An Alternative Theory of Stable Revision for Nondeterministic Approximation Fixpoint Theory and the Relationships

Spencer Killen[1], Jia-Huai You[1], Jesse Heyninck[2,3]
[1]University of Alberta, Canada
[2]Open Universiteit, the Netherlands
[3]University of Cape Town, South-Africa
sjkillen@ualberta.ca, jyou@ualberta.ca, jesse.heyninck@ou.nl

Abstract

Approximation fixpoint theory (AFT) is a robust and popular mathematical framework that characterizes many nonmonotonic semantics, where the construction of stable fixpoints, called *stable revision*, plays a central role. Nondeterministic AFT is a recent development that redefines AFT for a nondeterministic setting to capture disjunctive semantics. This theory departs from traditional AFT by introducing distinct definitions, thus raising the question of whether deterministic AFT can be adopted directly to define nondeterministic stable revision. This work proposes such an alternate theory and creates a new way to study disjunctive semantics in terms of normal (non-disjunctive) knowledge bases. To demonstrate the viability of our framework, we show how to capture stable and partial stable models for disjunctive logic programs. We then study the relationships between this alternative theory and the state-of-the-art nondeterministic AFT.

1 Introduction

Nonmonotonic semantics allow new facts to invalidate previously established conclusions. This feature accurately reflects certain aspects of human reasoning. Thus, it is vital for descriptions of our knowledge to capture nonmonotonicity. Fixpoint theory, an elegant method of capturing semantics, requires monotonicity (Tarski 1955), so is not immediately apparent how to represent a nonmonotonic knowledge base using fixpoint functions. A widely adopted branch of fixpoint theory, Approximation Fixpoint Theory (AFT) (Denecker, Marek, and Truszczyński 2000), leverages intervals (approximations) to monotonically capture fixpoints of nonmonotonic functions. In AFT, this construction is called *stable revision*, and the corresponding fixpoints are called the *stable fixpoints* of the underlying approximator (a.k.a. an approximating operator). Stable revision is the key to capturing the minimality principle commonly adopted in knowledge representation. In general, to apply AFT to a semantics is to formulate an approximator whose stable fixpoints characterize the intended semantics. Due to its relative ease of use and generality, AFT has been used for the comparison, refinement, and the creation of semantics; see, e.g., Marynissen, Bogaerts, and Denecker (2024); Vanbesien, Bruynooghe, and Denecker (2022); Antic, Eiter, and Fink (2013); Denecker, Marek, and Truszczynski (2003); Liu and You (2022); Marynissen, Bogaerts, and Denecker (2021).

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

However, the original AFT is unsuitable for nondeterministic semantics, which are essential for capturing uncertainty. While functions in AFT map lattice values to other values in the same lattice, nondeterminism requires mappings from lattice elements to higher-order structures, such as a set of lattice elements. This mismatch of domain and range prompts the development of a framework with the broad applicability of AFT that can express nondeterministic semantics.

Heyninck et al. (2024) tackle this challenge by formalizing a new theory of nondeterministic AFT, which has already proven useful in providing semantics for disjunctive logic programs (Heyninck, Arieli, and Bogaerts 2024), aggregate disjunctive logic programs (Heyninck and Bogaerts 2023) and choice programs (Heyninck 2024). They build their theory from the ground up, establishing new definitions that generalize those from deterministic AFT of Denecker et al. (2000). Their theory of nondeterministic AFT is highly suitable for characterizing disjunctive semantics.

However, their theory departs significantly from standard AFT. Namely, it introduces its own notion of fixpoints, exactness, and monotonicity. The framework also relies on different orderings than deterministic AFT. Though nondeterministic AFT faithfully generalizes deterministic AFT, that is, nondeterministic AFT can express deterministic AFT, further questions remain as to how these theories are related. It is natural to ask whether a nondeterministic operator can be alternatively viewed as a family of deterministic operators. More generally, the question is whether we can provide the lattice-theoretic characterizations of a nondeterministic semantics by relying only on the notations and constructions of deterministic AFT.

Because stable revision is at the center of AFT, we focus on the following critical question - whether nondeterministic stable revision can be defined by deterministic AFT. Our proposed solution lifts deterministic stable revision to a nondeterministic setting using sets of approximators and simplifies the application of AFT to nondeterministic semantics by requiring fewer definitions. Our approach clarifies the relationship between deterministic and nondeterministic AFT. Namely, we generalize a prior idea that disjunctive

semantics may be expressed as a family of normal semantics. When applied to disjunctive logic programs, this means that the semantics based on stable and partial stable models can be expressed by the corresponding semantics of a set of normal logic programs. More importantly, we use lattice-theoretic constructions to lift this idea to the abstract level of algebra so that our approach may be applied to other nonmonotonic systems as well.

Our core contributions in this work are as follows.

- We leverage sets of stable operators to define nondeterministic stable revision,
- Demonstrate our formalism by capturing partial stable models of disjunctive logic programs, and
- Compare our theory with nondeterministic AFT (Heyninck et al. 2024).

This paper is organized as follows. In Section 2, we cover necessary background material, including lattice and AFT theory. In Section 3, we introduce our framework, then demonstrate its viability by applying it to disjunctive logic programs in Section 4. Next, we relate our theory to nondeterministic AFT (Heyninck et al. 2024) and compare the theories in detail (Section 5). Finally, we discuss related work, limitations, and future directions in Sections 6 and 7. Some extended proofs are left to the technical appendix.

2 Preliminaries

We review common lattice theory (Roman 2008). A complete lattice $\langle \mathcal{L}, \preceq \rangle$ is a poset s.t. every subset $S \subseteq \mathcal{L}$ has a least upper bound and a greatest lower bound, denoted $\bigvee_{\preceq}$ and $\bigwedge_{\preceq}$ respectively, with $\top := \bigvee_{\preceq} \mathcal{L}$ and $\bot := \bigwedge_{\preceq} \mathcal{L}$. Given two preorders $\langle \mathcal{L}_a, \preceq_a \rangle$ and $\langle \mathcal{L}_b, \preceq_b \rangle$, we say that a function $f : \mathcal{L}_a \to \mathcal{L}_b$ is *order-preserving* from $\preceq_a$ to $\preceq_b$ if for each $x, y \in \mathcal{L}_a$ we have $x \preceq_a y \Rightarrow f(x) \preceq_b f(y)$ When $\langle \mathcal{L}_a, \preceq_a \rangle = \langle \mathcal{L}_b, \preceq_b \rangle$, we instead say that f is $\preceq_a$-*monotone*. When the relation for the $\preceq_b$ ordering is flipped, that is, $x \preceq_a y \Rightarrow f(y) \preceq_b f(x)$, we say that f is *anti-order-preserving* (or *anti-monotone* if $\langle \mathcal{L}_a, \preceq_a \rangle = \langle \mathcal{L}_b, \preceq_b \rangle$).

