Using AFT to Characterize Shen and Eiter's Disjunctive Logic Program Semantics

Spencer Killen[1,*], Jia-Huai You[1]

[1]University of Alberta, 11011 - 88 Avenue, Edmonton, AB, Canada, T6G 2G5

Abstract

Disjunction in knowledge representation concisely expresses uncertainty as sets of possibilities. Due to the nondeterministic nature of these sets, we require specialized tools to establish the meaning of languages that incorporate disjunction. In this work, we further prove the effectiveness of one such tool: Our recent extension of approximation fixpoint theory (AFT) to disjunctive semantics. This approach has the advantage of simplicity because it directly utilizes the existing objects and definitions of AFT. To exercise our approach, we capture *the determining inference semantics* of Shen and Eiter, a class of disjunctive semantics where any non-disjunctive answer set semantics is extended to disjunctive logic programs. This leads to a fixpoint characterization of this class of disjunctive semantics. Additionally, we extend determining inference semantics to the three-valued case. In the case of disjunctive logic programs, we combine the determining inference stable models with partial stable models. A combination of these semantics that is faithful presents a challenge due to incompatibility concerning truth minimality. However, due to the flexibility of our approach, we are able to remedy this challenge by introducing a new truth ordering.

1. Introduction

Approximation fixpoint theory [1] can not accommodate disjunctive semantics. As a result, various approaches [2, 3] have emerged that capture specific nondeterministic semantics by formulating operators that map sets to other sets. Heyninck et al. [4] recently developed a general theory of nondeterministic AFT that generalizes such an approach. Their theory is highly suitable for characterizing disjunctive semantics [4, 5, 6]. However, it departs significantly from standard AFT by introducing its own notion of fixpoints, exactness, monotonicity, etc., which leads to increased challenges in transitioning non-disjunctive semantics to a nondeterministic setting. In prior work [7], we offer an alternative to Heyninck et al.'s approach, called an *approximator set*, which leverages a set of operators rather than a single operator.

Using a family of operators has a few interesting benefits. Firstly, the definitions and objects of deterministic AFT can be directly reused in this lifted theory. Unlike nondeterministic AFT [4], there is no need to introduce separate notions of fixpoints or monotonicity. This reduces the overhead of applying AFT nondeterministically, and it can also tighten the relationship between nondeterministic semantics and their deterministic counterparts. To capture a nondeterministic semantics using a set of operators is to break the semantics down into a family of deterministic semantics, each captured by its own operator. Because many deterministic semantics have been treated by AFT, there are many operators that can be reused in a nondeterministic setting. For example, a disjunctive logic program can be expressed as a family of normal logic programs. We can capture the disjunctive semantics using the set of operators defined for the normal logic programs and an operation to merge these semantics [7].

By parameterizing this merge operation, the approximator set approach is highly flexible and a question is raised of what other disjunctive semantics we can capture with this approach. In this work, we show that we can capture the determining inference semantics of disjunctive logic programs by

23rd International Workshop on Nonmonotonic Reasoning, November 11-13, 2025, Melbourne, Australia

*Corresponding author.

sjkillen@ualberta.ca (S. Killen); jyou@cs.ualberta.ca (J. You)

https://sjkillen.ca (S. Killen)

0000-0003-3930-5525 (S. Killen); 0000-0001-9372-4371 (J. You)

tweaking the merge operation for the set of operators for partial stable models. The relative ease with which this is accomplished further supports our approach.

Shen and Eiter [8] introduced determining inference semantics as a relaxation of Gelfond and Lifschitz's answer set semantics [9] ($\mathcal{SM}$ semantics). Borrowing from their example (Example 1), the following disjunctive logic program has no answer sets under $\mathcal{SM}$ semantics.

$$a \leftarrow \qquad\qquad a, b \leftarrow \qquad\qquad b \leftarrow \textbf{\textit{not}}\ b$$

We can separate the program into two non-disjunctive programs: (a) one which replaces $a, b \leftarrow$ with $a \leftarrow$, effectively "choosing" the atom a, and (b) the program which "chooses" b. Under $\mathcal{SM}$ semantics, program (a) does not have an answer set and program (b) has a single answer set $\{a, b\}$ which assigns both a and b to be true. Disjunction in DLPs is not equivalent to disjunction in classical logic [10]. Shen and Eiter [8] argue that determining inference semantics is more faithful to the constructive nature of disjunction as used in logic programming. They argue that the additional answer sets the semantics brings are reasonably acceptable. The determining inference semantics provides a mechanism to lift any semantics defined for non-disjunctive logic programs to disjunctive programs. By demonstrating that approximator sets [7] can capture determining inference semantics, we also show that approximator sets can be used as a means to lift any approximator defined for non-disjunctive programs to disjunctive programs.

This paper reports the results of our investigation, and the main contributions are as follows:

1. Under reasonable syntactic restrictions, we successfully apply approximator sets [7] to capture the DI-$\mathcal{X}$ semantics [8], a family of disjunctive semantics induced by a non-disjunctive semantics $\mathcal{X}$. This demonstrates the flexibility of approximator sets and provides an automatic method of lifting non-disjunctive fixpoint semantics to disjunctive programs.
2. We examine the three-valued fixpoints in our approach to identify a three-valued DI-$\mathcal{X}$. The newly formulated DI-$\mathcal{PSM}$ semantics are faithful to both DI-$\mathcal{SM}$ and $\mathcal{PSM}$ semantics.

The paper is organized as follows. In Section 2, we cover the necessary preliminaries and establish notation for disjunctive logic programs (Section 2.1), approximation fixpoint theory (Section 2.2), and determining inference semantics (Section 2.3). In Section 3 we apply approximator sets to determining inference semantics and establish a three-valued semantics. Finally, we conclude and discuss in Section 4.

2. Preliminaries

2.1. Disjunctive Logic Programs

A *disjunctive logic program* (DLP) $\mathcal{P}$ is a set of rules. Each rule $r \in \mathcal{P}$ consists of a set of atoms called its head, denoted $hd(r)$, and a set of possibly negated atoms called its body, denoted $body(r)$. A rule r with the parts $hd(r) = \{h_1, \ldots, h_n\} \neq \emptyset$ and $body(r) = \{p_1, \ldots, p_k, \textbf{\textit{not}}\ b_1, \ldots, \textbf{\textit{not}}\ b_i\}$ is written as $h_1, \ldots, h_n \leftarrow p_1, \ldots, p_k, \textbf{\textit{not}}\ b_1, \ldots, \textbf{\textit{not}}\ b_i$. For such a rule r, we also define $body^+(r) \coloneqq \{p_1, \ldots, p_k\}$ and $body^-(r) \coloneqq \{b_1, \ldots, b_i\}$ to extract the positive body and the negative body of the rule. Note that the elements in $body^-(r)$ are atoms rather than negated atoms. We say a rule r is *normal* if there is exactly one atom in $hd(r)$. A program that only contains normal rules is a *normal program* (a non-disjunctive program).

