none, weak \rangle$	**df**	$\langle strong, none \rangle$	**t**	$\langle weak, strong \rangle$	**III**
$\langle weak, none \rangle$	**dt**	$\langle strong, strong \rangle$	**I**	$\langle weak, weak \rangle$	**IV**

By $\perp$ we mean no opinion (or undecidedness or undefinedness), and corresponds to the bottom element of the lattice according to the knowledge ordering. The two extra logical values correspond to default falsity (**df**) and default truth (**dt**). We have also the classical **t** and **f** values and four degrees of contradictory information. Figure 1 is the Hasse diagram of bilattice $\mathcal{NINE}$. In order to ob-

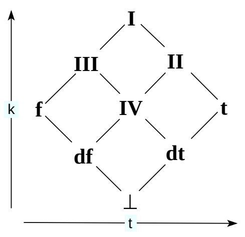

Fig. 1. Logic $\mathcal{NINE}$

tain a correspondence with the paraconsistent semantics for extended logic programs we need to define two forms of negation and the interpretation of the rule implication sign. According to Fitting [6] a negation operator in a (interlaced) bilattice should satisfy the following three properties: (1) $a \leq_k b \Rightarrow \neg a \leq_k \neg b$; (2) $a \leq_t b \Rightarrow \neg b \leq_t \neg a$; (3) $a = \neg\neg a$. If the double negation condition (3) is not verified then the negation operator is said to be weak. The intuition is that negations should reverse truth but preserve knowledge. In what follows we will also use $\mathcal{NINE}$, when no confusion arises, to represent the set of logical values induced by bilattice $\mathcal{NINE}$.

Definition 3. Let $\neg : \mathcal{NINE} \mapsto \mathcal{NINE}$ and *not* $: \mathcal{NINE} \mapsto \mathcal{NINE}$ be the unary operators defined by the following truth-tables.

a	$\perp$	**df**	**dt**	**f**	**t**	**I**	**II**	**III**	**IV**
$\neg a$	$\perp$	**dt**	**df**	**t**	**f**	**I**	**III**	**II**	**IV**
not a	$\perp$	**t**	**f**	**t**	**f**	**I**	**I**	**I**	**I**

Proposition 4. *Operators* $\neg : \mathcal{NINE} \mapsto \mathcal{NINE}$ *and not* $: \mathcal{NINE} \mapsto \mathcal{NINE}$ *are, respectively, a negation and a weak negation.*

The rationale is that $\neg$ finds the degree of conviction for the negation of a proposition, exchanging the roles of what counts for and what counts against. The *not* operator determines if the negation of a proposition is at least believed. We have four cases to consider: there is no conviction in the truth nor falsity of a proposition; there is at least weak conviction in the falsity of a proposition and no conviction in its truth; there is no conviction in the falsity of a proposition and at least weak conviction in its truth; there is at least weak conviction for both the falsity and the truth of a proposition. In the first case we remain undecided. In the second case *not* returns true, and in the third case false. In the fourth and last case we have contradictory information and *not* returns contradiction.

The definition of the rule implication sign is rather simple: it enacts the principle that if we have conviction in the antecedent we must have conviction in the consequent.

Definition 5. Let $\leftarrow: \mathcal{NINE} \times \mathcal{NINE} \mapsto \mathcal{NINE}$ be the binary operator defined by the following rules: Rule implication $a \leftarrow b$ is $\mathbf{f}$ iff b is $\mathbf{I}$, $\mathbf{II}$ or $\mathbf{t}$ but a is none of these truth values; otherwise, $a \leftarrow b$ is $\mathbf{t}$.

Our goal is to define a notion of model which complies with the alternating fixpoint definition of $\mathit{WFSX}_{\mathrm{p}}$. Namely, we would like the paraconsistent well-founded model to be a $\mathcal{NINE}$ model of the program. To attain this we first define as usual the syntax, and the notions of interpretation, valuation function, and model.

