

Paraconsistent Semantics for Extended Fuzzy Logic Programs via Approximation Fixpoint Theory

Pascal Kettmann¹, Hannes Strass^{1,4}, Jesse Heyninck^{2,3}, and Jeroen Spaans²

¹ TU Dresden, Germany {pascal.kettmann,hannes.strass}@tu-dresden.de

² Open Universiteit, The Netherlands {jesse.heyninck,jeroen.spaans}@ou.nl

³ University of Cape Town, South Africa

⁴ ScaDS.AI Center for Scalable Data Analytics and AI, Dresden/Leipzig, Germany

Abstract In logic programming, negation can be interpreted in various ways. Probably best known is the concept of “negation as failure”, where “*not p*” is true if we have no evidence for *p*. On the other hand, strong negation requires that we have evidence for *p* being false. Defining semantics for logic programs containing both kinds of negation is a challenging task, and this becomes even more challenging when combining this with other extensions of logic programming, e.g. fuzziness. In this work, we use the framework of approximating fixpoint theory to formulate well-behaved semantics for fuzzy logic programs containing both “by-failure” and strong negation. We show that this framework generalizes several existing semantics as well as giving rise to a host of new semantics.

1 Introduction

Logic programs (LPs) are among the oldest rule-based formalisms within the field of artificial intelligence [25]. While originally used for declarative programming, their modern incarnation in answer set programming [6,15] enjoys widespread practical success for declarative problem solving, while using the same basic constructs at the core of the language. Researchers noted early on that the inherent restriction to two truth values limits applicability, and proposed generalizations from *crisp* (two-valued) to *fuzzy* (many-valued) logics – for resolution in general [26], and for (definite) logic programs in particular [35,37,19].

Naturally, “weak” negation (indicating absence of provability and thus also known as negation by failure [7]) has been combined with fuzziness, leading to *normal* fuzzy logic programs offering various semantics, including stable models [40,8] and well-founded semantics [27]. On the other hand, crisp normal LPs have also been extended with “strong” negation (indicating established falsity [16]), leading to *extended* logic programs with both kinds of negation. While the original proposal of semantics for extended LPs required answer sets (stable models) of programs to be consistent, that is, disallowed the presence

of an atom and its classical negation in the model, it did not take long until *paraconsistent* semantics were proposed for extended LPs – again both well-founded semantics [32] and stable model semantics [33]. The idea of paraconsistent reasoning is that inconsistency is “allowed” to occur in a model, but is “locally contained” and thus does not lead to the usual *ex falso quodlibet* issues of classical logic [4], i.e. it avoids trivializing logical reasoning. And while *extended fuzzy* LPs with stable model semantics do exist, they either do not offer paraconsistent reasoning at all [31,21], or do not offer it in full generality: some works [2,11,1] base their semantics on the *coherence principle*, which requires them to make restrictive assumptions on the negation functions that can possibly be employed.

The common theme in all of these works on various extensions of normal LPs is their piecemeal nature, the high amount of manual labour involved, and the sheer repetitiveness with which authors took the same ideas, such as the monotone one-step consequence operator of van Emden and Kowalski [13] and the reduct of Gelfond and Lifschitz [16], and manually adapted them to cover the ever next extension. The result is a scattered landscape with no uniform syntax, semantics with the same name but formally unclear relationships, and varying assumptions on possible sets of truth values and logical connectives.

In this paper, we unify a considerable portion of the existing work in this area. First, we employ a syntax that covers major recurring features of the literature on extended fuzzy logic programming. Second, and more importantly, we define an operator that captures a considerable number of existing semantics. This is possible because we employ the powerful framework of *approximation fixpoint theory* (AFT) [12]. AFT reduces the amount of manual labour required to a minimum, as it already incorporates – within its standard definitions – ideas that previous authors had to rework explicitly in their respective settings, such as the program reduct of Gelfond and Lifschitz [16]. More concretely, we show how our *single* operator reconstructs the semantics of Sakama [32], Sakama and Inoue [33], Saad [31], and Cornejo et al. [9], while also giving rise to new semantics, in particular the first stable model and well-founded semantics offering paraconsistent reasoning for extended fuzzy LPs that allows for non-involutive negations (where double negation is not always identity, e.g. Gödel negation).

The rest of the paper proceeds as follows. In the next section, we give the necessary background. Section 3 introduces the syntax of the language we use, while Section 4 defines our main operator. The following two sections reconstruct existing semantics of extended (fuzzy) logic programs, with Section 7 providing additional results beyond mere reconstruction. We finally discuss avenues for future work. An extended version of this paper containing more details and proofs of all results is available as a preprint at <https://arxiv.org/abs/2605.05286>.

2 Background

2.1 Approximation Fixpoint Theory (AFT)

A particular way of defining semantics for programming languages is to associate a program P with a *transformation* operator T_P that transforms a given input-

output relation R into an updated relation $T_P(R)$. In this view, the semantics of program P is given by the least fixpoint of operator T_P . Van Emden and Kowalski [13] applied this way of defining semantics to logic programming, developing the first one-step logical consequence operator and employing its monotonicity in conjunction with Tarski's fixpoint theorem [36] to define the semantics of definite logic programs with potentially recursive predicate definitions.

More formally, an *operator* is a (total) function $O: V \rightarrow V$ on a partially ordered set $(V, \leq)$. Here, we even require $(V, \leq)$ to be a *complete lattice*, that is, we require that every subset $S \subseteq V$ has a *least upper bound (lub)* $\bigvee S \in V$ and *greatest lower bound (glb)* $\bigwedge S \in V$; this entails that $(V, \leq)$ has both a least element $0 = \bigwedge V = \bigvee \emptyset$ and a greatest element $1 = \bigvee V = \bigwedge \emptyset$. An operator $O: V \rightarrow V$ is *monotone* iff $x \leq y$ implies $O(x) \leq O(y)$, and *antimonotone* iff $x \leq y$ implies $O(y) \leq O(x)$. An element $x \in V$ with $O(x) = x$ is a *fixpoint* of O ; an $x \in V$ with $O(x) \leq x$ is a *pre-fixpoint* of O ; an $x \in V$ with $x \leq O(x)$ is a *post-fixpoint* of O .

A fundamental result in lattice theory, Tarski's fixpoint theorem [36], states that whenever O is a monotone operator on a complete lattice $(V, \leq)$, the set $\{x \in V \mid O(x) = x\}$ of its fixpoints forms a complete lattice itself, and so has a least element $lfp(O)$ and a greatest element $gfp(O)$. While this result is immensely useful, it is not constructive; however, constructive versions exist and tell us how to actually construct least fixpoints [28, 10]: To this end, one defines, for any $x \in V$, the iterated (transfinite) application of O to x via $O^0(x) := x$, $O^{\alpha+1}(x) := O(O^\alpha(x))$ for successor ordinals, and $O^\gamma(x) := \bigvee \{O^\alpha(x) \mid \alpha < \gamma\}$ for limit ordinals. It is then known that (for monotone O) there exists an ordinal α such that $O^\alpha(0) = lfp(O)$ [5]; by duality, we have $gfp(O) = O^\kappa(1)$ for some κ .

In order to apply Tarski's fixpoint theorem [36] (or its constructive versions), it is thus necessary to have an operator that is monotone, a condition that is no longer fulfilled when moving from *definite* to *normal* logic programs, i.e., when allowing weak negation in rule bodies. To still have a useful fixpoint theory in the case of non-monotone operators, Denecker, Marek, and Truszczyński (DMT) [12] fundamentally generalized the theory underlying the fixpoint-based approach to semantics, thereby founding what is now known as *approximation fixpoint theory*. Its underlying idea is that when an operator of interest does not have properties that guarantee the existence of fixpoints, one can *approximate* this operator in a more fine-grained algebraic structure where fixpoint existence can be guaranteed.

Formally – following ideas of Belnap [4], Ginsberg [17], and Fitting [14] – DMT [12] moved from a complete lattice $(V, \leq)$ to its associated *bilattice* on the set $V^2 := V \times V$; where elements of V correspond to interpretations, the pairs contained in V^2 correspond to *approximations* of such interpretations. More technically, a pair $(x, y) \in V^2$ approximates all $z \in V$ with $x \leq z \leq y$. A pair $(x, y) \in V^2$ is called *ordered* iff $x \leq y$, in other words, iff the interval $[x, y] := \{z \in V \mid x \leq z \leq y\}$ is non-empty;⁵ a pair (x, x) is called *exact*.