Every preorder $\langle \mathcal{L}, \preceq \rangle$ has a dual ordering $\langle \mathcal{L}, \succeq \rangle$ where the ordering has been "flipped". We also define a strict variant $\prec$ to mean two elements are comparable and not equal. We use $\wp(S)$ to indicate the powerset of the set S and denote the powerset without the empty set as $\wp^o(S)$. Let $o : \mathcal{L} \to \mathcal{L}$ be a $\preceq$-monotone function over $\langle \mathcal{L}, \preceq \rangle$. An element $x \in \mathcal{L}$ is a $\preceq$-*prefixpoint* of o if $o(x) \preceq x$. We call a prefixpoint x of o a *fixpoint* of o if $o(x) = x$. The fixpoints of a $\preceq$-monotone function $o : \mathcal{L} \to \mathcal{L}$ on a complete lattice $\langle \mathcal{L}, \preceq \rangle$ form a complete lattice (Tarski 1955). We call the least element of this lattice *the least fixpoint* and denote it with $\mathbf{lfp}_{\preceq_\alpha} o$. For a lattice $\langle \mathcal{L}, \preceq \rangle$, we define the upwards closure on a set $C \subseteq \mathcal{L}$ as $C\uparrow := \{x \in \mathcal{L} \mid \exists y \in C, y \preceq x\}$. We define $\downarrow$ identically using the flipped ordering. We use $\boldsymbol{min}_{\preceq}(S)$ to denote the subset of $\preceq$-minimal elements from S, i.e., $\boldsymbol{min}_{\preceq}(S) := \{b \in S \mid \neg\exists b' \in S, b' \prec b\}$. For an infinitely descending $\preceq$-chain C, $\boldsymbol{min}_{\preceq}(C)$ is empty.

Introduced by Denecker et al. (2000), AFT (Approximation Fixpoint Theory) is a framework for defining operators and constructing their fixpoints over a bilattice.

Definition 2.1. *Given a complete lattice $\langle \mathcal{L}, \preceq \rangle$, its bilattice is the pair of complete lattices $\langle \mathcal{L}^2, \preceq_t^2 \rangle$ (the truth-ordering) and $\langle \mathcal{L}^2, \preceq_p^2 \rangle$ (the precision-ordering) where $\preceq_t^2$ and $\preceq_p^2$ are defined for any $(x', y'), (x, y) \in \mathcal{L}^2$:*

$$(x', y') \preceq_t^2 (x, y) \quad \textit{iff} \quad x' \preceq x \textit{ and } y' \preceq y$$
$$(x', y') \preceq_p^2 (x, y) \quad \textit{iff} \quad x' \preceq x \textit{ and } y' \succeq y$$

We call a pair $(x, x) \in \mathcal{L}^2$ *exact* and we call a function $o : \mathcal{L} \to \mathcal{L}$ *exact* if it maps exact pairs to exact pairs, that is, if for any $x \in \mathcal{L}$, $o(x, x) = (o(x, x)_1, o(x, x)_1)$. We follow Heyninck et al. (2024) in adopting a definition of Denecker et al.'s (2000)'s approximators based on exactness.

Definition 2.2. *A (deterministic)* approximator *is an exact $\preceq_p^2$-monotone function $o : \mathcal{L}^2 \to \mathcal{L}^2$ on a bilattice $\langle \mathcal{L}^2, \preceq_p^2 \rangle$.*

The stable revision operator (Denecker et al. 2000) converts an approximator into an operator with more precise fixpoints.

Definition 2.3. *Given an approximator o, the* stable revision operator $S(o)$ *is defined as follows:*

$$S(o)(x, y) := (\mathbf{lfp}_{\preceq}(o(\cdot, y)_1), \mathbf{lfp}_{\preceq}(o(x, \cdot)_2))$$

We refer to the fixpoints of the stable revision operator as *stable fixpoints*. The smallest fixpoint w.r.t. $\preceq_p^2$ is called the *least stable fixpoint* and every fixpoint of the stable revision operator is a fixpoint of the original approximator (Denecker et al. 2000). We call a pair (x, y) *consistent* if it has the property $x \preceq y$. Similarly, we call a function $f : \mathcal{L}^2 \to \mathcal{L}^2$ *consistent* if it maps consistent pairs to consistent pairs.

3 Approximator Sets

The main goal of this paper is to capture the semantics of nonmonotonic nondeterministic operators as sets of deterministic operators. For example, a disjunctive logic program $\{a, b \leftarrow\}$ can be captured using the associated normal logic programs $\{a \leftarrow\}$ and $\{b \leftarrow\}$. Here, we take heavy inspiration from Heyninck et al. (2024) by crafting similar examples and properties, yet we directly leverage traditional AFT. At the center of our theory are sets of approximators:

Definition 3.1. *An* approximator set (apprx. set) *H is a nonempty set of approximators.*

Truth minimality ($\preceq_t^2$) is a core property of deterministic stable fixpoints. When we lift deterministic stable fixpoints to apprx. sets below, we perform $\preceq_t^2$-minimization.

Definition 3.2. *A pair (x, y) is a γ-*stable fixpoint *of an apprx. set H if $(x, y) \in \boldsymbol{min}_{\preceq_t^2}\{S(h)(x, y) \mid h \in H\}$*

Example 1. *Consider the following set of approximators $H = \{h_a, h_b\}$ over $\langle \wp(\{a, b\}), \subseteq \rangle$:*

h	$h(\emptyset, y)_1$	$h(\{a\}, y)_1$	$h(\{b\}, y)_1$	$h(\{a, b\}, y)_1$
h_a	$\{a\}$	$\{a\}$	$\{a\}$	$\{a\}$
h_b	$\{b\}$	$\{b\}$	$\{a, b\}$	$\{a, b\}$

with $y \subseteq \{a, b\}$, and $h'(x, y)_2 = h'(y, x)_1$ for any $x, y \subseteq \{a, b\}$, $h' \in H$. It's easy to show that H is an apprx. set and that $(\{a\}, \{a\})$ is a stable fixpoint of h_a while $(\{b\}, \{b\})$ is not a stable fixpoint of h_b. Further, $(\{a\}, \{a\})$ is a γ-stable fixpoint of H because $S(h_b)(\{a\}, \{a\}) \not\preceq_t^2 S(h_a)(\{a\}, \{a\})$

By definition, every γ-stable fixpoint is a stable fixpoint of some approximator in the apprx. set H. We can show that these stable fixpoints are $\preceq_t^2$-minimal w.r.t. to one another.