In this work, without loss of generality, we do not consider logic programs with variables. That is, we assume every program is *variable-free* or *ground*.

An interpretation of a DLP $\mathcal{P}$ is a true/false assignment to every atom that appears in $\mathcal{P}$. A set of atoms I is an interpretation that assigns its contained atoms to be true and all other atoms to be false. We also consider three-valued interpretations represented by pairs of sets of atoms (described later). When it is necessary to disambiguate between types of interpretations, we call them two-valued interpretations or three-valued interpretations.

Gelfond and Lifschitz [9] define the *answer set semantics* (a.k.a *the stable model semantics*) of normal programs using two-valued interpretations. We also denote this as the $\mathcal{SM}$ semantics with their $\mathcal{SM}$ answer sets.

Definition 1. *Given a normal program $\mathcal{P}$ and an interpretation I, the* (normal) reduct $\mathcal{P}^I$ *of $\mathcal{P}$ w.r.t. I is defined as* $\mathcal{P}^I \coloneqq \{hd(r) \leftarrow body^+(r) \mid r \in \mathcal{P}, body^-(r) \cap I = \emptyset\}$. *A rule r is satisfied by I if*

$$((body^+(r) \subseteq I) \wedge (body^-(r) \cap I = \emptyset)) \implies (hd(r) \subseteq I)$$

is true. An interpretation I is a model of $\mathcal{P}$ if every rule is satisfied by I. A model I of $\mathcal{P}$ is an answer set of $\mathcal{P}$ if there does not exist a model I' of $\mathcal{P}^I$ s.t. $I' \subset I$.

Przymusinski [11] generalizes $\mathcal{SM}$ semantics to three-valued logic by defining *partial answer sets* (a.k.a. partial stable models). Before we can introduce these $\mathcal{PSM}$ answer sets, we must introduce interpretation pairs. While we restrict our attention to three-valued logic, it is convenient to introduce four-valued interpretations.

We briefly summarize the four-valued logic given by Belnap and Nuel [12] to give meaning to four, three, and two-valued interpretations. The logic introduces four truth values: $\boldsymbol{t}$ (true), $\boldsymbol{u}$ (undefined), $\boldsymbol{f}$ (false), and $\boldsymbol{c}$ (contradictory). In this logic, there are two orderings (complete lattices), the *truth-ordering* and the *precision-ordering*.

The truth-ordering places $\boldsymbol{t}$ at the top, $\boldsymbol{f}$ at the bottom, and $\boldsymbol{u}$ and $\boldsymbol{c}$ are incomparable. The precision-ordering places $\boldsymbol{c}$ at the top, $\boldsymbol{u}$ at the bottom, while $\boldsymbol{t}$ and $\boldsymbol{f}$ are incomparable.

We formulate a four-valued interpretation as a pair of sets (T, P) where T and P are interpretations. A *valuation* $(T, P)(a)$ of an atom a against an interpretation (T, P) is $\boldsymbol{t}$ iff $a \in T \cap P$, $\boldsymbol{f}$ iff $a \notin T \cup P$, $\boldsymbol{u}$ iff $a \notin T$, $a \in P$, and $\boldsymbol{c}$ iff $a \in T$, $a \notin P$. The set T contains true atoms, while P contains possibly true atoms (either true or undefined) while the set $T \setminus P$ contains all contradictory atoms. False atoms are not contained in either set T or P. We say such an interpretation is *consistent* (three-valued) if $T \subseteq P$. We call an interpretation (T, T) exact or two-valued.

We define the truth-ordering ($\preceq^2_t$) and precision-ordering ($\preceq^2_p$) for interpretations as follows.

Definition 2. *For two interpretations (T, P) and (T', P')*

$$(T, P) \preceq^2_t (T', P') \iff T \subseteq T' \text{ and } P \subseteq P'$$
$$(T, P) \preceq^2_p (T', P') \iff T \subseteq T' \text{ and } P' \subseteq P$$

Intuitively, these orderings relate interpretations whose individual atom valuations relate according to Belnap's logic. With $(T, P) \preceq^2_p (T', P')$, we have $F \subseteq F'$ where F and F' contain the set false of atoms in (T, P) and (T', P') respectively. We use the following to relate rules and interpretations by describing the set of rules whose bodies are satisfied by an interpretation.

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

Given a consistent (T, P), we say a rule r's body is *satisfied* by (T, P) if $r \in bodysat_{(P,T)}(\{r\})$.

With slight abuse of notation, we use $hd(\mathcal{P})$ (normally $hd(r)$ with a particular rule r) to denote the set of atoms that appear in the heads of rules in a normal program $\mathcal{P}$. That is, $hd(\mathcal{P}) \coloneqq \{hd(r) \mid r \in \mathcal{P}\}$.

We adopt the method of Killen et al. [7] to convert a DLP into a set of normal programs.

Definition 3. *Given a DLP $\mathcal{P}$, the set $normal(\mathcal{P})$ is defined as follows*

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

For example, $normal(\{a, b \leftarrow\}) = \{\{a \leftarrow\}, \{b \leftarrow\}\}$. The programs completion for disjunctive programs shows us that we can deal with disjunctive programs by treating the normal programs in the set above [13]. We follow Killen et al. [7]'s definition of Przymusinski's three-valued semantics of DLPs ($\mathcal{PSM}$ semantics).

Definition 4. *We call a consistent interpretation* (T, P) *a* model *of a DLP* $\mathcal{P}$ *if for some* $\mathcal{P}' \in normal(\mathcal{P})$

$$(hd(bodysat_{(T,P)}(\mathcal{P}')), hd(bodysat_{(P,T)}(\mathcal{P}'))) \preceq_t^2 (T, P)$$

We say a consistent (T', P') is a model of the reduct of $\mathcal{P}$ w.r.t. (T, P) *if for some* $\mathcal{P}' \in normal(\mathcal{P})$

$$(hd(bodysat_{(T',P)}(\mathcal{P}')), hd(bodysat_{(P',T)}(\mathcal{P}'))) \preceq_t^2 (T', P')$$

A model (T, P) *of a DLP* $\mathcal{P}$ *is a* $\mathcal{PSM}$ *answer set of* $\mathcal{P}$ *if there does not exist* $(T', P') \prec_t^2 (T, P)$ *s.t.* (T', P') *is a model of the reduct of* $\mathcal{P}$ *w.r.t.* (T, P).