There are two natural orderings on V^2 : (1) the *truth ordering* extending the (truth) lattice ordering $\leq$, $(x_1, x_2) \leq_t (y_1, y_2) :\iff x_1 \leq y_1$ and $x_2 \leq y_2$, and (2)

⁵ Ordered pairs are called “consistent” in the AFT literature [12]. In this paper, we will use “consistent” in a logical sense and want to avoid confusion.

the *precision ordering* $(x_1, x_2) \leq_p (y_1, y_2) :\iff x_1 \leq y_1 \text{ and } y_2 \leq x_2$, comparing intervals by precision of approximation. The lattice operations induced by $\leq_t$ are glb $\wedge_t$ with $(x_1, x_2) \wedge_t (y_1, y_2) := (x_1 \wedge y_1, x_2 \wedge y_2)$ and lub $\vee_t$ with $(x_1, x_2) \vee_t (y_1, y_2) := (x_1 \vee y_1, x_2 \vee y_2)$.

Just as pairs of V^2 approximate elements of V , a foundational idea of DMT [12] was that operators $O: V \rightarrow V$ can be approximated by operators on V^2 : An *approximator* is an operator $\mathcal{A}: V^2 \rightarrow V^2$ that is (a) $\leq_p$ -monotone and (b) maps exact pairs to exact pairs. By condition (b), we say that $\mathcal{A}$ *approximates* the operator $O: V \rightarrow V$ with $O(x) = \mathcal{A}(x, x)_1$ (where $(x, y)_1 = x$ and $(x, y)_2 = y$); by condition (a), $\mathcal{A}$ is guaranteed to possess ($\leq_p$ -least) fixpoints. An approximator $\mathcal{A}$ is *symmetric* iff for all $(x, y) \in V^2$, we have $\mathcal{A}(x, y)_2 = \mathcal{A}(y, x)_1$; thus symmetry allows one to specify $\mathcal{A}$ by giving only $\mathcal{A}(\cdot, \cdot)_1$.

The main contribution of DMT [12] was the association of the *stable approximator* $\mathcal{A}^{st}$ to an approximator $\mathcal{A}$: for a complete lattice $(V, \leq)$ and an approximator $\mathcal{A}: V^2 \rightarrow V^2$, define the *stable approximator for $\mathcal{A}$* as $\mathcal{A}^{st}: V^2 \rightarrow V^2$ by $\mathcal{A}^{st}(x, y) := (\text{lfp}(\mathcal{A}(\cdot, y)_1), \text{lfp}(\mathcal{A}(x, \cdot)_2))$, where $\mathcal{A}(\cdot, y)_1$ (resp. $\mathcal{A}(x, \cdot)_2$) denotes the operator that maps any $z \in V$ to $\mathcal{A}(z, y)_1$ (resp. to $\mathcal{A}(x, z)_2$). Among other results, DMT [12] also showed that this construction is well-defined because both relevant operators $\mathcal{A}(\cdot, y)_1: V \rightarrow V$ and $\mathcal{A}(x, \cdot)_2: V \rightarrow V$ are $\leq$ -monotone (as $\mathcal{A}$ is an approximator) and thus possess least fixpoints. Furthermore, any approximator maps ordered pairs to ordered pairs, and the stable approximator $\mathcal{A}^{st}$ maps ordered post-fixpoints of $\mathcal{A}$ to ordered pairs. Consequently, not only is $\text{lfp}(\mathcal{A})$ ordered (and called the *Kripke-Kleene fixpoint* of $\mathcal{A}$), but so is $\text{lfp}(\mathcal{A}^{st})$, the *well-founded fixpoint* of $\mathcal{A}$. Moreover, a fixpoint (x, x) of $\mathcal{A}^{st}$, called a *stable fixpoint* of $\mathcal{A}$, always induces a $\leq$ -minimal fixpoint x of O (when $\mathcal{A}$ approximates O). The term “stable” is not coincidental: When using (the normal logic program version of) the transformation operator T_P [13] as O and defining a suitable approximator $\mathcal{A}$ (see Section 4 for a generalization of such an approximator), then the exact fixpoints of $\mathcal{A}^{st}$ correspond one-to-one to the stable models of P [12].

2.2 Fuzzy Logic Programming

Generalizing *crisp* logic programming over $V = \{0, 1\}$, *fuzzy* logic programming allows more general sets V of truth values, with a prominent example being $V = [0, 1]$, the real unit interval [26]. This necessitates also generalizing the syntax of rule bodies from mere conjunctions of atoms as in the crisp case to fuzzy logics’ richer offering of connectives [20]. Formally, given a complete lattice $(V, \leq)$ of truth values, a set Π of propositional atoms, and a family $\mathcal{F}$ of fuzzy connectives (where each $f \in \mathcal{F}$ has an associated *arity* $n \in \mathbb{N}$ denoted by $f^{(n)}$ when important), designating a finitely representable subset $V' \subseteq V$ of truth values that can be explicitly mentioned in programs, the syntax of a *positive fuzzy formula* (1) and a *positive fuzzy logic program rule* (2) is defined via

$$\varphi ::= c \mid p \mid f(\varphi, \dots, \varphi) \quad (1) \qquad p \overset{\vartheta}{\leftarrow} \mathcal{B} \quad (2)$$

where $c \in V'$, $p \in \Pi$, and $f \in \mathcal{F}$, and where $p \in \Pi$, $\vartheta \in V'$, $\leftarrow$ is an impicator, and $\mathcal{B}$ is a positive fuzzy formula. The atom p in (2) is called the *head* of the

rule, $\mathcal{B}$ is called its *body*, and ϑ its *weight* (intuitively indicating the degree of truth assigned to the rule). A *positive fuzzy logic program* (pFLP) P is then a finite set of rules (2).

Every connective $f^{(n)} \in \mathcal{F}$ has an associated truth function $f^* : V^n \rightarrow V$ on the lattice $(V, \leq)$ of truth values; we require that each $f \in \mathcal{F}$ that occurs in a rule body is monotone in each argument, and each implicator $\leftarrow^*$ is monotone in the first and antimonotone in the second argument.

For a complete lattice $(V, \leq)$ of truth values and Π a set of atoms, a *non-paraconsistent (fuzzy) interpretation* is a total function $K : \Pi \rightarrow V$. The set of all non-paraconsistent interpretations is denoted by $\mathcal{K}$ and again forms a complete lattice with the pointwise lifting of $\leq$. The evaluation of atoms by an interpretation K extends to arbitrary formulas φ via structural induction, where we denote the truth value of φ under K by $[K](\varphi)$: for connective $f^{(n)} \in \mathcal{F}$, we define $[K](f(\varphi_1, \dots, \varphi_n)) := f^*([K](\varphi_1), \dots, [K](\varphi_n))$ (and in the base case clearly $[K](p) := K(p)$). Given an interpretation $K \in \mathcal{K}$, a weighted rule $p \xleftarrow{\vartheta} \mathcal{B}$ is *satisfied* by K iff $\vartheta \leq K(p) \leftarrow^* [K](\mathcal{B})$. An interpretation K is a *model* of the program P iff it satisfies all rules in P . Intuitively, K satisfies a rule if it makes the rule “at least as true” as its weight ϑ – the typical notion of fuzzy modelhood.