Theorem 3.1. *For any γ-stable fixpoint (x, y) of an apprx. set H, there is no γ-stable fixpoint (x', y') of H s.t. $(x', y') \prec_t^2 (x, y)$.*

Proof. Let (x', y') and (x, y) be two γ-stable fixpoints of H s.t. $(x', y') \preceq_t^2 (x, y)$. As a stable fixpoint of some $h' \in H$ (resp. $h \in H$) (x', y') (resp. (x, y)) is a fixpoint of h' (resp. h) (Denecker et al. 2000). We have $x' \preceq x$ and $y' \preceq y$. We briefly follow Denecker et al.'s (2000) proof (Theorem 4). The $\preceq_p^2$-monotonicity of h' ensures that $h'(\cdot, y')_2$ is $\preceq$-antimonotone. Thus, y' is a prefixpoint of $h'(x, \cdot)$ and the least fixpoint of $h'(x, \cdot)$ is less than y' (Tarski 1955). We have $\mathbf{lfp}_{\preceq}\, h'(x, \cdot) \preceq y'$, with $y' \preceq y$, it follows by transitivity that $(\mathbf{lfp}_{\preceq}\, h'(x, \cdot)) = S(h')(x, y)_2 \preceq y$. We can apply a similar approach to show $S(h')(x, y)_1 \preceq x$, thus $S(h')(x, y) \preceq_t^2 (x, y)$. But, $\neg(S(h')(x, y) \prec_t^2 (x, y))$, thus $S(h')(x, y) = (x, y)$. By the truth-minimality of stable fixpoints of h' (Denecker et al. 2000), it follows that $(x', y') = (x, y)$. □

While it may seem as though the above claim comes automatically from the use of $\boldsymbol{min}_{\preceq_t^2}$, we must note that γ-stable fixpoints do not minimize stable fixpoints, but rather we confirm that (x, y) is not a (strict) $\prec_t^2$-prefixpoint of any stable operator. It is possible for a stable fixpoint to be a $\prec_t^2$-prefixpoint of another stable operator which does not have a $\prec_t^2$-smaller stable fixpoint. In contradistinction to deterministic AFT, a γ-stable fixpoint may not exist. A finite example showing that a γ-stable fixpoint does not exist is nontrivial. Thus, we defer it until Section 4 (Example 6). Following Heyninck et al. (Example 22 (2024)) in the following, we can construct and example using an infinite lattice.

Example 2. *Let $\langle \mathcal{L}, \leq \rangle$ be the complete lattice consisting of the negative integers and an added least element $\bot$. Define an apprx. set H as follows*

$$H \coloneqq \{h_i \mid \bot < i\} \qquad h_i(x, y) \coloneqq (i, i)$$

Clearly, an $h_i \in H$ is $\preceq_p^2$-monotone, exact, and its only stable fixpoint is (i, i). For each i, there exists $\bot < i' < i$ where $S(h_{i'})(i', i') = (i', i') \prec_t^2 (i, i)$. Thus, (i, i) is not a γ-stable fixpoint of H.

In traditional AFT, a Kripke-Kleene fixpoint approximates all other fixpoints using the ordering $\preceq_p^2$. For apprx. sets, this easily follows from deterministic AFT.

Proposition 3.1. *Given an apprx. set H, For every fixpoint (x, y) of an approximator h in H, we have*

$$\mathbf{lfp}_{\preceq_p^2}(h) \preceq_p^2 (x, y)$$

4 Example: Disjunctive Logic Programs

Guided by our alternative theory, we formally establish a connection between disjunctive logic programs and normal logic programs via AFT. Namely, the partial stable models (Przymusinski 1991), and stable models of a disjunctive logic program can be expressed by the stable fixpoints of a collection of induced normal programs.

First, we introduce some preliminary definitions for disjunctive logic programs. A *disjunctive logic program* (DLP) $\mathcal{P}$ is a set of rules. A rule r is an object with three components: a head, a positive body, and a negative body, which we denote as $\mathit{head}(r)$, $\mathit{body}^+(r)$, and $\mathit{body}^-(r)$, respectively. Each component is a set of ground atoms from a propositional language. We use $Atoms(\mathcal{P})$ to denote the set of all atoms in the program $\mathcal{P}$. For a rule r where $\mathit{head}(r) = \{h_1, \ldots, h_i\}$, $\mathit{body}^+(r) = \{p_1, \ldots, p_j\}$, and $\mathit{body}^-(r) = \{n_1, \ldots, n_k\}$ we write r as follows:

$$h_1, \ldots, h_i \leftarrow p_1, \ldots, p_j, \mathbf{not}\, n_1, \ldots, \mathbf{not}\, n_k$$

We also use $\mathit{body}(r)$ to denote the entire expression to the right of the arrow above. A *normal logic program (NLP)* $\mathcal{P}$ contains only rules with a single atom in their head.

Following Belnap's logic (1977), a *(four-valued) interpretation* (T, P) of a DLP $\mathcal{P}$ is a pair of sets containing atoms from $Atoms(\mathcal{P})$. When formulating approximators for DLPs $\mathcal{P}$, we rely on the underlying complete lattice $\langle \wp(Atoms(\mathcal{P})), \subseteq \rangle$. Thus, for two interpretations (T', P') and (T, P), we have $(T', P') \preceq_p^2 (T, P)$ iff $T' \subseteq T$ and $P' \supseteq P$. The set T contains true atoms, while P contains possibly true atoms (either true or undefined). The set $T \setminus P$ contains all contradictory atoms. An interpretation (T, P) is *consistent* if $T \preceq P$. If $T = P$, then the interpretation is *two-valued*, that is, every atom is either assigned $\mathbf{t}$ or $\mathbf{f}$.

We use $\mathit{head}(\mathcal{P})$ to denote the set of all head atoms in the program $\mathcal{P}$, i.e., the set $(\bigcup_{r \in \mathcal{P}} \mathit{head}(r))$. We use $\mathit{satisfied}_{(T,P)}(\mathcal{P})$ to denote the program obtained by deleting all rules in $\mathcal{P}$ whose body is not satisfied by (T, P).

$$\mathit{satisfied}_{(T,P)}(\mathcal{P}) \coloneqq \{r \in \mathcal{P} \mid \mathit{body}^+(r) \subseteq T, \mathit{body}^-(r) \cap P = \emptyset\}$$

$$\Gamma_{\mathcal{P}}(A, B) \coloneqq \mathit{head}(\mathit{satisfied}_{(A,B)}(\mathcal{P}))$$

Above, $\mathit{satisfied}_{(T,P)}(\mathcal{P})$ checks whether the bodies of rules are either true or contradictory and if we flip T and P, it checks whether the bodies are true or undefined. $\Gamma_{\mathcal{P}}$ is the immediate consequence operator for normal programs (Denecker et al. (2000)). A consistent interpretation (T, P) is a model of $\mathcal{P}$ if it is a fixpoint of $\Gamma_{\mathcal{P}}$. We say a consistent interpretation (T', P') is a model of the reduct of $\mathcal{P}$ w.r.t. an interpretation (T, P) if $T' = \mathit{head}(\mathit{satisfied}_{(T',P)}(\mathcal{P}))$ and $P' = \mathit{head}(\mathit{satisfied}_{(P',T)}(\mathcal{P}))$.

We give a definition of partial stable models for disjunctive logic programs equivalent to Przymusinski's (1991).