Definition 6. Let $\mathcal{L}$ be a set of atomic propositions. The language $\mathcal{L}_9$ is defined inductively as follows: constants $\mathbf{t}$ and $\mathbf{f}$ belong to $\mathcal{L}_9$; if $a \in \mathcal{L}$ then $a \in \mathcal{L}_9$; if $A \in \mathcal{L}_9$ then $\neg A$ and $\mathit{not}\, A$ are in $\mathcal{L}_9$ too; if A and B belong to $\mathcal{L}_9$ then their boolean combinations $A \wedge B$, $A \vee B$ and $A \leftarrow B$ also belong to $\mathcal{L}_9$; nothing else belongs to $\mathcal{L}_9$.

We define a nine valued interpretation as a function assigning to each atomic proposition one of the truth values of $\mathcal{NINE}$. The definition of truth valuation $\hat{I}_9$ wrt to an interpretation I_9 is a function mapping formulae into logical values of $\mathcal{NINE}$, with the logical connectives "$\wedge$", "$\vee$", "$\neg$", "*not*" and "$\leftarrow$" as per definitions 1, 3, and 5. An interpretation is a model of a set of formulae iff there is conviction in each formula of the set.

Definition 7. Let I be a $\mathcal{NINE}$ interpretation. Interpretation I is a $\mathcal{NINE}$ model of a set of formulae $\mathcal{S}$, represented by $I \models_9 S$, iff $\forall_{s \in \mathcal{S}} \hat{I}(s) = \mathbf{I} \vee \hat{I}(s) = \mathbf{II} \vee \hat{I}(s) = \mathbf{t}$

For simplicity, we write $I \models_9 s$, standing for $I \models_9 \{s\}$, where s is an arbitrary formula of $\mathcal{L}_9$. By definition of valuation of implicational formulae, we have for any formula s of $\mathcal{L}_9$ the equivalence $\hat{I}(s \leftarrow \mathbf{t}) = \mathbf{t}$ iff $I \models_9 s$.

The commutative, associative, idempotency and absorption laws of '$\wedge$' and '$\vee$' are inherited from the complete lattice structure of $\mathcal{NINE}$. Remark that property $\mathit{not}\, \mathit{not}\, A = \mathit{not}\, \neg A = \neg \mathit{not}\, A$ holds in $\mathcal{NINE}$, and therefore all possible combinations of the negation operators can reduce to one of the four following cases, where a is an atomic proposition: a, $\mathit{not}\, a$, $\neg a$ or $\mathit{not}\, \neg a$.

The mapping between single literal based interpretations and $\mathcal{NINE}$ truth-values is immediate. The formal mapping aspects can be found in definition 8.

Definition 8. Let I be a set of literals, default or objective, and $\mathcal{L}_I$ the set of atoms of the language of I. The $\mathcal{NINE}$ interpretation $\tau(I)$, with underlying language $\mathcal{L}_I$, is the interpretation where to an atomic proposition a of $\mathcal{L}_I$ is assigned one logical value, as follows:

$\bot$	iff $a\notin I \wedge \neg a\notin I \wedge not\, a\notin I \wedge not\, \neg a\notin I$	**I**	iff $a\in I \wedge \neg a\in I \wedge not\, a\in I \wedge not\, \neg a\in I$
df	iff $a\notin I \wedge \neg a\notin I \wedge not\, a\in I \wedge not\, \neg a\notin I$	**II**	iff $a\in I \wedge \neg a\notin I \wedge not\, a\in I \wedge not\, \neg a\in I$
dt	iff $a\notin I \wedge \neg a\notin I \wedge not\, a\notin I \wedge not\, \neg a\in I$	**III**	iff $a\notin I \wedge \neg a\in I \wedge not\, a\in I \wedge not\, \neg a\in I$
f	iff $a\notin I \wedge \neg a\in I \wedge not\, a\in I \wedge not\, \neg a\notin I$	**IV**	iff $a\notin I \wedge \neg a\notin I \wedge not\, a\in I \wedge not\, \neg a\in I$

Notice that the τ mapping is a bijection among the set of interpretations and the set of $\mathcal{NINE}$ interpretations based on the same set of atomic propositional symbols.

Our first result is the desirable statement relating the paraconsistent partial stable models of a logic program with $\mathcal{NINE}$ models, i.e. the fixpoints of $\Gamma\Gamma_s$. For the definition of Γ and Γ_s operators see [1, 2]. According to theorem 9 we have succeeded in finding in $\mathcal{NINE}$ a model theory for the paraconsistent well-founded semantics with explicit negation.