To guarantee soundness of inference, fuzzy logic programming usually employs pairs of implicators $\leftarrow^*$ and conjunctors $\wedge$ that “belong together”: formally, a pair $(\leftarrow^*, \wedge)$ is called an *adjoint pair* [29] iff for all $x, y, z \in V$ of a partially ordered set $(V, \leq)$, $x \wedge y \leq z$ iff $y \leq z \leftarrow^* x$. In view of adjoint pairs, satisfaction of a rule $p \xleftarrow{\vartheta} \mathcal{B}$ by an interpretation K can equivalently be stated as $\vartheta \wedge [K](\mathcal{B}) \leq K(p)$. For one, this allows one to define a fuzzy generalization of the one-step consequence operator given by van Emden and Kowalski [13]: For pFLP P , operator $T_P : \mathcal{K} \rightarrow \mathcal{K}$ [41, 29] is defined by setting, for each $p \in \Pi$,

$$T_P(K)(p) := \bigvee \left\{ \vartheta \wedge [K](\mathcal{B}) \mid p \xleftarrow{\vartheta} \mathcal{B} \in P \right\} \quad (3)$$

The semantics of a pFLP P can then be characterized by the pre-fixpoints of T_P [41, Theorem 2.2]: interpretation $K \in \mathcal{K}$ is a model of P iff $T_P(K) \leq K$, generalizing a classical result by van Emden and Kowalski [13]. An equally useful consequence of adjoint pairs is that they allow us to “move” weights of rules into the body, i.e. writing (2) equivalently as its *weight-free* version $p \leftarrow \vartheta \wedge \mathcal{B}$ (using the $\wedge$ for which $(\leftarrow^*, \wedge)$ is an adjoint pair). As this allows for a simpler notation without sacrificing expressiveness (e.g. the RHS of (3) simplifies to $\bigvee \{ [K](\mathcal{B}) \mid p \leftarrow \mathcal{B} \in P \}$), in what follows we consider only weight-free rules (also for future extensions, i.e. normal and extended rules).

2.3 Normal Fuzzy Logic Programs

Syntactically, normal FLPs (nFLPs) allow (in their bodies) *normal* fuzzy formulas, that are built by extending (1) with the alternative $\sim p$, where $\sim$ is “negation as failure”. The remaining notions for positive formulas, rules, and programs carry

over to the normal case as expected; in particular, the evaluation of a normal fuzzy formula φ under $K \in \mathcal{K}$ has the additional inductive case $[K](\sim\varphi) := \sim[K](\varphi)$. This directly extends the one-step consequence operator (3) to the case of *normal* FLPs, with models of P and pre-fixpoints of T_P still being in one-to-one correspondence. Alas, as in the crisp case, the occurrence of $\sim$ leads to T_P no longer being necessarily monotone, whence several authors generalized (crisp) semantics to the fuzzy case [27,21,8], by hand-crafting the definitions.

Recently, we [23] have presented an approximator $\mathcal{A}_P$ of T_P for the case of nFLPs, and showed that this one approximator (via the standard definitions of approximation fixpoint theory) uniformly reconstructs several hand-crafted semantics: the well-founded fixpoint of $\mathcal{A}_P$ coincides with the fuzzy well-founded semantics defined by Loyer and Straccia [27]; the exact stable fixpoints of $\mathcal{A}_P$ coincide with the fuzzy stable models defined by Cornejo et al. [8]. The operator we will present in Section 4 is a generalization of $\mathcal{A}_P$. In our previous work [23], we have not explicitly discussed the work of Janssen et al. [21]; however, as the latter base their semantics on the definitions of Medina et al. [29], it follows that the semantics used by Janssen et al. [21] is also covered by our $\mathcal{A}_P$ [23].

3 Extended Fuzzy Logic Programs

In this paper, we are interested in *extended* fuzzy logic programs. Firstly, “extended” means that we allow two kinds of negation – *weak (default) negation* $\sim$, which captures the absence of a proof, and *strong (classical) negation* $\neg$, which expresses explicit falsity. Secondly, we abstract away from concrete sets V of truth values, e.g. $V = \{0, 1\}$ in the Boolean case, or $V = [0, 1]$ in the case of fuzzy LPs [41], but merely require the set V of truth values to be equipped with a (possibly partial) truth ordering $\leq$ such that $(V, \leq)$ forms a complete lattice.

Syntactically, the move from normal to *extended* FLPs sees us replacing atoms with literals across definitions. Based on a set Π of propositional atoms, we define the set of *literals* as $\Pi^\pm := \Pi \cup \Pi^-$, where Π is the set of *positive* literals and $\Pi^- := \{\neg p \mid p \in \Pi\}$ the set of *negative* literals.

Definition 1. Let $(V, \leq)$ be a complete lattice of truth values with $V' \subseteq V$ computable and Π be a set of atoms. With $c \in V'$, $\ell \in \Pi^\pm$, and $f \in \mathcal{F}$, an extended fuzzy formula φ is of the form

$$\varphi ::= c \mid \ell \mid \sim\ell \mid f(\varphi, \dots, \varphi) \quad (4)$$

and an extended fuzzy logic program rule is of the form $\ell \leftarrow \mathcal{B}$, where, differently from (1) and (2), $\ell \in \Pi^\pm$, and $\mathcal{B}$ is an extended fuzzy formula. An extended fuzzy logic program (eFLP) P is then a finite set of (extended fuzzy) rules.

For the negators $\neg$ and $\sim$, the corresponding truth functions $\neg^\cdot$ and $\sim^\cdot$ must (1) be antimonotone on $(V, \leq)$; and (2) behave classically on $0, 1 \in V$, that is, must satisfy $\neg^\cdot 0 = 1$ and $\neg^\cdot 1 = 0$ (and equivalently for $\sim^\cdot$). Our model theory is motivated by the foundational insights deriving from (a) paraconsistent semantics

for *crisp extended* logic programs (using $V = \{0, 1\}$) [32,33], and (b) our previous treatment of *fuzzy normal* LPs in approximation fixpoint theory [23].

Definition 2. A (paraconsistent fuzzy) interpretation is a function $I: \Pi \rightarrow V^2$; we denote the set of all interpretations by $\mathcal{I}$. A pair $(v_1, v_2) \in V^2$ is consistent iff $v_1 \leq \neg v_2$; an $I \in \mathcal{I}$ is consistent iff $I(p)$ is consistent for all $p \in \Pi$.

For an extended fuzzy formula φ and a paraconsistent fuzzy interpretation $I \in \mathcal{I}$, we inductively define $\widehat{I}(\varphi)$ (the evaluation of φ by I), with base cases $\widehat{I}(c) := c$ (for $c \in V'$), $\widehat{I}(p) := I(p)_1$ and $\widehat{I}(\neg p) := I(p)_2$ (for $p \in \Pi$), and $\widehat{I}(\sim \ell) := \sim \widehat{I}(\ell)$ (for $\ell \in \Pi^\pm$); and $\widehat{I}(f(\varphi_1, \dots, \varphi_n)) := f(\widehat{I}(\varphi_1), \dots, \widehat{I}(\varphi_n))$ for any $f^{(n)} \in \mathcal{F}$ in the inductive case.

Intuitively, for interpretation I and atom $p \in \Pi$, the value $I(p)_1$ expresses how *true* p is; similarly, the value $I(p)_2$ expresses how *false* p is. This representation generalizes Belnap's four-valued logic [4], which takes into account information about truth and falsity of atoms independently of each other. A consistent I gives an interval $[I(p)_1, \neg I(p)_2]$ of possible truth values for every $p \in \Pi$. In the special case of fuzzy logic with $\neg x = 1 - x$, we obtain that (x_1, x_2) is consistent iff $x_1 \leq 1 - x_2$ iff $x_1 + x_2 \leq 1$. Note that $\neg$ is only used to determine consistency, but not for evaluating formulas; also note that $\widehat{I}(\varphi) \in V$ for any $I \in \mathcal{I}$ and formula φ : that is, while interpretations assign *pairs* of truth values to atoms (one for truth and one for falsity), the evaluation of a formula is a *single* truth value. The truth ordering $\leq_t$ on pairs of truth values extends naturally (pointwise) to interpretations: Formally, for $I, J \in \mathcal{I}$ we have $\text{glb } \sqcap$ with $I \sqcap J := \{p \mapsto I(p) \wedge_t J(p) \mid p \in \Pi\}$, and $\text{lub } \sqcup$ with $I \sqcup J := \{p \mapsto I(p) \vee_t J(p) \mid p \in \Pi\}$; the binary notation carries over to arbitrary subsets $\mathcal{J} \subseteq \mathcal{I}$, viz. $\sqcap \mathcal{J}$ and $\sqcup \mathcal{J}$.

A paraconsistent fuzzy interpretation $I \in \mathcal{I}$ is a *model* of an eFLP rule $\ell \leftarrow \mathcal{B}$ iff $\widehat{I}(\mathcal{B}) \leq \widehat{I}(\ell)$. I is a *model* of an eFLP P if it is a model of all rules in P .