Przymusinski [14] shows that $\mathcal{PSM}$ answer sets are faithful to Gelfond and Lifschitz's answer set semantics [15].

2.2. Approximation Fixpoint Theory (AFT)

Approximation Fixpoint Theory (AFT) [1] generalizes a large body of research, including $\mathcal{PSM}$ semantics for non-disjunctive programs. AFT algebraically characterizes semantics as fixpoints of certain functions, called approximators.

First, we introduce some lattice theory fundamentals for notation [16]. A *complete lattice* $\mathcal{L}$ is a poset s.t. every set $S \subseteq \mathcal{L}$ a unique least upper bound $\bigvee S$, and unique greatest lower bound $\bigwedge S$. We use $\prec$ to denote the strict variant of a relation $\preceq$ s.t. $(a \prec b) \iff ((a \preceq b) \wedge a \neq b)$.

Given an ordering $\langle \mathcal{L}, \preceq \rangle$ and a function $f : \mathcal{L} \to \mathcal{L}$, we say that f is *monotone* if $a \preceq b$ implies $f(a) \preceq f(b)$. We use $\mathcal{L}^2$ to denote the set of pairs $\{(x, y) \mid x, y \in \mathcal{L}\}$. For a function $f : \mathcal{L}^2 \to \mathcal{L}^2$, we use $f(x, y)_1$ and $f(x, y)_2$ to project the first and second elements, respectively, of the resulting pair. We partially apply and project such functions using the notation $f(\cdot, y)_1$ and $f(x, \cdot)_2$ to construct the new unary functions $\lambda x, f(x, y)_1$ and $\lambda y, f(x, y)_2$ respectively. A *fixpoint* x of a function f is an element such that $f(x) = x$. We use ***fix*** f to denote the set of all fixpoints of f. We use **lfp** f to denote the *least fixpoint* of a function f, that is, **lfp** $f \coloneqq \bigwedge($ ***fix*** $f)$. If a function is monotone over a complete lattice, then **lfp** f is well-defined [17].

AFT [1] leverages the same orderings $\preceq_t^2$ and $\preceq_p^2$ from Definition 2, except $\subseteq$ is replaced with a provided ordering from a complete lattice. We call pairs (z, z) *exact*. We call a function o exact if it maps exact pairs to exact pairs.

Definition 5. *Given a complete lattice* $\langle \mathcal{L}, \preceq \rangle$, *an* approximator $o : \mathcal{L}^2 \to \mathcal{L}^2$ *is an exact and* $\preceq_p^2$*-monotone function.*

An approximator may have fixpoints that do not correspond to the intended semantics. The stable revision operator is formed from an approximator and has fewer fixpoints.

Definition 6. *Given an an approximator* o, *its stable revision operator* $S(o)$ *is*

$$S(o) \coloneqq (\textbf{\textit{lfp}}\ o(\cdot, y)_1, \textbf{\textit{lfp}}\ o(x, \cdot)_2)$$

Denecker et al. [1] show that all fixpoints of a stable operator, called *stable fixpoints* are $\preceq_t^2$-minimal.

2.3. Determining Inference Semantics

We introduce Shen and Eiter's [8] determining inference semantics for DLPs and its associated definitions. First, a DLP induces a family of normal logic programs via a head selection function and a disjunctive reduct.

Definition 7. *Given a DLP* $\mathcal{P}$, *a* head selection function $hs(I, r)$ *is a function that accepts an interpretation* I *and a rule* $r \in \mathcal{P}$ *and returns an element* $h \in hd(r) \cap I$, *unless* $hd(r) \cap I = \emptyset$ *in which case it returns* $\bot$.

Shen and Eiter impose an additional constraint requiring that for any two rules $r, r' \in \mathcal{P}$ s.t. $hd(r) = hd(r')$, then $hs(I, r) = hs(I, r')$. This choice appears to be made to accommodate their more generalized semantics that allow arbitrary formulas in the heads of rules. For simplicity, we do not consider arbitrary formulas and we do not explore this difference, and in the remainder of this work, we assume $hd(r) \neq hd(r')$ when $r \neq r'$ and $|hd(r)| > 1$. This is not so limiting as the semantics of an excluded program can be mimicked by adding fresh atoms to the head of each disallowed rule and additional rules that result in every atom being true if one of these atoms is selected from the head of a rule[1]. Without this syntactic restriction, we cannot establish a strong relationship between Shen and Eiter's determining inference semantics and $\mathcal{PSM}$ answer sets. However, if this restriction is removed, this work holds over a slightly modified version of determining inference semantics, which we soon describe. We leave a deeper investigation into this detail for future work.

From a head-selection function, normal programs are obtained as a disjunctive reduct.

Definition 8. *A disjunctive reduct $\mathcal{P}'$ of a DLP $\mathcal{P}$ w.r.t. an interpretation I and a head selection function hs is a normal program such that $\mathcal{P}' = \{hs(I, r) \leftarrow body(r) \mid r \in \mathcal{P},\ I \text{ satisfies } body(r)\}$.*

Soon, we will define a simpler version of the disjunctive reduct that does not involve an interpretation. We introduce determining inference semantics, which is parameterized by a semantics for normal logic programs. *An answer set semantics $\mathcal{X}$ for normal programs* identifies a subset of models of normal programs as $\mathcal{X}$-answer sets s.t. they are truth-minimal.

Definition 9. *Given an answer set semantics $\mathcal{X}$ for normal programs, a model I of a DLP $\mathcal{P}$ is a DI-$\mathcal{X}$ (determining inference $\mathcal{X}$) answer set of $\mathcal{P}$ if*

1. *I is an $\mathcal{X}$-answer set of some disjunctive reduct of $\mathcal{P}$ w.r.t. I and some head selection function, and*
2. *there is no model of $\mathcal{P}$ $I' \subset I$ that satisfies (1).*

It is convenient for us to make a few simplications and generalizations to Definition 9. We rely on $normal(\mathcal{P})$ (Definition 3) instead of interpretations in disjunctive reducts. Every model of a $\mathcal{P}' \in normal(\mathcal{P})$ is a model of the DLP $\mathcal{P}$ [7], so the model requirement in Definition 9 can be dropped. Additionally, we generalize DI-$\mathcal{X}$ semantics to accommodate $\mathcal{PSM}$ answer sets by parameterizing the underlying interpretation structure and ordering. This is so that later we can apply the semantics with three-valued interpretations. The modified definition is presented below.