Theorem 9. *Let P be an extended logic program. If T is a fixpoint of $\Gamma_P\Gamma_{P_s}$ then $I = \tau\,(T \cup not\,\mathcal{H}_P - \Gamma_{P_s})$ is a $\mathcal{NINE}$ model of P, i.e. $I \models_9 P_9$. The theory P_9 is obtained from P by replacing all rules $L_0 \leftarrow L_1, \ldots, L_m, not\, L_{m+1}, \ldots, not\, L_n$ by the $\mathcal{L}_9$ formula $L_0 \leftarrow L_1 \wedge \ldots \wedge L_m \wedge not\, L_{m+1} \wedge \ldots \wedge not\, L_n$*

Example 3. Consider again program P of example 2. Its WFM_p is $\{\neg a, not\, a, not\, \neg b\}$. We obtain its $\mathcal{NINE}$ model by using the τ transformation. To a is assigned the value **f** and to b the value **dt**. As the reader can check, this is a $\mathcal{NINE}$ model of the corresponding P_9 program.

3 Supported Models

The program of example 2 has other $\mathcal{NINE}$ models which do not correspond to any fixpoint of $WFSX_{\mathrm{p}}$. One such model is obtained by assigning **f** to a, and to b the undefined truth value $\bot$. But this model is smaller under the knowledge order than the corresponding model generated by the WFM_p. This raises the question why we infered **dt** for b instead of $\bot$. In general terms, we have an additional mechanism for infering weak conviction, i.e. negation by default. Because we have no conviction in the falsity of b we may weakly believe in its truth. This form of reasoning provides new insights into the stronger conditions that the $\mathcal{NINE}$ models generated by $WFSX_{\mathrm{p}}$ obey.

Definition 10. Let I be a $\mathcal{NINE}$ model. We define relation $I \leadsto F$ as follows, where L is an objective literal and A and B $\mathcal{L}_9$ formulae without occurrences of the rule implication symbol:

$$\begin{array}{lll@{\qquad}lll} I \leadsto L & \text{iff} & I \models_9 not\, L & I \leadsto (A \wedge B) & \text{iff} & (I \leadsto A) \text{ or } (I \leadsto B) \\ I \leadsto not\, L & \text{iff} & I \models_9 L & I \leadsto (A \vee B) & \text{iff} & (I \leadsto A) \text{ and } (I \leadsto B) \end{array}$$

When $I \leadsto F$ we have reasons to doubt of F. The cases on the left express formally the intuitions given in the introduction of the paper. Now we can give a better characterization of the properties obeyed by $WFSX_{\mathrm{p}}$ fixpoints.

Definition 11. Let P_9 be a $\mathcal{NINE}$ theory generated by a logic program P. We say that I is a supported model of P_9 iff the following holds: $I \models_9 L$ iff there is an implication in P_9 with consequent L and antecedent $Body$ such that $I \models_9 Body$; $I \models_9 not\, L$ iff $I \models_9 \neg L$ or for every implication with conclusion L and premise $Body$ we have $I \rightsquigarrow Body$.

According to the above definition, we have conviction in the truth of a literal if we have at least one rule for that literal for which we are convinced of the truth of the body. We believe in the falsity of a literal iff we are convinced of its falsity (coherence principle), or we doubt each body of the rules with that literal in the head (principle of introspective doubt).

Theorem 12. *Let P be an extended logic program and P_9 its corresponding $\mathcal{NINE}$ theory. If T is a fixpoint of $\Gamma_P\Gamma_{P_s}$ then $I = \tau\left(T \cup not\, \mathcal{H}_P - \Gamma_{P_s}\right)$ is a supported model of P_9.*

The converse of the above theorem is not valid, i.e. there are supported models which are not fixpoints of $\Gamma\Gamma_s$. Therefore supported models do not fully characterize the fixpoints of $\Gamma\Gamma_s$, as illustrated in the following example.