Example 1. Consider $V = \{0, 1\}$ and the two eFLPs $P_1 = \{\neg q \leftarrow \sim p, p \leftarrow p\}$ and $P_2 = \{\neg q \leftarrow \neg p, p \leftarrow p\}$. The $\leq_t$ -least (and intended) model of P_1 is interpretation $I_1 = \{p \mapsto (0, 0), q \mapsto (0, 1)\}$ where p is false (as it cannot be derived in a well-founded way), thus $\sim p$ is true and by the first rule $\neg q$ is true, while q is false as there is no rule for it. In contrast, the least model of P_2 is given by $I_2 = \{p \mapsto (0, 0), q \mapsto (0, 0)\}$ where all literals are false.

The first step in our reconstruction is defining a consequence operator that characterizes this model semantics.

Definition 3. Let P be an extended fuzzy logic program. Its associated immediate consequence operator T_P is given by $T_P(I)(p) := (T_P^+(I)(p), T_P^-(I)(p))$ with

$$T_P^+(I)(p) := \bigvee \left\{ \widehat{I}(\mathcal{B}) \mid p \leftarrow \mathcal{B} \in P \right\} \text{ and } T_P^-(I)(p) := \bigvee \left\{ \widehat{I}(\mathcal{B}) \mid \neg p \leftarrow \mathcal{B} \in P \right\}$$

T_P is still a van-Emden-Kowalski T_P -operator, i.e. exactly characterizes the models through its prefixpoints.

Proposition 1. For any eFLP P and $I \in \mathcal{I}$, I is a model of $P \iff T_P(I) \leq_t I$.

For formulas not containing default negation $\sim$, evaluation of formulas behaves monotonically with respect to the relevant truth orderings. This means that for $\sim$ -free extended logic programs, the consequence operator is $\leq_t$ -monotone. This will be useful in later reconstructions of stable model semantics, which are based on a reduct program not containing default negation.

Proposition 2. *For a $\sim$ -free eFLP P , $I, J \in \mathcal{I}$: $I \leq_t J \implies T_P(I) \leq_t T_P(J)$.*

While this result on T_P is useful for later reconstructions, it does not provide information on programs with *both* kinds of negation. There, as in the case of nFLPs, it is clear that T_P is not necessarily $\leq_t$ -monotone and therefore cannot be used to define a (least-fixpoint) semantics. Approximation fixpoint theory comes to the rescue here: in the next section, we will define an approximator for T_P and employ the machinery of AFT to obtain several semantics at once.

4 An Approximator for Extended Fuzzy LPs

We now define an approximator of the immediate consequence operator for the full language of eFLPs with weak negation. The relevant lattice-theoretic structure for the immediate consequence operator, i.e. the operator we approximate, is $(\mathcal{I}, \leq_t)$. As is customary for AFT, this means that our approximator will operate on the bilattice based on $(\mathcal{I}, \leq_t)$, which we denote by $(\mathcal{I}^2, \leq_p)$ and refer to as the (precision) *tetralattice*.⁶ More formally, for two pairs $(L, U), (M, T) \in \mathcal{I}^2$ we have $(L, U) \leq_p (M, T) :\iff L \leq_t M$ and $T \leq_t U$. The least element of $(\mathcal{I}^2, \leq_p)$ is the pair $(\mathbf{0}, \mathbf{1})$, where $\mathbf{0} := \{p \mapsto (0, 0) \mid p \in \Pi\}$ and $\mathbf{1} := \{p \mapsto (1, 1) \mid p \in \Pi\}$ are the least and greatest element of $(\mathcal{I}, \leq_t)$, respectively.

Definition 4. *For eFLP P , define symmetric approximator $\mathcal{T}_P: \mathcal{I}^2 \rightarrow \mathcal{I}^2$ with*

$$\begin{aligned} \mathcal{T}_P(L, U) &:= (\mathcal{T}_P(L, U)_1, \mathcal{T}_P(U, L)_1) \\ \mathcal{T}_P(L, U)_1(p) &:= (\mathcal{T}_P^+(L, U)_1(p), \mathcal{T}_P^-(L, U)_1(p)) \\ \mathcal{T}_P^+(L, U)_1(p) &:= \bigvee \left\{ \widehat{\widehat{(L, U)}}(\mathcal{B}) \mid p \leftarrow \mathcal{B} \in P \right\} \\ \mathcal{T}_P^-(L, U)_1(p) &:= \bigvee \left\{ \widehat{\widehat{(L, U)}}(\mathcal{B}) \mid \neg p \leftarrow \mathcal{B} \in P \right\} \end{aligned}$$

where by induction $\widehat{\widehat{(L, U)}}(\ell) := \widehat{L}(\ell)$ and $\widehat{\widehat{(L, U)}}(\sim \ell) := \sim \widehat{U}(\ell)$ for $\ell \in \Pi^\pm$, and $\widehat{\widehat{(L, U)}}(f(\varphi_1, \dots, \varphi_n)) := f(\widehat{\widehat{(L, U)}}(\varphi_1), \dots, \widehat{\widehat{(L, U)}}(\varphi_n))$ for any $f^{(n)} \in \mathcal{F}$.

Note that again, $\widehat{\widehat{(L, U)}}(\varphi) \in V$, so formula evaluation even with elements of the tetralattice yields a single truth value. We look at an example to illustrate the operator, with special attention to the treatment of weak and strong negation.

⁶ The notion of tetralattice is applicable since $(\mathcal{I}, \leq_t)$ is technically isomorphic to the (truth) bilattice based on $(\mathcal{K}, \leq)$.

Example 2. Recall eFLP P_1 (over $V = \{0, 1\}$) from Example 1. Applying the approximator to the pair $(\mathbf{0}, \mathbf{1})$ yields $\mathcal{T}_{P_1}(\mathbf{0}, \mathbf{1}) = (\mathcal{T}_{P_1}(\mathbf{0}, \mathbf{1})_1, \mathcal{T}_{P_1}(\mathbf{1}, \mathbf{0})_1) = (\mathbf{0}, \{p \mapsto (1, 0), q \mapsto (0, 1)\})$. While all literals stay false in the lower bound, in the upper bound we can see that $\neg p$ and q turn false because they have no rules.

The operator $\mathcal{T}_P$ is $\leq_p$ -monotone, allowing us to generally obtain its least fixpoint by transfinite iteration (i.e. $\text{lfp}(\mathcal{T}_P) = (\prod_{i \geq 0} \mathcal{T}_P^i(\mathbf{0}, \mathbf{1})_1, \bigsqcup_{i \geq 0} \mathcal{T}_P^i(\mathbf{0}, \mathbf{1})_2)$). As $\mathcal{T}_P$ also agrees with T_P on exact inputs, $\mathcal{T}_P$ approximates T_P .

Proposition 3. *For any eFLP P , $\mathcal{T}_P$ is an approximator for T_P .*

5 Reconstructions: Paraconsistent Semantics for Extended Crisp Logic Programs

In this section, we restrict our attention to the classical two truth values $V = \{0, 1\}$ and extended rules $\ell \leftarrow \mathcal{B}$ with $\mathcal{B} = \ell_1 \wedge \dots \wedge \ell_m \wedge \sim \ell_{m+1} \wedge \dots \wedge \sim \ell_n$ denoting $\mathcal{B}^\sim = \{\ell_1, \dots, \ell_m\}$ and $\mathcal{B}^\sim = \{\ell_{m+1}, \dots, \ell_n\}$. Existing (paraconsistent) semantics for extended logic programs [16, 32, 33] typically employ a seemingly different notion of interpretation – sets $S \subseteq \Pi^\pm$. We start off with showing that our notion of interpretation is actually order-isomorphic to such sets of literals.

Proposition 4. *The function $\zeta: \mathcal{I} \rightarrow 2^{\Pi^\pm}$ with $\zeta(I) := \{p \in \Pi \mid I(p)_1 = 1\} \cup \{\neg p \mid I(p)_2 = 1\}$ is an order-isomorphism from $(\mathcal{I}, \leq_t)$ to $(2^{\Pi^\pm}, \subseteq)$, that is, (1) ζ is a bijection, and (2) for all $I, J \in \mathcal{I}$, we have $I \leq_t J \iff \zeta(I) \subseteq \zeta(J)$.*

This order-isomorphism thus preserves all lubs and glbs: for example, we have $\zeta(\bigsqcup \mathcal{J}) = \bigcup \zeta(\mathcal{J})$ for any $\mathcal{J} \subseteq \mathcal{I}$. We will employ this later on.