A *DI-structure* $\langle \mathcal{X}, \langle \mathcal{L}, \preceq \rangle \rangle$ is an answer set semantics $\mathcal{X}$ and an ordering $\langle \mathcal{L}, \preceq \rangle$. The ordering, an interpretation structure, is used identify elements in $\mathcal{L}$ as models of specific normal programs with some being $\mathcal{X}$-answer sets.

Definition 10. *A* (generalized) DI-$\mathcal{X}$ semantics *is given by a* DI-structure $\langle \mathcal{X}, \langle \mathcal{L}, \preceq \rangle \rangle$. *We say $I \in \mathcal{L}$ is a* (generalized) DI-$\mathcal{X}$ answer set *of a DLP $\mathcal{P}$ if*

1. *I is an $\mathcal{X}$-answer set of some $\mathcal{P}' \in normal(\mathcal{P})$, and*
2. *there is no $I' \in \mathcal{L}$ s.t. $I' \prec I$ and I' satisfies (1).*

Proposition 1. *Using the DI-structure $\langle \mathcal{X}, \langle \mathcal{I}, \subseteq \rangle \rangle$, Definitions 9 and 10 are equivalent for programs $\mathcal{P}$ s.t. for every rule $r \in \mathcal{P}$ with $|hd(r)| > 1$, there does not exist $r' \in \mathcal{P}$ s.t. $r \neq r'$ and $hd(r) = hd(r')$.*

Proof. If I is an $\mathcal{X}$ answer set of a program $\mathcal{P}' \in normal(\mathcal{P})$, then we can construct a head selection function that chooses the true atoms from the heads of rules in $\mathcal{P}'$. Removing the rules whose bodies are not satisfied by I does not affect the status of I as an answer set.

Clearly we have $\mathcal{P}' \in normal(\mathcal{P})$ s.t. $\mathcal{P}'$ has an answer set $I' \subset I$ iff there exists a disjunctive reduct that corresponds to $\mathcal{P}'$. Note that head selection function for $\mathcal{P}'$ uses I', not I, thus the removal of unsatisfied rules has no bearing on I''s status as an answer set of $\mathcal{P}'$. □

From here on, we adopt Definition 10 for DI-$\mathcal{X}$ semantics.

[1]While one might wish to use a constraint instead of making every atom true, under determining inference semantics, adding constraints can introduce new answer sets [8].

Definition 11. *The semantics DI-$\mathcal{SM}$ are given by applying DI-$\mathcal{X}$ to Gelfond and Lifschitz's answer set semantics (Definition 1) using the the DI-structure $\langle \mathcal{SM}, \langle \mathcal{I}, \subseteq \rangle \rangle$.*

We return to Shen and Eiter's example [8] (Example 1)

Example 1. *Define $\mathcal{P}$ to be the following DLP*

$$a \leftarrow \qquad\qquad a, b \leftarrow \qquad\qquad b \leftarrow \textbf{\textit{not}}\ b$$

The program has no answer set under Gelfond and Lifschitz's answer set semantics [15]. We have

$$normal(\mathcal{P}) \coloneqq \{\mathcal{P}_a, \mathcal{P}_b\}, \text{ where}$$
$$\mathcal{P}' \coloneqq \{a \leftarrow, b \leftarrow \textbf{\textit{not}}\ b\} \qquad \mathcal{P}_a \coloneqq \mathcal{P}' \cup \{a \leftarrow\} \qquad \mathcal{P}_b \coloneqq \mathcal{P}' \cup \{b \leftarrow\}$$

The program $\mathcal{P}_a$ has no $\mathcal{SM}$-answer sets, whereas $\mathcal{P}_b$ has the $\mathcal{SM}$-answer set $\{a, b\}$. Thus, under DI-$\mathcal{SM}$, this program has one DI-$\mathcal{SM}$ answer set $\{a, b\}$.

3. Capturing DI-Semantics

We are now ready to capture determining inference semantics with approximator sets. First, we capture the semantics defined by Shen and Eiter, that is, the two-valued determining inference semantics. We show that given any two-valued answer set semantics $\mathcal{X}$, we can capture the two-valued DI-$\mathcal{X}$ semantics using approximator sets.

First, we introduce approximator sets [7] to use AFT in a nondeterministic setting.

Definition 12. *An* approximator set *is a finite[2] set of approximators.*

To capture the DI semantics in the two-valued case, we can simply isolate the truth-minimal stable fixpoints of approximators in an approximator set. Recall that we use $\textbf{\textit{fix}}\ S(h)$ to denote stable fixpoints of the approximator h.

Definition 13. *The exact DI-$\mathcal{X}$ stable fixpoints of an approximator set H are contained in the set*

$$\textbf{\textit{min}}_{\preceq_t^2}(\bigcup\{(x, x) \in \textbf{\textit{fix}}(S(h)) \mid h \in H\})$$

It is immediate that every exact DI-$\mathcal{X}$ stable fixpoints is $\preceq_t^2$-minimal.

We now show how the above use of approximator sets can capture the DI-$\mathcal{X}$ semantics for an answer set semantics $\mathcal{X}$. Given a semantics $\mathcal{X}$ for normal programs and a DLP $\mathcal{P}$, an *exact corresponding approximator set* is a set H of cardinality $|normal(\mathcal{P})|$ s.t. for each $\mathcal{P}' \in normal(\mathcal{P})$, there is an approximator in H whose exact stable fixpoints correspond to the $\mathcal{X}$-answer sets of $\mathcal{P}'$.