Example 4. Consider the following two-rule program $\{a \leftarrow b; b \leftarrow a\}$. The corresponding $\mathcal{NINE}$ theory has four supported models, namely $\{a = \mathbf{dt}, b = \mathbf{dt}\}$, $\{a = \mathbf{t}, b = \mathbf{t}\}$, $\{a = \mathbf{II}, b = \mathbf{II}\}$ and $\{a = \mathbf{IV}, b = \mathbf{IV}\}$. The least model under the knowledge ordering is the first one listed. Mark that the unique fixpoint of the original program corresponds to the latter model.

The problem with example 4 is the "positive loop" between a and b. In general, when programs have no infinite positive dependencies among literals, the supported models are isomorphic to the fixpoints of $WFSX_{\mathrm{p}}$. We first define the positive dependency graph:

Definition 13. Let P be an extended logic program. The vertices of the positive dependency directed graph $Dep^+(P)$ of P are the literals of $\mathcal{H}_P$. For each rule $L \leftarrow L_1, \ldots, L_m, not\, L_{m+1}, \ldots, not\, L_n$ of P there is an edge from node L_i, $1 \leq i \leq m$, to node L. If there are no infinite descending chains in $Dep^+(P)$ we say that the program has no infinite positive recursion.

Theorem 14. *Let P be an extended logic program without infinite positive recursion, if $I = \tau(T \cup not\, F)$ is a supported model of the corresponding P_9 theory then $T \cup not\, F$ is a fixpoint of $\Gamma\Gamma_s$.*

The importance of this result is due to the circumstance that an extended logic program P can be transformed into an equivalent program with no objective literals in the bodies of rules. This means that it is possible to find an equivalent program where the characterization of $WFSX_{\mathrm{p}}$ fixpoints by the supported models holds, i.e. a model-theoretic definition for $WFSX_{\mathrm{p}}$. This program is obtained by iterated application of an appropriate partial evaluation transformation: let $no^+(P)$ be the set of rules in P which do not have positive literals in their bodies. Let $L \leftarrow Body$ be a rule in $no^+(P)$. Substitute every positive occurrences of L in a body of P by the conjunction $Body, not\,\neg L$. Add these new rules to P. Iterate this step till all rules of $no^+(P)$ were substituted, obtaining program P'. Repeat the whole process with program P' till no new rules are added, resulting program P_f. Program $no^+(P_f)$ is equivalent to the original program P. This bottom-up construction is similar to Brass and Dix's residual program computation [5].

Example 5. Consider the program of example 4. By applying the above process we obtain as a result the empty program, which has the unique supported model $\{a = \mathbf{IV}, b = \mathbf{IV}\}$, corresponding to its paraconsistent well-founded model.

4 Conclusions

We have provided a new truth-functional nine-valued logic which furnishes an elegant model-theory for *WFSX* and *WFSX*$_\mathrm{p}$. In this logic, a bilattice, it is possible to define, by a truth table, the meaning of the "*not*" and "$\neg$" negation operators, besides the usual boolean connectives "$\vee$", "$\wedge$" and "$\leftarrow$". We were able to prove that every paraconsistent partial stable model is a model of the theory readily obtained from the program, under our $\mathcal{NINE}$ logic. Then we introduced the notion of supported model in order to establish a one-to-one correspondence between supported models and paraconsistent partial stable models. We have shown that this is not possible in general, but we have identified a class of the programs (the ones which do not have infinite positive dependencies) where the "equivalence" holds. Fortunately, every extended logic program has an equivalent one in this stricter class of programs, and therefore we can characterize model-theoretically the semantics of any extended logic program. It remains to be settled if such one-to-one characterization is possible without using this other equivalent, but distinct, program. It would also be very interesting if the main ideas underlying our *WFSX*$_\mathrm{p}$ semantics could be generalized for arbitrary interlaced bilattices. A similar nine-valued logic was independently proposed in [20] to formalize a semantics for disjunctive extended logic programs, but with different interpretations of the negation operators, and based on an Answer-Set like semantics. There are results showing the natural embedding of the semantics [4, 18, 19, 22] in *WFSX*$_\mathrm{p}$, which will be the subject of a forthcoming paper.