5.1 Paraconsistent Well-Founded Semantics [32]

Sakama [32] defines a well-founded semantics for extended (crisp) logic programs. His definition uses operators and is based on Przymusiński's definition [30] of the well-founded semantics for normal logic programs [39]. Again, Sakama's model theory is seemingly different: an interpretation $\langle \sigma, \delta \rangle \in 2^{\Pi^\pm} \times 2^{\Pi^\pm}$ is a pair of sets of literals where σ represents *proven facts* while δ represents *default facts*. Similarly to (simple) paraconsistent interpretations, for $V = \{0, 1\}$ these interpretations are isomorphic to pairs $(L, U) \in \mathcal{I}^2$ of paraconsistent interpretations via $\zeta_2: \mathcal{I}^2 \rightarrow 2^{\Pi^\pm} \times 2^{\Pi^\pm}$ with $\zeta_2(L, U) := (\zeta(L), \overline{\zeta(U)})$, where $\overline{S} := \Pi^\pm \setminus S$.

For an interpretation $\langle \sigma, \delta \rangle$, the *satisfaction* of a formula F , denoted $\langle \sigma, \delta \rangle \models F$, is defined inductively as follows: for any literal $\ell \in \Pi^\pm$, $\langle \sigma, \delta \rangle \models \ell$ iff $\ell \in \sigma$, and $\langle \sigma, \delta \rangle \models \sim \ell$ iff $\ell \in \delta$; then, $\langle \sigma, \delta \rangle \models F \wedge G$ iff $\langle \sigma, \delta \rangle \models F$ and $\langle \sigma, \delta \rangle \models G$.

Sakama's paraconsistent well-founded semantics [32] for extended LPs is then defined using the following operators:

$$\begin{aligned} \Phi_{P, \langle \sigma, \delta \rangle}(\alpha) &:= \{\ell \in \Pi^\pm \mid \exists \ell \leftarrow \mathcal{B} \in P : \mathcal{B}^\sim \subseteq \sigma \cup \alpha \text{ and } \mathcal{B}^\sim \subseteq \delta\} \\ \Psi_{P, \langle \sigma, \delta \rangle}(\alpha) &:= \{\ell \in \Pi^\pm \mid \forall \ell \leftarrow \mathcal{B} \in P : \mathcal{B}^\sim \cap (\alpha \cup \delta) \neq \emptyset \text{ or } \mathcal{B}^\sim \cap \sigma \neq \emptyset\} \\ \Theta_P(\sigma, \delta) &:= (\sigma \cup \text{lfp}(\Phi_{P, \langle \sigma, \delta \rangle}), \delta \cup \text{gfp}(\Psi_{P, \langle \sigma, \delta \rangle})) \end{aligned}$$

where the *Sakama well-founded model* of P is given by $\text{lfp}(\Theta_P)$ in the complete lattice $(2^{\Pi^\pm} \times 2^{\Pi^\pm}, \leq_S)$ with $(A, B) \leq_S (C, D)$ iff $A \subseteq C$ and $B \subseteq D$.

In what follows, we will formally establish that the Sakama well-founded model of an extended FLP P coincides (modulo ζ_2) with the standard AFT definition of the well-founded fixpoint of our approximator $\mathcal{T}_P$ from Definition 4.

One of the main observations is that via ζ_2 , evaluations of rule bodies $\mathcal{B}$ exactly correspond in the two approaches, that is, $\widehat{\widehat{(I, J)}}(\mathcal{B}) = 1$ iff $(\zeta(I), \overline{\zeta(J)}) \Vdash \mathcal{B}$. As rule body evaluation directly determines literal evaluation in the operators' return interpretations, this can be leveraged to show that for any extended LP P , the operators $\mathcal{T}_P(\cdot, \cdot)_1$ and $\Phi_{P, \langle \sigma, \delta \rangle}$ coincide (modulo ζ_2); likewise, but with more involved proofs, $\mathcal{T}_P(\cdot, \cdot)_2$ and the complement of $\Psi_{P, \langle \sigma, \delta \rangle}$ coincide (in accordance with ζ_2). From this, it directly follows that the least fixpoints of both operators also coincide, for all pairs of parameter interpretations (i.e. $\langle \sigma, \delta \rangle$ in the case of Sakama's operators). Using the fact that we can reach the least fixpoints of $\mathcal{T}_P^{st}$ and Θ_P by iterating from the least lattice element (and thus building a chain), the slight difference between the definitions of $\mathcal{T}_P^{st}$ and Θ_P (the latter being increasing by definition) can also be shown to be immaterial, whence their respective least fixpoints also coincide modulo isomorphism; this yields the main result.

Theorem 1. *For any extended LP P , we have $\zeta_2(\text{lfp}(\mathcal{T}_P^{st})) = \text{lfp}(\Theta_P)$.*

5.2 Paraconsistent Stable Models [33]

While Sakama and Inoue [33] define their semantics for *disjunctive* extended logic programs (i.e. allowing for disjunction in the head), we consider only single-literal heads here and leave the disjunctive case for future work. The paraconsistent stable model semantics for extended LPs by Sakama and Inoue [33] is defined on the basis of a reduct: Given a program P and an interpretation I , the *reduct* of P w.r.t. I is the program P^I containing exactly all rules $\ell \leftarrow \mathcal{B}^\sim$ for which there is a rule $\ell \leftarrow \mathcal{B} \in P$ with $\widehat{I}(\ell') = 0$ for all $\ell' \in \mathcal{B}^\sim$. The reduct P^I does not contain default negation $\sim$, whence T_{P^I} is $\leq_t$ -monotone by Proposition 2 and thus possesses a least fixpoint that corresponds to the unique least model of P^I (Proposition 1). An interpretation I is then a *stable model* of P iff $I = \text{lfp}(T_{P^I})$ [33].

These considerations suggest a clear path for showing equivalence of Sakama and Inoue's stable models [33] with the stable fixpoints of $\mathcal{T}_P$ as obtained by AFT. The first step is showing that the operator for revising the lower bound equals the van Emden and Kowalski operator of the reduct.

Proposition 5. *For any extended LP P and $I \in \mathcal{I}$, we have $T_{P^I} = \mathcal{T}_P(\cdot, I)_1$.*

It thus follows that the least fixpoints of these operators coincide and thus also the semantics defined through those.

Theorem 2. *Let P be an extended logic program and $I \in \mathcal{I}$. I is a Sakama and Inoue [33] stable model of P if and only if (I, I) is a stable fixpoint of $\mathcal{T}_P$.*

As Sakama and Inoue [33] generalize the semantics of Gelfond and Lifschitz [16], it is clear that for the consistent case, our approximator (and AFT-based stable model semantics) also covers the (non-paraconsistent) stable model semantics of Gelfond and Lifschitz [16].

Corollary 1. *Let P be a crisp extended LP. A consistent interpretation I is a Gelfond and Lifschitz [16] stable model of P iff (I, I) is a stable fixpoint of $\mathcal{T}_P$.*

6 Reconstructions: Stable-Model Semantics for Extended Fuzzy Logic Programs

We now again consider the general case of truth values V , that is, no longer restrict to the two-valued case. This in particular subsumes the case of multi-adjoint fuzzy logic programming, where we reconstruct two stable model semantics for extended *fuzzy* logic programs. It should be noted that neither of the two existing approaches we subsequently reconstruct allows for *paraconsistent* fuzzy reasoning, so we effectively add this capability to these approaches.

6.1 Saad's Stable Models for Extended Fuzzy Logic Programs [31]

Saad [31] uses a slightly different syntax: instead of the rules themselves, all *literals* occurring in rules are annotated. Concretely, a Saad rule [31] has the form

$$\ell_0:c_0 \leftarrow \ell_1:c_1, \dots, \ell_m:c_m, \sim \ell_{m+1}:c_{m+1}, \dots, \sim \ell_n:c_n \quad (5)$$

with $\ell_i \in \Pi^\pm$ and $c_i \in V'$ for $0 \leq i \leq n$, where intuitively $\ell:c$ expresses a lower truth value bound c for literal ℓ . The relevant notion of interpretation for the work of Saad [31] is the set $\mathcal{G}$ of *extended* fuzzy interpretations, where $G \in \mathcal{G}$ is a *partial* function $G: \Pi^\pm \rightarrow [0, 1]$. By permitting literals to remain *undefined*, the interpretation can distinguish between explicit falsity and the absence of information. Then, G *satisfies* $\ell:c$ iff $G(\ell) \geq c$ and ℓ is in the domain of G (denoted $\ell \in \text{dom}(G)$), G *satisfies* $\sim \ell:c$ iff $\ell \notin \text{dom}(G)$ or $[\ell \in \text{dom}(G) \text{ and } G(\ell) < c]$, and G *satisfies* a rule (5) iff the head is satisfied or some body literal is not satisfied. An interpretation G is considered *inconsistent* if there exist complementary literals $\ell, \neg \ell \in \text{dom}(G)$ such that $G(\neg \ell) \neq 1 - G(\ell)$.