Definition 4.1. *A model (T, P) of a DLP $\mathcal{P}$ is a* (partial) stable model *of $\mathcal{P}$ if there is no consistent interpretation (T', P') such that $(T', P') \prec_t^2 (T, P)$ and (T', P') is a model of the reduct of $\mathcal{P}$ w.r.t. (T, P).*

Given a DLP $\mathcal{P}$, we use $normal(\mathcal{P})$ to denote the set of all NLPs obtained by deleting some atoms from the heads of rules in $\mathcal{P}$. We formally define this below.

$$normal(\mathcal{P}) \coloneqq \{\{hc(r) \mid r \in \mathcal{P}\} \mid \exists hc, \forall r \in \mathcal{P}, \exists a \in \mathit{head}(r), hc(r) = (a \leftarrow \mathit{body}(r))\}$$

Example 3. *For the DLP* $\mathcal{P} = \{a, b \leftarrow\}$*, we have* $\mathit{normal}(\mathcal{P}) = \{\mathcal{P}_a, \mathcal{P}_b\} = \{\{a \leftarrow\}, \{b \leftarrow\}\}$.

We are now ready to capture the stable models of DLPs, by defining the apprx. set for DLPs:

Definition 4.2. *Given a DLP* $\mathcal{P}$*, we define* $H(\mathcal{P})$

$$h_{\mathcal{P}}(T, P) \coloneqq (\Gamma_{\mathcal{P}}(T, P), \Gamma_{\mathcal{P}}(P, T))$$

$$H(\mathcal{P}) \coloneqq \{h_{\mathcal{P}'} \mid \mathcal{P}' \in \mathit{normal}(\mathcal{P})\}$$

Each $h_{\mathcal{P}'}$ in $H(\mathcal{P})$ is equivalent to the approximator defined by Denecker et al ((2000)) applied to the NLP $\mathcal{P}'$. We instantiate $H(\mathcal{P})$ in the following as an example.

Example 4. *For the* $\mathcal{P}$*, from Example 3, we have*

$h_{\mathcal{P}_a}(\emptyset, \{a, b\}) = (\{a\}, \{a\})$ $\quad h_{\mathcal{P}_b}(\emptyset, \{a, b\}) = (\{b\}, \{b\})$

Denecker et al. (2000) show that $S(h_{\mathcal{P}'})$ is consistent, that is, $S(h_{\mathcal{P}'})$ maps consistent pairs to consistent pairs. We say an apprx. set H is *stable-consistent* if $S(h)$ is consistent for each $h \in H$.

Proposition 4.1. $H(\mathcal{P})$ *is a stable-consistent apprx. set.*

We intend to show that the γ-stable fixpoints of $H(\mathcal{P})$ correspond to the stable models of $\mathcal{P}$. First, we take a brief detour to establish connections between DLPs and their associated normal programs. Once these connections are in place, our goal is almost immediate.

Following Heyninck et al. (2024), we call a model (T, P) of $\mathcal{P}$ a *weakly supp. (weakly supported) model* if for every atom in $a \in T$ there exists a rule $r \in \mathit{satisfied}_{(T,P)}(\mathcal{P})$ where $a \in \mathit{head}(r)$ and for every atom $a \in P$, there exists a rule $r \in \mathit{satisfied}_{(P,T)}(\mathcal{P})$ where $a \in \mathit{head}(r)$. We call such a model *strongly supportd*, if every atom in $a \in T$ there exists a rule $r \in \mathit{satisfied}_{(T,P)}(\mathcal{P})$ where $\{a\} = \mathit{head}(r) \cap T$ and for every atom $a \in P$, there exists a rule $r \in \mathit{satisfied}_{(P,T)}(\mathcal{P})$ where $\{a\} = \mathit{head}(r) \cap P$. Strong support can be linked to Clark's completion lifted to DLPs (Lee and Lifschitz 2003) and lifted to three-valued semantics. For convenience, we refer to a model (T, P) of a DLP $\mathcal{P}$ that is not a stable model of $\mathcal{P}$ as an *unstable model*.

We draw the connections between each model type for DLPs and the models of their associated normal programs.

Lemma 4.1. *A model of* $\mathcal{P}$ *is a model of some* $\mathcal{P}' \in \mathit{normal}(\mathcal{P})$*. If it is a strongly supported, stable, or unstable model of* $\mathcal{P}$*, then it is respectively a strongly supported, stable, or unstable model of some* $\mathcal{P}'' \in \mathit{normal}(\mathcal{P})$*.*

Lemma 4.2. *A model of* $\mathcal{P}' \in \mathit{normal}(\mathcal{P})$ *is a model of the DLP* $\mathcal{P}$*. If it is a weakly supported or unstable model of* $\mathcal{P}'$*, then it is respectively a weakly supported or unstable model of* $\mathcal{P}$*.*

Each lemma comes rather naturally if we consider forming a disjunctive logic program from a normal logic program by adding additional atoms to the heads of rules, or for the other direction, deleting atoms from the heads of rules.

We visualize both Lemmas below.

(T, P) is a	DLP $\mathcal{P}$ to NLP $\mathcal{P}'$
model	$\iff$
weakly supp. model	$\Leftarrow$
strongly supp. model	$\Rightarrow$
unstable model	$\iff$
stable model	$\Rightarrow$

Lemmas 4.1 and 4.2 immediately provide a method of verifying whether an interpretation (T, P) is a stable model of a DLP $\mathcal{P}$. First, to confirm that (T, P) is a model of $\mathcal{P}$, we find a program $\mathcal{P}' \in \mathit{normal}(\mathcal{P})$ s.t. (T, P) is a model of $\mathcal{P}'$. Next, we locate a program $\mathcal{P}' \in \mathit{normal}(\mathcal{P})$ s.t. (T, P) is a stable model of $\mathcal{P}'$. Finally, we confirm that there is no program $\mathcal{P}' \in \mathit{normal}(\mathcal{P})$ s.t. (T, P) is an unstable model of $\mathcal{P}$ so that we can conclude (T, P) is a stable model of $\mathcal{P}$.

With this method in hand, we need only link individual notions (models, weakly supp. models, unstable models, etc.) to apprx. sets to capture stable semantics.

Lemma 4.3. *Let* $\mathcal{P}$ *be a DLP and* (T, P) *a consistent interpretation. Each row below independently resolves some* $h \in H(\mathcal{P})$ *to form a separate claim.*

(T, P) *is a this of* $\mathcal{P}$	*if/iff*	(T, P) *is a*
model	$\iff$	$\preceq_t^2$*-prefixpoint of* h
weakly supp. model	$\Leftarrow$	*fixpoint of* h
strongly supp. model	$\Rightarrow$	*fixpoint of* h
unstable model	$\iff$	*model of* $\mathcal{P}$ *and a* $\prec_t^2$*-prefixpoint of* $S(h)$

Most of the above claims come rather immediately, with the exception of the claim on unstable models. Intuitively, a single application of the stable revision operator computes the least model of the reduct of a normal program w.r.t. the given interpretation. The asymmetry of weakly and strongly supported models is surprising but is made apparent later in Example 5.

When we combine the links above with the aforementioned algorithm for computing (partial) stable models, we arrive at γ-stable fixpoints.