Theorem 1. *Given an answer set semantics $\mathcal{X}$, a DLP $\mathcal{P}$, and an exact corresponding approximator set H, we have that (T, T) is an exact DI-$\mathcal{X}$ stable fixpoint of H (Definition 13) iff it is an DI-$\mathcal{X}$ answer set of $\mathcal{P}$ (Definition 10) under a two-valued DI-structure $\langle \mathcal{X}, \langle \mathcal{I}, \preceq \rangle \rangle$.*

Proof. ($\Rightarrow$) By definition, there is no approximator $h_{\mathcal{P}'} \in H$ s.t. $(T', T') \in \textbf{\textit{fix}}\ S(h_{\mathcal{P}'})$ and $T' \subset T$. Thus, there is no $\mathcal{P}' \in normal(\mathcal{P})$ with such an answer set T'. ($\Leftarrow$) Similarly, with no $\mathcal{P}' \in normal(\mathcal{P})$ with an answer set $T' \subset T$, there is no $h \in H$ with the exact stable fixpoint (T', T'). □

Intuitively, if a two-valued non-disjunctive semantics $\mathcal{X}$ has an approximator, then we can immediately derive a two-valued determining inference semantics to handle the disjunctive case.

For example, to capture determining inference semantics for Gelfond and Lifschitz's answer set semantics using stable revision, we adopt the approximator set defined for a set of normal programs by Killen et al. [7].

[2] We restrict ourselves to finite sets here as infinite sets introduce additional complications that are not relevant to disjunctive logic programs [7].

Definition 14. *Given a DLP $\mathcal{P}$,*

$$\Gamma_{\mathcal{P}}(T,P) \coloneqq \{h \mid hd(r) = \{h\}, r \in \mathcal{P}, body^+(r) \subseteq T, body^-(r) \cap P = \emptyset\}$$
$$h_{\mathcal{P}}(T,P) \coloneqq (\Gamma_{\mathcal{P}}(T,P), \Gamma_{\mathcal{P}}(P,T)) \qquad H(\mathcal{P}) \coloneqq \{h_{\mathcal{P}'} \mid \mathcal{P}' \in normal(\mathcal{P})\}$$

$H(\mathcal{P})$ is an approximator defined with the approximators that capture the $\mathcal{SM}$ semantics of the programs in $normal(\mathcal{P})$. The following comes immediately from Theorem 1 and because $H(\mathcal{P})$ is a corresponding approximator set for $\mathcal{SM}$ semantics.

Theorem 2. *Let I be an interpretation of a DLP $\mathcal{P}$. It is a DI-$\mathcal{SM}$ of $\mathcal{P}$ if and only if (I,I) is an exact DI-$\mathcal{SM}$ stable fixpoint of $H(\mathcal{P})$.*

We have established that we can use AFT to capture the Shen and Eiter's [8] DI-$\mathcal{SM}$ semantics.

AFT distinguishes two types of fixpoints that always exist, exhibit special properties, and that can be used to characterize new or existing semantics.

Definition 15. *Given an approximator o, $(\textbf{\textit{lfp}}_{\preceq_p^2} o)$ is the Kripke-Kleene fixpoint of o and $(\textbf{\textit{lfp}}_{\preceq_p^2} S(o))$ is the well-founded fixpoint of o.*

Several properties of the above can easily be shown due to the algebraic nature of fixpoints. Namely, that the Kripke-Kleene fixpoint is less than any other fixpoint, the well-founded fixpoint is less than any other stable fixpoint, and the Kripke-Kleene fixpoint is less than the well-founded fixpoint.

We now lift Kripke-Kleene and well-founded fixpoints to approximator sets.

Definition 16. *For an approximator set H, a*

- **local** *Kripke-Kleene fixpoint of H is a Kripke-Kleene fixpoint of some approximator from H, and a*
- **local** *well-founded fixpoint of H is a well-founded fixpoint of some approximator from H.*

Definition 17. *For an approximator set H, its **global** Kripke-Kleene (resp. **global** well-founded) fixpoints are $\preceq_t^2$-minimal local Kripke Kleene (resp. well-founded) fixpoints of H.*

The following comes immediately due to the finitude of an approximator set and the complete lattice structure of $\preceq_t^2$ [12].

Lemma 1. *Every approximator set H has at least one global Kripke-Kleene fixpoint and at least one global well-founded fixpoint.*

In a similar vein to Heyninck et al. [4], we define disjunctive states.

Definition 18. *The* Kripke-Kleene state *(resp. well-founded state) of an approximator set is a nonempty set containing all of its global Kripke-Kleene fixpoints (resp. global well-founded fixpoints).*

The existence of these states follows immediately from Lemma 1.

Corollary 1. *For an approximator set H, its Kripke-Kleene state (resp. well-founded state) is unique and nonempty.*

We briefly instantiate the discussed definitions in the following example.

Example 2. *Let $\mathcal{L} = \{\bot, \top, a, b\}$ be the complete lattice where $\bot \preceq a \preceq \top$ and $\bot \preceq b \preceq \top$. Define H to be a set of approximators indexed by $\mathcal{L} \setminus \{\bot\}$ such that for $\alpha \in (\mathcal{L} \setminus \{\bot\})$ and $h_\alpha \in H$*

$$h_\alpha(x,y)_1 \coloneqq \bigvee\{\alpha, x\}$$
$$h_\alpha(x,y)_2 \coloneqq h_\alpha(y,x)_1$$

Clearly, each approximator is $\preceq_p^2$-monotone and symmetric. For each approximator $h_\alpha \in H$, we have that $(\alpha, \top)$ is the Kripke-Kleene fixpoint and (α,α) is the well-founded fixpoint. Clearly $(\alpha,\top) \preceq_p^2 (\alpha,\alpha)$. These are local Kripke-Kleene (resp. well-founded) fixpoints of H. The Kripke-Kleene state of H is $\{(a,\top),(b,\top)\}$ and the well-founded state of H is $\{(a,a),(b,b)\}$. If we were to add $h_\bot$ to H, then its Kripke-Kleene state would be $\{(\bot,\top)\}$ and its well-founded state would be $\{(\bot,\bot)\}$.

3.1. Three-Valued Semantics

For exact DI-$\mathcal{X}$ stable fixpoints (Definition 13) we only minimize against exact stable fixpoints. However, our approximators for DI-$\mathcal{SM}$ semantics have stable fixpoints that are not exact. These partial stable fixpoints are meaningful because they approximate exact stable fixpoints (like well-founded models). Our goal is to combine determining inference semantics with the partial stable model semantics to obtain a new semantics, DI-$\mathcal{PSM}$. If we minimize $\mathcal{PSM}$ answer sets using the $\preceq_t^2$ ordering, we do not preserve DI-$\mathcal{PSM}$ answer sets. We briefly demonstrate how using $\preceq_t^2$, that is, the DI-structure $\langle \mathcal{PSM}, \langle \mathcal{L}^2, \preceq_t^2 \rangle \rangle$, is problematic.