As in crisp logic programs, *fuzzy stable models* are defined in terms of a reduct. Given a Saad program Q and a (guessed) interpretation G , the *reduct* w.r.t. G is defined as $Q_G := \{r_G \mid r \in Q \text{ and } G \text{ satisfies all } \sim \ell_j:c_j \text{ in } r\}$, where r_G is obtained from r by removing all negative body literals. Then, Saad [31] shows that $G \in \mathcal{G}$ is a *fuzzy stable model* of Q iff it is the least fixpoint of the immediate consequence-operator O_{Q_G} on the complete lattice $(\mathcal{G}, \leq_A)$ with $H_1 \leq_A H_2$ iff $\text{dom}(H_1) \subseteq \text{dom}(H_2)$ and $H_1(\ell) \leq H_2(\ell)$ for all $\ell \in \Pi^\pm$. For a Saad program Q , the operator $O_Q: \mathcal{G} \rightarrow \mathcal{G}$ is defined by:

$$O_Q(G)(\ell) := \max \left\{ c \mid \begin{array}{l} \ell:c \leftarrow \ell_1:c_1, \dots, \ell_m:c_m, \sim \ell_{m+1}:c_{m+1}, \dots, \sim \ell_n:c_n \in Q \\ \text{and } G \text{ satisfies all body literals of the rule} \end{array} \right\}$$

and is undefined otherwise (i.e., whenever ℓ has no satisfied rules in Q). However, the least fixpoint of O_Q may be inconsistent. Following the approach of Gelfond and Lifschitz [16], Saad [31] stipulates that any inconsistent fuzzy stable model is replaced by the mapping $\Pi^\pm \rightarrow \{1\}$. In what follows, we reconstruct this least-fixpoint characterization within our framework (by means of a translation).

Firstly, to an extended interpretation $G \in \mathcal{G}$, we associate a corresponding paraconsistent interpretation $I_G: \Pi \cup \{d_\ell \mid \ell \in \Pi^\pm\} \rightarrow V^2$, where the fresh atoms d_ℓ serve to record whether a literal ℓ belongs to the domain of G . We set $I_G(p)_1 := G(p)$ if $p \in \text{dom}(G)$ and 0 otherwise; $I_G(p)_2 := G(\neg p)$ if $\neg p \in \text{dom}(G)$ and 0 otherwise, for $p \in \Pi$; $I_G(d_\ell) := (1, 0)$ if $\ell \in \text{dom}(G)$ and $(0, 0)$ otherwise.

The idea for our translation is to introduce, for each $c \in V'$ occurring in a Saad program [31], a new unary connective $\succeq_c$ with associated truth function $\succeq_c(x) := 1$ if $x \geq c$ and 0 otherwise, and to use it to check lower bounds in rule bodies. For any literal ℓ occurring as $\ell:0$, the treatment is different: $\succeq_0(\ell)$ simply becomes the atom d_ℓ used to check whether “ $\ell \in \text{dom}(G)$ ”.

Definition 5. A Saad rule (5) is translated into eFLP rule $\ell_0 \leftarrow c_0 \wedge \mathcal{B}^\prec \wedge \mathcal{B}^\sim$ with $\mathcal{B}^\prec := \succeq_{c_1}(\ell_1) \wedge \dots \wedge \succeq_{c_m}(\ell_m)$ and $\mathcal{B}^\sim := \sim \succeq_{c_{m+1}}(\ell_{m+1}) \wedge \dots \wedge \sim \succeq_{c_n}(\ell_n)$.

Example 3. Consider $Q = \{p:1 \leftarrow \sim q:0\}$. The rule is satisfied by $G = \{p \mapsto 1\}$ and $H = \{q \mapsto 0\}$ with G being the only fuzzy stable model. The only stable model of the translation $Q' = \{p \leftarrow 1 \wedge \sim d_q\}$ is $I_G = \{p \mapsto (1, 0); q, d_q \mapsto (0, 0)\}$.

First, we show that this translation preserves rule body satisfaction.

Lemma 1. Let r be a Saad rule (5). For any extended interpretation $G \in \mathcal{G}$ with corresponding paraconsistent interpretation I_G , we have

$$G \text{ satisfies the body of } r \iff \widehat{I_G}(c_0 \wedge \mathcal{B}^\prec \wedge \mathcal{B}^\sim) \geq c_0,$$

where $c_0 \wedge \mathcal{B}^\prec \wedge \mathcal{B}^\sim$ is the body of the translation of r .

Definition 6. A Saad program Q is translated into an extended fuzzy logic program P_Q (Definition 1) rule by rule; whenever for some $\ell \in \Pi^\pm$ both $\ell:0$ and $\ell:c$ for some $c > 0$ occur in Q , we add a rule $d_\ell \leftarrow \succeq_c(\ell)$ to P_Q (for each such c).

The rules $d_\ell \leftarrow \succeq_c(\ell)$ in the translation are necessary to avoid translated programs allowing additional (stable) models. E.g., consider $Q = \{p:1 \leftarrow \sim p:0\}$ with “translation” $Q' = \{p \leftarrow 1 \wedge \sim d_p\}$ having $\{p \mapsto (1, 0), d_p \mapsto (0, 0)\}$ as stable fixpoint of $\mathcal{T}_{Q'}$, although Q does not have a stable model. In any case, we have the following correspondence between the consequence operators O_Q and $T_{Q'}$.

Lemma 2. Let Q be an extended fuzzy program according to [31], Q' its translation as in Definition 6, and $G \in \mathcal{G}$ an extended fuzzy interpretation with counterpart I_G . Then for all $\ell \in \text{dom}(O_Q(G))$, we have $\widehat{T_{Q'}}(\widehat{I_G})(\ell) = O_Q(G)(\ell)$.

This culminates in the desired correspondence on the relevant (reduct) operators (Lemma 3) and hence this section’s main result, the correspondence of the semantics on programs (Theorem 3).

Lemma 3. *Let Q be a Saad program [31], Q' its translation as in Definition 6, and $G \in \mathcal{G}$ an extended fuzzy interpretation with counterpart I_G . For any extended fuzzy interpretation $H \in \mathcal{G}$ with counterpart I_H , we have*

$$\mathcal{T}_{Q'}(\widehat{I_H, I_G})_1(\ell) = O_{Q_G}(H)(\ell) \text{ if } \ell \in \text{dom}(O_{Q_G}(H))$$

Theorem 3. *Consider an extended fuzzy LP Q according to Saad [31] and its translation Q' obtained via Definition 6. An interpretation $G \in \mathcal{G}$ is a Saad [31] fuzzy stable model of Q iff (I_G, I_G) is a stable fixpoint of $\mathcal{T}_{Q'}$.*

6.2 Stable Models for Extended Fuzzy LPs by Cornejo et al. [9]

Generalizing earlier work on *normal* logic programs [8], Cornejo et al. [9] define a (non-paraconsistent) stable model semantics for *extended* (multi-adjoint) FLPs. In this section, we recall their semantics and show how we can map their syntax into ours to obtain a one-to-one-correspondence of the respective stable models.