Theorem 4.1. *The* γ*-stable fixpoints of* $H(\mathcal{P})$ *are equivalent to the (partial) stable models of* $\mathcal{P}$*.*

Proof. ($\Rightarrow$) Let (T, P) be a γ-stable fixpoint of $H(\mathcal{P})$. It is a stable fixpoint of some $h_{\mathcal{P}'} \in H(\mathcal{P})$ and a stable model of $\mathcal{P}'$ (Denecker et al. (2000)), and thus a model of $\mathcal{P}$ (Lemma 4.2). Suppose, for the sake of contradiction, that (T, P) is an unstable model of $\mathcal{P}$, then there exists $h_{\mathcal{P}''} \in H(\mathcal{P})$ such that $S(h_{\mathcal{P}''})(T, P) \prec_t^2 (T, P)$ (Lemma 4.3), which directly contradicts the assumption that $(T, P) \in \boldsymbol{min}_{\preceq_t^2}\{S(h)(T, P) \mid h \in H(\mathcal{P})\}$. It follows that (T, P) is not an unstable model of $\mathcal{P}$ and, as a model of $\mathcal{P}$, must be a stable model of $\mathcal{P}$.

($\Leftarrow$) Let (T, P) be a stable model of $\mathcal{P}$. It is a stable model of some $\mathcal{P}' \in \mathit{normal}(\mathcal{P})$ (Lemma 4.1), a stable fixpoint of $S(h_{\mathcal{P}'})$, and fixpoint of $h_{\mathcal{P}'}$ (Denecker et al. (2000)). As a stable model, (T, P) is not an unstable model of $\mathcal{P}$, thus by contrapositive, it is not a $\prec_t^2$-prefixpoint of $S(h_{\mathcal{P}''})$ for any $\mathcal{P}'' \in \mathit{normal}(\mathcal{P})$ (Lemma 4.3). We arrive at $(T, P) \in \boldsymbol{min}_{\preceq_t^2}\{S(h)(T, P) \mid h \in H(\mathcal{P})\}$. □

Answer set semantics (i.e., two-valued stable model semantics) and the well-founded model are special cases of partial stable models.

Proposition 4.2. *A* γ*-stable fixpoint* (T, P) *of* $H(\mathcal{P})$ *is equivalent to an*

1. Answer set of $\mathcal{P}$ *if* $T = P$*, or a*

2. *Well-founded model of each $\mathcal{P}' \in normal(\mathcal{P})$ and stable model of $\mathcal{P}$ if* $\mathbf{lfp}_{\preceq_p^2} S(h) = (T, P)$ *for each* $h \in H(\mathcal{P})$

Proof. (1) Partial stable models are faithful to two-valued stable models (Przymusinski 1991). (2) ($\Rightarrow$) $\mathbf{lfp}_{\preceq_p^2} S(h)$ exists by Denecker et al. (2000) and is the well-founded model. ($\Leftarrow$) As a fixpoint of every stable operator, it is a γ-stable fixpoint. □

We finish this section with a few examples to concretize our definitions and shine light on a few subtler details.

Example 5. *For normal programs, both types of supported models are equivalent, that is, every weakly supp. model of a program $\mathcal{P}' \in normal(\mathcal{P})$ is a strongly supp. model of $\mathcal{P}'$. Define $\mathcal{P}$ as a program containing two rules such that $\mathcal{P} = \{(a, b, c \leftarrow), (a, b, d \leftarrow)\}$. We have a $\mathcal{P}' \in normal(\mathcal{P})$ where $\mathcal{P}' = \{a \leftarrow, b \leftarrow\}$. The interpretation $(\{a, b\}, \{a, b\})$ is a strongly supp. model of $\mathcal{P}'$, but only a weakly supp. model of $\mathcal{P}$. Now, using $\mathcal{P} = \{a, b \leftarrow\}$ we have that $(\{a, b\}, \{a, b\})$ is a weakly supp. model of $\mathcal{P}$, but not a weakly supp. model of any $\mathcal{P}' \in normal(\mathcal{P})$.*

We adopt the following example from Heyninck et al. (2024) for a DLP with no partial stable model.

Example 6. *Let $\mathcal{P} = \{(a, b, c \leftarrow), (a \leftarrow \mathbf{not}\, b), (b \leftarrow \mathbf{not}\, c), (c \leftarrow \mathbf{not}\, a)\}$. We have $H(\mathcal{P}) = \{h_{\mathcal{P}_a}, h_{\mathcal{P}_b}, h_{\mathcal{P}_c}\}$ s.t. $(a \leftarrow) \in \mathcal{P}_a$ and so on. While each approximator has a single $\preceq_p^2$-least stable fixpoint $((\{a, b\}, \{a, b\}), (\{b, c\}, \{b, c\})$, and $(\{c, a\}, \{c, a\}))$ each is a $\prec_t^2$-prefixpoint of another stable operators. Thus, $H(\mathcal{P})$ has no γ-stable fixpoint.*

We've linked deterministic approximators to partial stable semantics, two types of supported models, well-founded models, and answer sets.

5 Relation to Nondeterministic AFT

We now relate apprx. sets from Section 3 to Heyninck et al.'s nondeterministic AFT (2024). We introduce stable revision in their theory now.

Notably, a nondeterministic operator returns a set given a pair. Thus, we introduce an ordering over sets similar to $\preceq_p^2$.

Definition 5.1. *Given a complete lattice $\langle \mathcal{L}, \preceq \rangle$, its bilattice $\langle \mathcal{L}^2, \preceq_p^2 \rangle$ induces a preorder $\precsim_p^2$ over $\wp^o(\mathcal{L})^2$.*

$$(X', Y') \precsim_p^2 (X, Y) \text{ iff } ((X' \uparrow) \supseteq X) \wedge ((Y' \downarrow) \supseteq Y)$$

While the ordering above lacks antisymmetry, it functions enough like a partial order for our purposes. We note that the ordering only inflates (via $\uparrow$ / $\downarrow$) the pair (X', Y').[1]

Heyninck et al.'s nondeterministic approximators ((2024)) generalize deterministic approximators (Denecker et al. (2000)); We recall them now:

Definition 5.2. *Given a complete lattice $\langle \mathcal{L}^2, \preceq \rangle$ A* nondeterministic approximator (or nondeterministic approximating operator (ndao)) $o : \mathcal{L}^2 \to \wp^o(\mathcal{L})^2$ *is*

1. *Order-preserving from $\langle \mathcal{L}^2, \preceq_p^2 \rangle$ to $\langle \wp^o(\mathcal{L})^2, \precsim_p^2 \rangle$*
2. Exact*: $\forall x \in \mathcal{L}, o(x, x)_1 = o(x, x)_2$*

We call a pair $(x, y) \in \mathcal{L}$ an *n-prefixpoint* of an ndao o if $o(x, y) \precsim_p^2 (\{x\}, \{y\})$ An *n-fixpoint* of o is a pair (x, y) s.t. $x \in o(x, y)_1$ and $y \in o(x, y)_2$.

Heyninck et al. (2024) generalize the standard AFT definition of the stable revision operator to ndaos as follows.