Example 3. *Let $\mathcal{P}$ be the program from Example 1.*

$$a \leftarrow \qquad\qquad a, b \leftarrow \qquad\qquad b \leftarrow \textbf{\textit{not}}\ b$$

There is one $\mathcal{PSM}$ answer set $(\{a\}, \{a, b\})$ under Przymusinski's semantics [14] (Definition 4). The interpretation $(\{a\}, \{a, b\})$ is a $\mathcal{PSM}$ answer set of the disjunctive reduct which selects a from the head of the rule $a, b \leftarrow$. In Example 1, we established that $(\{a, b\}, \{a, b\})$ is a DI-$\mathcal{SM}$ answer set of $\mathcal{P}$. However, we have $(\{a\}, \{a, b\}) \preceq_t^2 (\{a, b\}, \{a, b\})$. Thus, if we were to use $\preceq_t^2$ to formulate a three-valued variation of DI-$\mathcal{X}$ semantics, we would not preserve $(\{a, b\}, \{a, b\})$ as a DI-$\mathcal{SM}$ answer set as it is not minimal w.r.t. the ordering supplied in the DI-structure. Note that, as a non-exact pair, $(\{a\}, \{a, b\})$ does not participate in truth minimality checking for exact pairs in Definition 13.

We wish to preserve both the DI-$\mathcal{SM}$ answer sets and Przymusinski's $\mathcal{PSM}$ answer sets. One possible approach would be to remove every instance from the relation $\preceq_t^2$ that compares exact pairs with non-exact pairs. This technique is employed by Knorr et al. [18] when lifting the semantics of Hybrid MKNF from the two-valued case to the three-valued case.[3] Clearly, using this modified $\preceq_t^2$ ordering would tightly preserve all DI-$\mathcal{SM}$ answer sets because the truth minimization of exact interpretations reduces to $\subseteq$.

However, the approach of modifying $\preceq_t^2$ in this way yields an unnatural semantics. One simply has to add the rule $y \leftarrow \textbf{\textit{not}}\ y$ to a program (where y is a fresh atom), ignore y in the resulting interpretations, and the resulting semantics are equivalent having used $\preceq_t^2$ in the DI-structure. Later on, we demonstrate this concretely in Example 4. To address this peculiarity, we introduce a new ordering for truth-minimization.

Definition 19. $(x, y) \preceq_{qt}^2 (x', y') \iff x \preceq x'$ *and* $y \preceq (y' \setminus (x' \setminus x))$

Note that because the underlying ordering is $\subseteq$, the $\setminus$ operation is defined as the set difference. Intuitively, $\preceq_{qt}^2$ exhibits a quasi-truth ordering in that it removes certain pairings from the relation that compare exact pairs with non-exact pairs. Additionally, $\preceq_{qt}^2$ compares truth values of individual atoms in a way that resembles $\preceq_t^2$, with the difference that the values $\boldsymbol{t}$ and $\boldsymbol{u}$ are incomparable. That is, for an answer set to be favoured over another, it must make some true or undefined atoms false. In contrast to $\preceq_t^2$, in which to shrink an interpretation, it is sufficient to assign some true atoms the value of undefined.

First, we show that this new ordering is weaker than $\preceq_t^2$.

Lemma 2. *For two consistent interpretations (T, P) and (T', P'), we have that if $(T, P) \preceq_{qt}^2 (T', P')$, then $(T, P) \preceq_t^2 (T', P')$.*

Proof. Clearly, $(y \setminus (x \setminus x')) \preceq y$, thus $y' \preceq y$. □

Now, we demonstrate how this ordering minimizes truth.

Lemma 3. *Given two consistent interpretations (T, P) and (T', P') and an atom a s.t. $(T, P)(a) \neq \boldsymbol{t}$ and $(T', P')(a) = \boldsymbol{t}$, if $(T, P) \prec_{qt}^2 (T', P')$ then $(T, P)(a) = \boldsymbol{f}$.*

[3] In Knorr et al.'s Definition 9 [18], when an interpretation pair (M, N) is compared against a pair (M', N'), they require $M' = N'$ if $M = N$.

Proof. By contrapositive, suppose $(T, P)(a) = \boldsymbol{u}$. We have $a \notin T$ and $a \in P$. Because $(T', P')(a) = \boldsymbol{t}$, we have $a \in T'$, thus $a \in (T' \setminus T)$. It follows that $a \notin P' \setminus (T' \setminus T)$, and then we can conclude $\neg(P \preceq (P' \setminus (T' \setminus T)))$. □

Finally, we show that when minimizing truth, partial consistent interpretations will not be favoured over exact interpretations.

Lemma 4. *With $(T, P) \preceq^2_{qt} (T', T')$ s.t. $T \subseteq P$, we have $T = P$.*

Proof. Suppose $T \subset P$ and let $a \in (P \setminus T)$. We have $(T, P)(a) = \boldsymbol{u}$. From Lemma 2, it follows that $P \subseteq T'$, thus $(T', T')(a) = \boldsymbol{t}$. We can apply Lemma 3 to conclude $(T, P)(a) = \boldsymbol{f}$, a contradiction. □

We now lift $\mathcal{PSM}$ answer sets [11] from normal logic programs to DLPs using DI-$\mathcal{X}$ semantics as defined in Definition 10.

Definition 20. *The DI-$\mathcal{PSM}$ semantics are given by using the determining inference structure $\langle \mathcal{PSM}, \langle \mathcal{L}^2, \preceq^2_{qt} \rangle \rangle$ with the DI-$\mathcal{X}$ semantics (Definition 10).*

The resulting semantics DI-$\mathcal{PSM}$ is equivalent to $\mathcal{PSM}$ for normal programs [8], but differs for disjunctive programs. The following example demonstrates the new semantics.

Example 4. *Let $\mathcal{P}$ be the program given in Example 1. DI-$\mathcal{PSM}$ is faithful to both DI-$\mathcal{SM}$ (Definition 11) and $\mathcal{PSM}$ semantics (Definition 4). That is, both $(\{a, b\}, \{a, b\})$ and $(\{a\}, \{a, b\})$ are DI-$\mathcal{PSM}$ answer sets. No other interpretation is a DI-$\mathcal{PSM}$ answer set. If we add the rule $y \leftarrow \textbf{not}\ y$, then $(\{a, b\}, \{a, b, y\})$ and $(\{a\}, \{a, b, y\})$ are the DI-$\mathcal{PSM}$ answer sets. The interpretation $(\{a, b\}, \{a, b, y\})$ is neither a DI-$\mathcal{SM}$ nor a $\mathcal{PSM}$ answer set.*

The example above shows that some non-exact DI-$\mathcal{PSM}$s are not $\mathcal{PSM}$ answer sets. This is by choice as combining $\mathcal{PSM}$ answer sets and DI-$\mathcal{SM}$ under a single semantics will result in a semantics where not every DI-answer set is $\preceq^2_t$ minimal. The following lemma provides some further insight into the relation between DI-$\mathcal{PSM}$ answer sets and $\preceq^2_t$.