In their syntax for extended fuzzy rules [9], they do not have explicit classical negation, but instead allow a special kind of n -ary aggregation connective f_e such that for some m , $0 \leq m \leq n$, the function f_e is $\leq$ -monotone in arguments $1, \dots, m$ and $\leq$ -antimonotone in arguments $m+1, \dots, n$, where the underlying idea however still is that such extended aggregators can be expressed by body formulas similar to (4), employing monotone connectives on top of negation $\sim$. In contrast to the classical literature on extended logic programs, Cornejo et al. [9] have no negative literals in rule heads, as there is no classical negation symbol in the syntax. Instead, they allow *constraints*, rules whose head is a concrete truth value that intuitively provides an upper bound on the truth value of the constraint's body, following the extension of classical constraints to the fuzzy case by Janssen et al. [21]. Formally, these rules and constraints take the forms

$$p \leftarrow \mathcal{B} \quad \text{and} \quad c \leftarrow \mathcal{B}$$

with $c \in V'$ and $\mathcal{B}$ a normal fuzzy formula. The space of relevant interpretations is again the set $\mathcal{K}$ of all non-paraconsistent fuzzy interpretations $K: \Pi \rightarrow V$. Given a program Q and a candidate interpretation K , the *program reduct* Q_K is obtained by replacing every rule/constraint body $\mathcal{B}$ by its respective body reduct $\mathcal{B}_K$: explicitly denoting the (non-negated/negated) atoms that occur in a body by $\mathcal{B}[p_1, \dots, p_m, \sim p_{m+1}, \dots, \sim p_n]$, the *body reduct* $\mathcal{B}_K$ is then such that for all $x_1, \dots, x_m \in V$, $\mathcal{B}_K(x_1, \dots, x_m) := \mathcal{B}[x_1, \dots, x_m, \sim K(p_{m+1}), \dots, \sim K(p_n)]$. As expected, then, a candidate $K \in \mathcal{K}$ is a *stable model of Q* iff K is the least model of Q_K (with respect to the pointwise lifting of $\leq$ to $\mathcal{K}$) and K satisfies all constraints in Q_K (where a constraint $c \leftarrow \mathcal{B}$ is satisfied by K iff $[K](\mathcal{B}) \leq c$).

Towards our reconstruction of that semantics, we first define how we translate programs in the syntax of Cornejo et al. [9] into our syntax of Definition 1.

Definition 7. *Let Q be a fuzzy logic program over atoms Π according to [9]. For a rule $r = p \leftarrow \mathcal{B} \in Q$ for $p \in \Pi$, define $P_r := \{r\}$; for a constraint $r = c \leftarrow \mathcal{B} \in Q$ for $c \in V'$, define its translation $P_r := \{p_r \leftarrow \mathcal{B}, \neg p_r \leftarrow \neg c\}$, where p_r is a fresh atom not occurring in Q . Finally, define $P_Q := \bigcup_{r \in Q} P_r$.*

Intuitively, the rule for p_r records the actual truth value of the body $\mathcal{B}$; the rule for $\neg p_r$ records the greatest possible truth value c of $\mathcal{B}$ that is *permitted* by r , by requiring a least truth value $\neg c$ for $\neg p_r$.⁷ The main idea to guarantee constraint satisfaction in the semantics is now to restrict models to the consistent ones and thus to filter out all interpretations where some constraint is violated. Clearly, constraint violation is also the *only* reason for a model to be filtered out, as there are no other rules with negated heads and thus no other potential sources for classical inconsistency.

Our main result of this section shows that our translation allows us to reconstruct their stable model semantics with our approximator from Definition 4. Before proving the main result, we establish some useful prerequisites. Similar to before, we lift $K \in \mathcal{K}$ to a paraconsistent interpretation I_K by setting $I_K(p) := (K(p), \neg K(p))$ for atoms occurring in P_Q and $I_K(p) := (K(p), 0)$ otherwise.

Lemma 4. *For any rule/constraint body $\mathcal{B}$ occurring in an extended fuzzy logic program Q in the syntax of Cornejo et al. [9] and any non-paraconsistent fuzzy interpretation $K \in \mathcal{K}$, we have that for all $J \in \mathcal{I}$: $\widehat{(J, I_K)}(\mathcal{B}) = \widehat{J}(\mathcal{B}_K)$.*

Using this, we can again establish identity of operators.

Lemma 5. *For any extended fuzzy logic program Q according to [9] with translation P_Q , and non-paraconsistent fuzzy interpretation $K \in \mathcal{K}$ with counterpart I_K , we have $\mathcal{T}_{P_Q}(\cdot, I_K)_1 = \mathcal{T}_{P_{Q_K}}$.*

Theorem 4. *Consider an extended fuzzy logic program Q [9] and its translation P_Q obtained as above. An interpretation $K \in \mathcal{K}$ is a Cornejo et al. [9] stable model of Q if and only if (I_K, I_K) is a stable fixpoint of $\mathcal{T}_{P_Q}$.*

7 New Contributions Based on Reconstructions

Beyond the previously reconstructed semantics, the approximator $\mathcal{T}_P$ yields an immediate semantic payoff. First, we show that, with respect to an nFLP program P , the approximator $\mathcal{T}_P$ naturally generalizes the approximator $\mathcal{A}_P$ introduced by Kettmann et al. [23]. Here we define the corresponding interpretation $I_K^+ \in \mathcal{I}$ of a non-paraconsistent interpretation $K \in \mathcal{K}$ as $I_K^+ := (K, \{p \mapsto 0 \mid p \in \Pi\})$.

Proposition 6. *For any nFLP P and non-paraconsistent interpretations $K, L \in \mathcal{K}$ with paraconsistent counterparts $I_K^+, I_L^+ \in \mathcal{I}$, we have $\mathcal{T}_P(I_K^+, I_L^+)_1 = I_{\mathcal{A}_P(K, L)_1}^+$.*

In particular, this result implies that the well-founded semantics defined by Loyer and Straccia [27] arises as a special case of our fuzzy well-founded semantics. More generally, our framework provides a paraconsistent well-founded semantics for eFLP, extending the semantics of Loyer and Straccia [27] to paraconsistency, and lifting the paraconsistent semantics of Sakama [32] from the crisp to the fuzzy setting. It also induces paraconsistent stable models for eFLP, extending the work of Sakama and Inoue [33], Cornejo et al. [9], and Janssen et al. [21].

⁷ We fix $\neg$ to be an involution such that $\neg c \in V'$. This choice depends on the concrete V' ; for the typical fuzzy logic case of $V' = [0, 1] \cap \mathbb{Q}$, we can choose $\neg x := 1 - x$.

8 Discussion

We already discussed relations with a host of related work in the reconstructive sections (Sections 5 and 6). Here, we survey other related work. One line of work on extended logic programs that we did not reconstruct is that done by Alferes, Damásio, Pereira, and Alcântara [2,11,1]. They introduce the *well-founded semantics for extended logic programs*. Their semantics is a straightforward adaption of the well-founded semantics for normal logic programming [38], but is based on what they call the *principle of coherence*: “believing $\neg p$ entails believing $\sim p$ ”. By double negation, this also means that believing p entails believing $\sim \neg p$. In the presence of contradictions, the AFT-perspective we take (and, as shown by the reconstructions, also underlies other approaches) is incompatible with the principle of coherence. Indeed, as we only consider pairs (L, U) s.t. $L \leq_t U$, any time we believe p and $\neg p$ (i.e. $p, \neg p \in L$), we also believe in the non-falsity of p and $\neg p$ (i.e. $p, \neg p \in U$), which precludes us from believing $\sim \neg p$ respectively $\sim p$. On the other hand, Alcântara et al. [1] need some additional, restrictive assumptions on their negation functions to make full use of the coherence principle: both classical and default negations must be involutions (self-inverse bijections), and furthermore they must commute. In our framework, we need no such assumptions and can thus also accommodate Gödel negation (which is not involutive).

Another related work is that of Killen and You [24], who use tetralattices to propose an alternative definition of stable revision. Essentially, they use tetralattices to obtain pairs (T, F, U, P) , where T gives necessary truth, F well-established falsity, U possible falsity and P possible truth. On the other hand, we consider pairs of pairs $((L_1, L_2), (U_1, U_2))$ where U_1 and U_2 [L_1 and L_2] represent the possible [necessary] truth of p respectively $\neg p$.

We also provide a proof-of-concept implementation in ASP, using the *meta-programming* capabilities of clingo [22]. We restricted our implementation to fixpoint and Kripke-Kleene semantics for crisp eFLPs. More details can be found in the online repository at <https://github.com/JeroenSpaans/sningo>.