Definition 5.3. *Given an ndao $o : \mathcal{L}^2 \to \wp^o(\mathcal{L})^2$, its* nondeterministic stable revision operator $S(o)$ *follows.*

$$C^o_{low}(y) \coloneqq \{\phi \mid \phi \in o(\phi, y), \forall(\phi' \prec \phi), (\phi' \notin o(\phi', y))\}$$
$$C^o_{high}(x) \coloneqq \{\phi \mid \phi \in o(x, \phi), \forall(\phi' \prec \phi), (\phi' \notin o(x, \phi'))\}$$
$$S(o)(x, y) \coloneqq (C^o_{low}(y), C^o_{high}(x))$$

We call n-fixpoints of nondeterministic stable operators n-stable fixpoints

Comparing Frameworks

A direct comparison of frameworks is challenged by differences in how the frameworks treat consistency and exactness and the difference in the underlying orderings. An ndao o has a more relaxed notion of exactness and consistency because some non-exact pairs (resp. inconsistent pairs) may occur in $o(x, y)_1 \times o(x, y)_2$ where $x = y$ (resp. $x \preceq y$).

Example 7. *For the constant ndao o equal to $(\{\bot, \top\}, \{\bot, \top\})$, we have $(\top, \bot) \in o(\top, \top)_1 \times o(\top, \top)_2$.*

Every apprx. set H can be converted to a function of the same type as Heyninck et al.'s (2024) ndaos.

Definition 5.4. *We say an apprx. set H* induces *a function $o : \mathcal{L}^2 \to \wp^o(\mathcal{L})^2$ s.t.*

$$o(x, y) \coloneqq (\{h(x, y)_1 \mid h \in H\}, \{h(x, y)_2 \mid h \in H\})$$

The "induces" relation is the glue that enables us to connect apprx. sets with Heyninck et al.'s ndaos. An approximator's monotonicity and exactness both carry the order-preserving and exactness requirements of an ndao when the set is converted, thus the following comes easily.

Proposition 5.1. *Given an apprx. set H, its induced function is an ndao.*

Note that this relation is not bijective, i.e., many apprx. sets may induce the same function o and for some ndao o, there may be no apprx. sets that induce o (Discussed further in the next subsection). However, every apprx. set induces an ndao. We link stable fixpoints between frameworks.

Theorem 5.1. *Given an ndao o and an apprx. set H, an n-stable fixpoint (x, y) is an γ-stable fixpoint of H if*

- *H induces o, and*
- *(x, y) is a fixpoint of some $h \in H$*

Proof. Let $h \in H$ be the approximator s.t. (x, y) is a fixpoint of h. Because (x, y) is an n-stable fixpoint, we have that there does not exist $x' \prec x$ s.t. $x' \in o(x', y)$. Because H induces o, there does not exist $h' \in H$ s.t. $x' = h'(x', y)$. With $h' = h$, it follows that $x = \mathbf{lfp}\ h(\cdot, y)$. Similarly, we can show $y = \mathbf{lfp}\ h(x, \cdot)$ and conclude $S(h)(x, y) = (x, y)$. Furthermore, we can conclude there does not exist $h' \in H$ s.t. $S(h')(x, y) \prec_t^2 S(h)(x, y)$. □

[1] Later (Definition 5.6), we use ordering $(X', Y') \precsim_p^2 (X, Y)$ which inflates (via $\uparrow$ / $\downarrow$) the pair (X, Y) instead of (X', Y').

The opposite direction of the above theorem does not hold, i.e., not every γ-stable fixpoint is an n-stable fixpoint. This is by design (it's sufficient to capture partial stable models of DLPs). We can narrow γ-stable fixpoint so that there is a bidirectional relationship with n-stable fixpoints.

Definition 5.5. *A pair (x, y) is a α-*stable fixpoint *if it is a fixpoint of some $h \in H$ and for each $h' \in H$, neither hold*

$$\textit{(i.) } S(h')(x,y)_1 \prec x \quad \textit{(ii.) } S(h')(x,y)_2 \prec y$$

Every α-stable fixpoint of H is a stable fixpoint of an approximator $h \in H$, thus linking this definition to deterministic AFT. This can easily shown by using $h = h'$. We can link α-stable fixpoints directly to ndaos.

Theorem 5.2. *An α-stable fixpoint (x, y) of an apprx. set H is an n-stable fixpoint of the ndao o induced by H.*

Proof. We have

$$\begin{aligned} x \in \boldsymbol{min}_{\preceq}\{\mathbf{lfp}_{\preceq}\, h(\cdot,y)_1 \mid h \in H\} \\ o(x,y)_1 = \{h(x,y)_1 \mid h \in H\} \\ C^o_{low}(y) = \{\phi \mid \phi = h(\phi,y)_1, \forall \phi' \prec \phi, \\ \neg\exists h', \phi' = h'(\phi', y)\} \end{aligned}$$

With a few steps, the first line can be converted to the third. The other side of the operator can be shown easily as well. □

We connect α-stable fixpoints with γ-stable fixpoints.

Proposition 5.2. *An α-stable fixpoint of a stable-consistent apprx. set H is a γ-stable fixpoint of H.*

Proof. When considering the contrapositive, observe that the condition $S(h')(x', y')_1 \prec S(x, y)_1$ or $S(h')(x', y')_2 \prec S(x, y)_2$ and weaker than $S(h')(x', y') \prec^2_t S(x, y)$. There are more γ-stable fixpoints than α-stable fixpoints. □

With Theorems 5.1, 5.2 and Proposition 5.2, we have a bidirectional link between frameworks.

Corollary 5.3. *Given a stable-consistent apprx. set H, a fixpoint (x, y) of some $h \in H$ is an α-stable fixpoint of H iff it's an n-stable fixpoint of the ndao induced by H.*

The truth-minimality of α-stable fixpoints follows from the above claim and the truth minimality of n-stable fixpoints (Heyninck et al. (2024)). α-Stable fixpoints bridge the gap between n-stable fixpoints and γ-stable fixpoints and demonstrates the flexibility of our framework. The other direction of Proposition 5.2 does not hold, namely, α-stable fixpoints do not capture all partial stable models of DLPs.

Example 8. *Define $\mathcal{P}$ as $(a, b \leftarrow), (a \leftarrow \textbf{not}\, a)$ $(\{b\}, \{a, b\})$ is a partial stable model of $\mathcal{P}$ and a γ-stable fixpoint of $H(\mathcal{P})$, but it is not an α-stable fixpoint of $H(\mathcal{P})$.*

Due to the link between α-stable fixpoints and n-stable fixpoint, α-stable fixpoint do capture the two-valued stable model of DLPs (Heyninck et al. 2024).

Framework Incompatibilities

Not every ndao can be represented as an apprx. set. We now discuss some incompatibilities between the frameworks and argue that these differences are insignificant in nonmonotonic systems.

The first difference stems from comparing the ordering over sets induced by a set of $\preceq^2_p$-monotone approximators with $\precsim^2_p$. Apprx. sets induce ndaos with an image that respects $\precsim^2_p$ (introduced below), whereas ndaos are only required to be $\preceq^2_p$-monotone (Definition 5.1) (distinct from $\precsim^2_p$-monotonicity which is introduced below).