Lemma 5. *If (T, P) is a $\mathcal{PSM}$ answer set of some $\mathcal{P}' \in normal(\mathcal{P})$ for a DLP $\mathcal{P}$, then $\mathcal{P}$ has a DI-$\mathcal{PSM}$ (T', P') s.t. $(T', P') \preceq^2_t (T, P)$.*

This shows that, if desired, one can leverage $\langle \mathcal{PSM}, \langle \mathcal{L}^2, \preceq^2_t \rangle \rangle$ to obtain a DI-$\mathcal{X}$ semantics that is $\preceq^2_t$ minimal at the cost of sacrificing some DI-$\mathcal{SM}$ answer sets (Example 4).

We show that in general, DI-$\mathcal{PSM}$ is faithful to both the original DI-$\mathcal{SM}$ semantics [8] and $\mathcal{PSM}$ answer sets [14].

Proposition 2. *If (T, P) is a $\mathcal{PSM}$ answer set of $\mathcal{P}$, then it is a DI-$\mathcal{PSM}$ answer set of $\mathcal{P}$.*

Proof. By contrapositive, assume (T, P) is not a DI-$\mathcal{PSM}$ of $\mathcal{P}$. We have $(T', P') \prec^2_{qt} (T, P)$ s.t. (T', P') is a $\mathcal{PSM}$ answer set of a disjunctive reduct $\mathcal{P}'$ of $\mathcal{P}$. It follows that $(T', P') \preceq^2_t (T, P)$ (Lemma 2). With $P' \subseteq P$ and as a model of $\mathcal{P}'$, we have that (T', P') satisfies the reduct of $\mathcal{P}$ w.r.t. (T, P), thus (T, P) is not a $\mathcal{PSM}$ answer set of $\mathcal{P}$. □

In general, the converse does not hold, that is, there are some DI-$\mathcal{PSM}$ stable fixpoints that are not $\mathcal{PSM}$ answer sets (see Example 4).

Proposition 3. *If (T, T) is a DI-$\mathcal{SM}$ answer set of $\mathcal{P}$, then it is a DI-$\mathcal{PSM}$ answer set of $\mathcal{P}$.*

Proof. For the sake of contradiction, suppose (T, T) is not a DI-$\mathcal{PSM}$. There exists $(T', P') \prec^2_{qt} (T, T)$ s.t. (T', P') is $\mathcal{PSM}$ answer sets of a disjunctive reduct of the DLP $\mathcal{P}$. By Lemma 4, $T' = P'$, which contradicts the assumption that (T, T) is a DI-$\mathcal{SM}$. □

Naturally, it follows that our semantics capture the $\mathcal{SM}$ answer sets.

Unlike $\mathcal{PSM}$ answer sets, a DI-$\mathcal{PSM}$s is guaranteed to exist.

Proposition 4. *Every DLP has a DI-$\mathcal{PSM}$ answer set.*

Proof. Let $\mathcal{P}'$ be some disjunctive reduct of the DLP $\mathcal{P}$. $\mathcal{P}'$ has a $\mathcal{PSM}$ answer set, then by Lemma 5, $\mathcal{P}$ has a DI-$\mathcal{PSM}$. □

Using $\boldsymbol{fix}f$ to denote the set of fixpoints of a function f, we now define a method of filtering stable fixpoints that is similar to Definition 13 but which minimizes across all stable fixpoints instead of just exact stable fixpoints.

Definition 21. *Given an approximator set H and a DI-structure $\langle \mathcal{X}, \langle \mathcal{L}^2, \preceq \rangle \rangle$ the* DI-$\mathcal{X}$ stable fixpoints *of H are contained in the set $\boldsymbol{min}_{\preceq}(\bigcup\{\boldsymbol{fix}(S(h)) \mid h \in H\})$.*

We instantiate the above for $\mathcal{PSM}$ answer sets.

Definition 22. *Given an approximator set H, the* DI-$\mathcal{PSM}$ stable fixpoints *of H are given as the DI-$\mathcal{X}$ stable fixpoints with $\langle \mathcal{PSM}, \langle \mathcal{L}^2, \preceq^2_{qt} \rangle \rangle$.*

Echoing Theorem 1, but this time for the three-valued case, we show that if a normal answer set semantics $\mathcal{X}$ has an approximator, the DI-$\mathcal{X}$ semantics can be characterized using an approximator set. A *corresponding approximator set* for $\mathcal{X}$ and a DLP $\mathcal{P}$ is a set H of cardinality $|normal(\mathcal{P})|$ s.t. for each $\mathcal{P}' \in normal(\mathcal{P})$, there is an approximator in H whose stable fixpoints correspond to the $\mathcal{X}$-answer sets of $\mathcal{P}'$.[4]

Theorem 3. *Given an answer set semantics $\mathcal{X}$, a DLP $\mathcal{P}$, and an exact corresponding approximator set H, we have that (T, T) is a DI-$\mathcal{X}$ stable fixpoint of H (Definition 21) iff it is a DI-$\mathcal{X}$ answer set of $\mathcal{P}$ (Definition 10).*

Proof. Follows proof of Theorem 1. □

Given that $H(\mathcal{P})$ is a corresponding approximator set for $\mathcal{PSM}$, we get the following result from Theorem 3.

Theorem 4. *Let (T, P) be an interpretation of a DLP $\mathcal{P}$. It is a DI-$\mathcal{PSM}$ answer set of $\mathcal{P}$ if and only if (T, P) is a DI-$\mathcal{PSM}$ stable fixpoint of $H(\mathcal{P})$.*

Example 5. *Define $\mathcal{P}$ as the program from Example 1 with the added rule $u \leftarrow \boldsymbol{not}\ u$. This program has no DI-$\mathcal{SM}$ answer sets and two DI-$\mathcal{PSM}$ answer sets, namely, the $\mathcal{PSM}$ answer sets $(\{a\}, \{a, b, u\})$ and $(\{a, b\}, \{a, b, u\})$.*

Let $\mathcal{P}_a$ and $\mathcal{P}_b$ be the programs from $normal(\mathcal{P})$ that select a and b respectively from the head of $a, b \leftarrow$ and let $h_{\mathcal{P}_a}$ and $h_{\mathcal{P}_b}$ be their approximators from $H(\mathcal{P})$.