Janssen et al. [21] showed that several common extensions of normal fuzzy logic programs, such as constraints and extended aggregators, can be simulated by normal rules while preserving the fuzzy answer set semantics. For the extension allowing strong negation, however, the semantics is based on a slightly different notion of modelhood, using the approach of Van Nieuwenborgh et al. [40]. In this setting, two special aggregators are employed: one to determine the degree to which an interpretation I is consistent (by aggregating the consistency values induced by an associated consistency function across all atoms), and another to determine the degree to which I is a model of P (by aggregating the satisfaction degrees of all rules of P). We aim to generalize this approach in future work. Other planned future work includes the generalization of this work to extended or fuzzy logic programs with disjunction, using non-deterministic AFT [18]. Furthermore, we see several potential applications of extended fuzzy logic programs, e.g. in law, where conflicting obligations have to be combined with fuzzy concepts [3], or in neuro-symbolic AI, where fuzzy logic is often used to integrate logical rules in a loss function [34], and where hallucinations might cause inconsistencies.

Acknowledgments. This work is partly supported by BMFTR (Federal Ministry of Research, Technology, and Space) in DAAD project 57616814 (SECAI).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Alcântara, J., Damásio, C.V., Pereira, L.M.: An encompassing framework for paraconsistent logic programs. *Journal of Applied Logic* **3**(1), 67–95 (2005)
2. Alferes, J.J., Damasio, C.V., Pereira, L.M.: A logic programming system for non-monotonic reasoning. *Journal of Automated Reasoning* **14**(1), 93–147 (1995)
3. Atkinson, K., Bench-Capon, T.: ANGELIC II: An improved methodology for representing legal domain knowledge. In: *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law*. pp. 12–21 (2023)
4. Belnap, N.D.: A useful four-valued logic. In: *Modern uses of multiple-valued logic*, pp. 5–37. Springer (1977)
5. Bourbaki, N.: Sur le théorème de Zorn. *Archiv der Mathematik* **2**(6), 434–437 (1949)
6. Brewka, G., Eiter, T., Truszczyński, M.: Answer set programming at a glance. *Commun. ACM* **54**(12), 92–103 (2011)
7. Clark, K.L.: Negation as failure. In: Gallaire, H., Minker, J. (eds.) *Logic and Data Bases, Symposium on Logic and Data Bases*. pp. 293–322. *Advances in Data Base Theory* (1977)
8. Cornejo, M.E., Lobo, D., Medina, J.: Syntax and semantics of multi-adjoint normal logic programming. *Fuzzy Sets and Systems* **345**, 41–62 (2018)
9. Cornejo, M.E., Lobo, D., Medina, J.: Extended multi-adjoint logic programming. *Fuzzy Sets and Systems* **388**, 124–145 (2020)
10. Cousot, P., Cousot, R.: Constructive versions of Tarski’s fixed point theorems. *Pacific Journal of Mathematics* **82**(1), 43–57 (1979)
11. Damásio, C.V., Pereira, L.M.: A model theory for paraconsistent logic programming. In: *Portuguese Conference on Artificial Intelligence*. pp. 377–386 (1995)
12. Denecker, M., Marek, V., Truszczyński, M.: Approximations, Stable Operators, Well-Founded Fixpoints and Applications in Nonmonotonic Reasoning. In: *Logic-Based Artificial Intelligence*, pp. 127–144. Kluwer Academic Publishers (2000)
13. van Emden, M.H., Kowalski, R.A.: The semantics of predicate logic as a programming language. *J. ACM* **23**(4), 733–742 (1976)
14. Fitting, M.: Fixpoint Semantics for Logic Programming: A Survey. *Theoretical Computer Science* **278**(1–2), 25–51 (2002)
15. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: *Answer Set Solving in Practice. Synthesis Lectures on Artificial Intelligence and Machine Learning*, Morgan & Claypool Publishers (2012)
16. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. *New Gener. Comput.* **9**(3/4), 365–386 (1991)
17. Ginsberg, M.L.: Multivalued logics: a uniform approach to reasoning in artificial intelligence. *Comput. Intell.* **4**, 265–316 (1988)
18. Heyninck, J., Arieli, O., Bogaerts, B.: Non-deterministic approximation fixpoint theory and its application in disjunctive logic programming. *Artif. Intell.* **331**, 104110 (2024)
19. Hinde, C.J.: Fuzzy prolog. *Int. J. Man Mach. Stud.* **24**(6), 569–595 (1986)

20. Hájek, P.: *Metamathematics of Fuzzy Logic*, Trends in Logic, vol. 4. Springer Netherlands (1998)
21. Janssen, J., Schockaert, S., Vermeir, D., de Cock, M.: A core language for fuzzy answer set programming. *Int. J. Approx. Reason.* **53**(4), 660–692 (2012)
22. Kaminski, R., Romero, J., Schaub, T., Wanko, P.: How to build your own ASP-based system?! *Theory and Practice of Logic Programming* **23**(1), 299–361 (2023)
23. Kettmann, P., Heyninck, J., Strass, H.: Approximation fixpoint theory as a unifying framework for fuzzy logic programming semantics. In: *Proceedings of the 34th International Joint Conference on Artificial Intelligence*. pp. 4544–4552 (2025)
24. Killen, S., You, J.H.: Eliminating Unintended Stable Fixpoints for Hybrid Reasoning Systems. *CoRR* **abs/2307.11286** (2023)
25. Kowalski, R.A., Kuehner, D.: Linear resolution with selection function. *Artif. Intell.* **2**(3/4), 227–260 (1971)
26. Lee, R.C.T.: Fuzzy logic and the resolution principle. *J. ACM* **19**(1), 109–119 (1972)
27. Loyer, Y., Straccia, U.: Approximate well-founded semantics, query answering and generalized normal logic programs over lattices. *Ann. Math. Artif. Intell.* **55**(3–4), 389–417 (2009)
28. Markowsky, G.: Chain-complete posets and directed sets with applications. *Algebra Universalis* **6**, 53–68 (1976)
29. Medina, J., Ojeda-Aciego, M., Vojtáš, P.: Multi-adjoint logic programming with continuous semantics. In: Eiter, T., Faber, W., Truszczyński, M. (eds.) *Logic Programming and Nonmonotonic Reasoning*, 6th International Conference, Proceedings. LNCS, vol. 2173, pp. 351–364 (2001)
30. Przymusiński, T.C.: Every logic program has a natural stratification and an iterated least fixed point model. In: *Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Proceedings. p. 11–21. PODS '89 (1989)
31. Saad, E.: Extended fuzzy logic programs with fuzzy answer set semantics. In: Godo, L., Pugliese, A. (eds.) *Scalable Uncertainty Management*, Third International Conference, SUM 2009, Proceedings. LNCS, vol. 5785, pp. 223–239 (2009)
32. Sakama, C.: Extended well-founded semantics for paraconsistent logic programs. In: *FGCS*. pp. 592–599 (1992)
33. Sakama, C., Inoue, K.: Paraconsistent stable semantics for extended disjunctive programs. *J. Log. Comput.* **5**(3), 265–285 (1995)
34. Sen, P., de Carvalho, B.W., Riegel, R., Gray, A.: Neuro-symbolic inductive logic programming with logical neural networks. In: *Proceedings of the AAAI conference on artificial intelligence*. vol. 36, pp. 8212–8219 (2022)
35. Shapiro, E.Y.: Logic programs with uncertainties: A tool for implementing rule-based systems. In: Bundy, A. (ed.) *Proceedings of the 8th International Joint Conference on Artificial Intelligence*. Karlsruhe, FR Germany. pp. 529–532 (1983)
36. Tarski, A.: A Lattice-Theoretical Fixpoint Theorem and Its Applications. *Pacific Journal of Mathematics* **5**(2), 285–309 (1955)
37. Van Emden, M.H.: Quantitative deduction and its fixpoint theory. *The Journal of Logic Programming* **3**(1), 37–53 (1986)
38. van Gelder, A.: Negation as failure using tight derivations for general logic programs. In: Minker, J. (ed.) *Foundations of Deductive Databases and Logic Programming*, pp. 149–176. Morgan Kaufmann (1988)
39. van Gelder, A., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. *J. ACM* **38**(3), 620–650 (1991)
40. Van Nieuwenborgh, D., De Cock, M., Vermeir, D.: An introduction to fuzzy answer set programming. *Ann. Math. Artif. Intell.* **50**(3–4), 363–388 (Aug 2007)
41. Vojtáš, P.: Fuzzy logic programming. *Fuzzy Sets Syst.* **124**(3), 361–370 (2001)