Definition 5.6.

$$\begin{aligned} (X', Y') \precsim^2_p (X, Y) \textit{ iff } ((X' \uparrow) \supseteq X) \wedge ((Y' \downarrow) \supseteq Y) \\ (X', Y') \precsim^2_p (X, Y) \textit{ iff } (X' \subseteq (X \downarrow)) \wedge (Y' \subseteq (Y \uparrow)) \\ A \lesssim^2_p B \textit{ iff } (A \preceq^2_p B) \wedge (A \precsim^2_p B) \end{aligned}$$

A nondeterministic approximator $o : \mathcal{L}^2 \to \wp^o(\mathcal{L})^2$ that is not order-preserving from $\preceq^2_p$ to $\precsim^2_p$ does not have an inducing apprx. set. We briefly demonstrate the case where a ndao does not respect the ordering $\precsim^2_p$.

Example 9. *Define the complete lattice $\langle \mathcal{L}, \preceq \rangle$ s.t. $\mathcal{L} = \{\bot, a_1, a_2, b_1, b_2, \top\}$ where $a_1 \preceq a_2$ and $b_1 \preceq b_2$.*

$$o(x,y)_1 = \begin{cases} \{a_1, b_1\} & \textit{if } x \in \{a_1, b_1\} \\ \{a_2\} & \textit{if } x \in \{a_2, b_2\} \\ \{x\} & \textit{otherwise} \end{cases}$$

where $o(x, y)_2$ is $o(y, x)_1$ for exact pairs, $\{\top\}$ for consistent pairs and $\{\bot\}$ otherwise. o can be quickly verified to be exact and order-preserving from $\preceq^2_p$ to $\precsim^2_p$. However, o is not order-preserving from $\preceq^2_p$ to $\precsim^2_p$ as we have $b_1 \in o(b_1, \top)_1$ but $o(b_2, \top)_1 = \{a_2\}$.

The class of ndaos without this stronger ordering does not include symmetric operators (Remark 11 from Heyninck et al. (2024)). Thus, we arrive at a weak condition regarding the non-existence of an inducing apprx. set.

Corollary 5.4. *If a non-symmetric ndao is not order-preserving from $\preceq^2_p$ to $\precsim^2_p$, then it does not have an inducing apprx. set.*

This identifies a populated class of ndaos that do not have corresponding apprx. sets. However, this is not much of a limitation as the underlying lattice $\mathcal{L}$ can be easily tweaked to preserve this ordering (e.g., by adding $\top$ to every set).

Even with the stronger ordering, not every $\precsim^2_p$ order-preserving ndao has an inducing apprx. set. Intuitively, to construct an apprx. set from an ndao, each element in the image of the ndao must be mapped to a deterministic approximator. We can construct an ndao over an infinite lattice s.t. any such mapping will violate an approximator's $\preceq^2_p$-monotonicity.

Example 10. *Have $o : \mathcal{L}^2 \to \wp^o(\mathcal{L})$ be an ndao such that there exists a sublattice $\{\top, a, b, \bot\}$ in $\mathcal{L}$ where a and b are incomparable and o has the following property: There exists*

an element $x \in o(\top,\top)_1$, such that there is only one element $x' \in o(a,\top)_1$ s.t. $x' \preceq^2_p x$. Similarly, there is only one element $x'' \in o(\bot,\top)_1$ s.t. $x'' \preceq^2_p x'$. Continuing, there is only one element $x''' \in o(b,\top)_1$ s.t. $x''' \succeq^2_p x'$. We can have x''' and x to be incomparable, while maintaining the underlying lattice structure by continuing this spiral infinitely. That is, there exists only one element $x'''' \in o(\top,\top)_1$ s.t. $x'''' \succeq^2_p x'''$, and so on. For o to have an inducing apprx. set, there must exist an approximator h such that $h(\top,\top)_1 = x$ and $h(a,b)_1 \in o(x,y)_1$ for every $(a,b) \in \mathcal{L}^2$. However, we are forced to assign $h(b,\top)_1$ to an element that is not comparable to x, thus h is not $\preceq^2_p$-monotone.

To summarize, we have linked apprx. sets and ndaos and shown a correspondence between each framework's notion of a stable fixpoint. We have also shown that some ndaos do not have an inducing apprx. set while every apprx. set has an ndao. When we narrow our notion of stable revision (α-stable fixpoints) there is a conditional one-to-one correspondence with n-stable fixpoints. In Section 4, we demonstrated that our framework can, like ndaos, be applied to DLPs, thus the gap between frameworks is not pragmatically limiting.

6 Related Work

Nondeterministic AFT was introduced only recently by Heyninck et al. (2024). To the best of our knowledge, the idea of capturing nondeterministic operators using sets of deterministic operators has not been investigated algebraically.

Several semantics for disjunctive logic programs rely on a transformation from disjunctive to normal logic programs. The *possible world semantics* (Sakama 1990) simply takes the stable models of programs obtained by replacing every disjunction with one of its disjuncts. Sakama (1990) shows that these semantics do not coincide with the stable model semantics for DLPs, which also illustrates the difference to our semantics, as we represent the latter. However, it is not hard to see that the total γ-stable fixpoints of $H(\mathcal{P})$ are all possible models of $\mathcal{P}$ according to Sakama (1990):

Proposition 6.1. *If (T,T) is a γ-stable fixpoint of $H(\mathcal{P})$ then it is a possible model according to Sakama (1990).*

Proof. Recall that every stable model of a $\mathcal{P}' \in \mathit{normal}(\mathcal{P})$ is a possible model of $\mathcal{P}$ as defined by Sakama (1990). Suppose now that (T,T) is a γ-stable fixpoint of $H(\mathcal{P})$. Then with Theorem 4.1, (T,T) is a stable model of $\mathcal{P}$. With Lemma 4.1, it is a stable model of some $\mathcal{P}' \in \mathit{normal}(\mathcal{P})$. □

Sakama (1992) defines well-founded semantics for disjunctive logic programs, based on the same idea as the possible models. However, he uses a different notion of well-founded semantics, based on a five-valued logic. These semantics are not faithful to traditional well-founded semantics for normal logic programs (Van Gelder, Ross, and Schlipf 1991), whereas we take a conservative approach.

Shen and Eiter (2019) define the *determining inference semantics* by transforming a DLP into a normal logic program and selecting minimal stable models of those normal logic programs. Again, it is shown that these semantics differ from the stable semantics for DLPs. In more detail, we obtain the following results:

Proposition 6.2. *(1) Given a DLP $\mathcal{P}$, if (T,T) is a γ-stable fixpoint of $H(\mathcal{P})$ then it is a DI-answer set (w.r.t. GL-semantics) according to Shen and Eiter (2019). (2) There exist DLPs $\mathcal{P}$ s.t. T is a DI-answer of $\mathcal{P}$ set but not a γ-stable fixpoints of $H(\mathcal{P})$*

Proof. Ad (1). This follows immediately from Corollary 1 of Shen and Eiter (2019), where it is shown that stable model of a DLP[2] $\mathcal{P}$ is a DI-answer set (w.r.t. GL-semantics), and by Theorem 4.1 that shows every γ-stable fixpoint of $H(\mathcal{P})$ is a stable model of $\mathcal{P}$. Ad (2). According to Example 6 of Shen and Eiter (2019), there is a DI-answer set that is not a stable model, which means with Theorem 4.1 that this DI-answer set cannot be a γ-stable fixpoint of $H(\mathcal{P})$. □

Note that Shen and Eiter (2019) only treat two-valued semantics. We leave generalizing their work to three-valued semantics to future work.