References

1. J. J. Alferes, C. V. Damásio, and L. M. Pereira. A logic programming system for non-monotonic reasoning. *Journal of Automated Reasoning*, Special Issue on Implementation of NonMonotonic Reasoning(14):93–147, 1995.
2. J. J. Alferes and L. M. Pereira. *Reasoning with Logic Programming.* Springer–Verlag, 1995. In print.
3. N. D. Belnap. A useful four-valued logic. In G. Epstein and J. M. Dunn, editors, *Modern Uses of Many-valued Logic*, pages 8–37. Reidel, 1977.

4. H. A. Blair and V. S. Subrahmanian. Paraconsistent logic programming. *Theoretical Computer Science*, 68:135–154, 1989.
5. S. Brass and J. Dix. A disjunctive semantics based on unfolding and bottom-up evaluation. In *Proc.* IFIP '94-Congress, Workshop FG2: Disjunctive Logic Programming and Disjunctive Databases, pages 83–91. Springer, 1994.
6. M. Fitting. Bilattices and the semantics of logic programming. *Journal of Logic Programming*, 11:91–116, 1991.
7. A. V. Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.
8. M. Gelfond and V. Lifschitz. Logic programs with classical negation. In Warren and Szeredi, editors, *7th ICLP*, pages 579–597. MIT Press, 1990.
9. M. L. Ginsberg. Multivalued logics: a uniform approach to reasoning in artificial intelligence. *Computational Intelligence*, 4:265–316, 1988.
10. C. M. Jonker and C. Witteveen. Revision by expansion. In G. Lakemeyer and B. Nebel, editors, *Proceedings ECAI'92 Workshop on Theoretical Foundations of Knowledge Representation*, pages 40–44. ECAI'92 Press, 1992.
11. R. Kowalski and F. Sadri. Logic programs with exceptions. In Warren and Szeredi, editors, *7th ICLP*. MIT Press, 1990.
12. D. Pearce and G. Wagner. Reasoning with negative information I: Strong negation in logic programs. In *Language, Knowledge and Intentionality*, pages 430–453. Acta Philosophica Fennica 49, 1990.
13. L. M. Pereira and J. J. Alferes. Well founded semantics for logic programs with explicit negation. In B. Neumann, editor, *Proc. ECAI*, pages 102–106. John Wiley & Sons, 1992.
14. L. M. Pereira, J. J. Alferes, and J. N. Aparício. Contradiction Removal within Well Founded Semantics. In A. Nerode, W. Marek, and V. S. Subrahmanian, editors, *LPNMR'91*, pages 105–119. MIT Press, 1991.
15. L. M. Pereira, J. J. Alferes, and J. N. Aparício. Contradiction removal semantics with explicit negation. In M. Masuch and L. Pólos, editors, *Knowledge Representation and Reasoning Under Uncertainty*, volume 808 of *LNAI*, pages 91–106. Springer-Verlag, 1994.
16. L. M. Pereira, J. N. Aparício, and J. J. Alferes. Non–monotonic reasoning with logic programming. *Journal of Logic Programming. Special issue on Nonmonotonic reasoning*, 17(2, 3 & 4):227–263, 1993.
17. S. G. Pimentel and W. L. Rodi. Belief revision and paraconsistency in a logic programming framework. In A. Nerode, W. Marek, and V. S. Subrahmanian, editors, *LPNMR'91*, pages 228–242. MIT Press, 1991.
18. T. Przymusinski. Extended stable semantics for normal and disjunctive programs. In Warren and Szeredi, editors, *7th ICLP*, pages 459–477. MIT Press, 1990.
19. C. Sakama. Extended well–founded semantics for paraconsistent logic programs. In *Fifth Generation Computer Systems*, pages 592–599. ICOT, 1992.
20. C. Sakama and K. Inoue. Paraconsistent stable semantics for extended disjunctive programs. *Journal of Logic and Computation*, 5(3), 1995.
21. G. Wagner. A database needs two kinds of negation. In B. Thalheim, J. Demetrovics, and H.-D. Gerhardt, editors, *Mathematical Foundations of Database Systems*, pages 357–371. LNCS 495, Springer–Verlag, 1991.
22. G. Wagner. Vivid logic: Knowledge-based reasoning with two kinds of negation. *LNAI*, 764, 1994.

This article was processed using the LaTeX macro package with LLNCS style