7 Discussion and Future Work

Nondeterministic AFT by (Heyninck et al. 2024) is an important extension of AFT, enabling the capture of nondeterministic semantics. Heyninck et al. demonstrate that an ndao can express a single deterministic approximator. It is natural to ask the converse: whether a set of deterministic approximators can express an ndao. We formulate apprx. sets by reusing deterministic AFT definitions to capture n-stable fixpoints and, more concretely, partial stable models of DLPs. Our work emphasizes the semantic connection between normal logic programs and disjunctive logic programs - namely that we can formulate disjunctive partial stable models in terms of a family of induced normal logic programs. We generalize this idea to the algebraic level using approximators to formulate a general theory.

In future work, we intend to investigate whether this approach makes it easier to lift new semantic developments in AFT to nondeterministic AFT. For example, Bi et al. (2014) explore a variation of AFT that relaxes the exactness condition, which Liu and You (2022) show can be applied to hybrid MKNF knowledge bases. Inconsistent pairs play a critical role in this theory. Thus, the theory cannot be adapted to ndaos, which deal exclusively with consistent pairs. Other avenues for future work include looking at other formalisms incorporating nondeterminism, such as choice programs (Heyninck 2024; Alviano, Faber, and Gebser 2023) or disjunctive default logic (Gelfond et al. 1991).

The relationship between normal and disjunctive programs, highlighted by our framework, offers an opportunity to study the properties of disjunctive systems in terms of the properties of their normal counterparts. For example, the reduction of a DLP to a normal program bears similarity to backdoors to normality (Fichte and Szeider 2015), which have been shown to have a strong impact on the reduction of complexity of numerous problems.

[2]Shen and Eiter (2019) call DLPs "simple disjunctive logic programs" . Generalizing to their full language is left for future work.

Acknowledgments

Spencer Killen was partially supported by Alberta Innovates and Alberta Advanced Education. Jesse Heyninck was partially supported by the project LogicLM: Combining Logic Programs with Language Model with file number NGF.1609.241.010 of the research programme NGF AiNed XS Europa 2024-1 which is (partly) financed by the Dutch Research Council (NWO).

References

Alviano, M.; Faber, W.; and Gebser, M. 2023. Aggregate semantics for propositional answer set programs. *Theory and Practice of Logic Programming*, 23(1): 157–194.

Antic, C.; Eiter, T.; and Fink, M. 2013. Hex Semantics via Approximation Fixpoint Theory. In Cabalar, P.; and Son, T. C., eds., *Logic Programming and Nonmonotonic Reasoning, 12th International Conference, LPNMR 2013, Corunna, Spain, September 15-19, 2013. Proceedings*, volume 8148 of *Lecture Notes in Computer Science*, 102–115. Springer.

Belnap, N. D. 1977. *A Useful Four-Valued Logic*, 5–37. Dordrecht: Springer Netherlands. ISBN 978-94-010-1161-7.

Bi, Y.; You, J.-H.; and Feng, Z. 2014. A Generalization of Approximation Fixpoint Theory and Application. In Kontchakov, R.; and Mugnier, M.-L., eds., *Web Reasoning and Rule Systems*, 45–59. Cham: Springer International Publishing.

Denecker, M.; Marek, V.; and Truszczyński, M. 2000. Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning. In *Logic-Based Artificial Intelligence*, 127–144. Springer.

Denecker, M.; Marek, V. W.; and Truszczynski, M. 2003. Uniform semantic treatment of default and autoepistemic logics. *Artif. Intell.*, 143(1): 79–122.

Fichte, J. K.; and Szeider, S. 2015. Backdoors to Normality for Disjunctive Logic Programs. *ACM Trans. Comput. Log.*, 17(1): 7.

Gelfond, M.; Lifschitz, V.; Przymusinska, H.; and Truszczynski, M. 1991. Disjunctive defaults. In *Proc. Second International Conf. on Principles of Knowledge Representation and Reasoning*, 230–237.

Heyninck, J. 2024. Operator-based semantics for choice programs: is choosing losing? (full version). *arXiv*, (2407.21556). A short version to appear in KR'24.

Heyninck, J.; Arieli, O.; and Bogaerts, B. 2024. Nondeterministic approximation fixpoint theory and its application in disjunctive logic programming. *Artificial Intelligence*, 331: 104110.

Heyninck, J.; and Bogaerts, B. 2023. Non-deterministic Approximation Operators: Ultimate Operators, Semi-equilibrium Semantics, and Aggregates. *Theory Pract. Log. Program.*, 23(4): 632–647.

Lee, J.; and Lifschitz, V. 2003. Loop Formulas for Disjunctive Logic Programs. In Palamidessi, C., ed., *Logic Programming, 19th International Conference, ICLP 2003, Mumbai, India, December 9-13, 2003, Proceedings*, volume 2916 of *Lecture Notes in Computer Science*, 451–465. Springer.

Liu, F.; and You, J. 2022. Alternating Fixpoint Operator for Hybrid MKNF Knowledge Bases as an Approximator of AFT. *Theory Pract. Log. Program.*, 22(2): 305–334.

Marynissen, S.; Bogaerts, B.; and Denecker, M. 2021. On the Relation Between Approximation Fixpoint Theory and Justification Theory. In Zhou, Z., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, 1973–1980. ijcai.org.

Marynissen, S.; Bogaerts, B.; and Denecker, M. 2024. Embedding justification theory in approximation fixpoint theory. *Artificial Intelligence*, 331: 104112.

Przymusinski, T. C. 1991. Stable Semantics for Disjunctive Programs. *New Gener. Comput.*, 9(3/4): 401–424.

Roman, S. 2008. *Lattices and Ordered Sets*. Springer New York. ISBN 978-0-387-78900-2 978-0-387-78901-9.

Sakama, C. 1990. Possible model semantics for disjunctive databases. In *Deductive and Object-Oriented Databases*, 369–383. Elsevier.

Sakama, C. 1992. Extended Well-Founded Semantics for Paraconsistent Logic Programs. In *FGCS*, 592–599.

Shen, Y.-D.; and Eiter, T. 2019. Determining inference semantics for disjunctive logic programs. *Artificial Intelligence*, 277: 103165.

Tarski, A. 1955. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2): 285 – 309.

Van Gelder, A.; Ross, K. A.; and Schlipf, J. S. 1991. The well-founded semantics for general logic programs. *Journal of the ACM (JACM)*, 38(3): 619–649.

Vanbesien, L.; Bruynooghe, M.; and Denecker, M. 2022. Analyzing Semantics of Aggregate Answer Set Programming Using Approximation Fixpoint Theory. *Theory Pract. Log. Program.*, 22(4): 523–537.