We have that $(\{a, b\}, \{a, b, u\})$ is a stable fixpoint of $S(h_{\mathcal{P}_b})$ and not a fixpoint of $S(h_{\mathcal{P}_a})$ with $S(h_{\mathcal{P}_a})((\{a, b\}, \{a, b, u\})) = (\{a\}, \{a, u\})$.

Killen et al. [19] define stable fixpoints of approximator sets and show that it can capture $\mathcal{PSM}$ answer sets of DLPs. We introduce this definition to compare.

Definition 23. *Given an approximator set H, a disjunctive stable fixpoint of H is a pair $(x, y) \in \boldsymbol{fix}(S(h))$ for some $h \in H$ s.t. $(x, y) \in \boldsymbol{min}_{\preceq^2_t}\{S(h)(x, y) \mid h \in H\}$.*

Comparing DI-stable fixpoint to disjunctive stable fixpoints, we can generalize Proposition 2 to the level of algebra.

Proposition 5. *Given a DI-structure $\langle \mathcal{X}, \langle \mathcal{L}^2\ \preceq^2_{qt} \rangle \rangle$, A disjunctive stable fixpoint (Definition 23) of an approximator set H is a DI-$\mathcal{X}$ stable fixpoint of H (Definition 21).*

That DI-$\mathcal{PSM}$ answer sets subsume $\mathcal{PSM}$ answer sets (Proposition 2) follows from the above.

[4] Unlike Theorem 1, the corresponding approximator set is not limited to exact pairs

Corollary 2. *A DI-$\mathcal{PSM}$ stable fixpoint is a disjunctive stable fixpoint (Definition 23).*

We can also show that the DI-$\mathcal{PSM}$ stable fixpoints capture the exact DI-$\mathcal{SM}$ stable fixpoints.

Proposition 6. *An exact DI-$\mathcal{SM}$ stable fixpoint (Definition 13) is a DI-$\mathcal{PSM}$ stable fixpoint (Definition 20).*

We can discuss the local/global Kripke-Kleene/well-founded fixpoints and states. Now that we have captured DI-$\mathcal{PSM}$ semantics via a set of approximators, we automatically obtain the distinguished fixpoints and states (Definitions 16, 17, and 18). We briefly instantiate these for a DI-$\mathcal{PSM}$ semantics.

Example 6. *Define $\mathcal{P}$ to be the following program*

$$a, b \leftarrow \qquad\qquad a, b, c \leftarrow \qquad\qquad c \leftarrow \textbf{\textit{not}}\ c$$

*There are six programs in $normal(\mathcal{P})$. We denote the program from $normal(\mathcal{P})$ that selects the atom α from $hd(a, b \leftarrow)$ and β from $hd(a, b, c \leftarrow)$ as $\mathcal{P}_{\alpha,\beta}$. Thus, $normal(\mathcal{P}) = \{\mathcal{P}_{a,a}, \mathcal{P}_{a,b}, \mathcal{P}_{a,c}, \mathcal{P}_{b,a}, \mathcal{P}_{b,b}, \mathcal{P}_{b,c}\}$. The Kripke-Kleene fixpoints of each program $\mathcal{P}_{\alpha,\beta}$, i.e., the local Kripke-Kleene fixpoints of $H(\mathcal{P})$, take the form $(\{\alpha, \beta\}, \{\alpha, \beta, c\})$. For example, **lfp** $h_{\mathcal{P}_{a,c}} = (\{a, c\}, \{a, c\})$. In this case, **lfp** $h_{\mathcal{P}_{a,c}}$ is not a global Kripke-Kleene fixpoint because **lfp** $h_{\mathcal{P}_{a,a}} = (\{a\}, \{a, c\}) \preceq_t^2 (\{a, c\}, \{a, c\})$. The interpretation $(\{a\}, \{a, c\})$ is a global Kripke-Kleene fixpoint and thus it is a part of the Kripke-Kleene state.*

4. Discussion

We have demonstrated the flexibility of Killen et al.'s [7] approach to disjunctive AFT by capturing Shen and Eiter's [8] determining inference semantics. Because AFT characterizes three-valued semantics, we have lifted determining inference semantics to the three-valued case. We have shown that for a three-valued semantics to be faithful to both the two-valued determining inference semantics and Przymusinski's three-valued semantics for DLPs, we must sacrifice $\preceq_t^2$-minimality. It is not difficult to see that the semantics could be tweaked by selecting a different ordering.

The connections between DI-$\mathcal{PSM}$ and $\mathcal{PSM}$ rely on the syntactic restriction that $hd(r) = hd(r')$ implies $r \neq r'$ for disjunctive rules. Shen and Eiter use a peculiar method of generating normal programs from a disjunctive logic program. Interestingly, their method of choosing normal programs has little impact on the semantics of disjunctive $\mathcal{PSM}$ answer sets. In general, the method of choosing normal programs greatly impacts the semantics in DI-$\mathcal{X}$ semantics. In future work, we wish to further generalize determining inference semantics such that this head selection function is parameterized and study this wider family of semantics through the lens of AFT.

Heyninck et al. lift AFT to a nondeterministic setting [4]. Due to its compatibility with approximator sets [7], it is likely this theory could also be used to define DI-$\mathcal{X}$ stable fixpoints. However, it is unclear how to define stable revision.

Acknowledgements

Spencer Killen was partially supported by Alberta Innovates and Alberta Advanced Education.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] M. Denecker, V. Marek, M. Truszczyński, Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning, in: Logic-Based Artificial Intelligence, Springer, 2000, pp. 127–144. doi:10.1007/978-1-4615-1567-8_6.

[2] C. Antić, T. Eiter, M. Fink, Hex semantics via approximation fixpoint theory, in: Proceedings of the 12th International Conference on Logic Programming and Nonmonotonic Reasoning - Volume 8148, LPNMR 2013, Springer-Verlag, Berlin, Heidelberg, 2013, p. 102–115. URL: https://doi.org/10.1007/978-3-642-40564-8_11. doi:10.1007/978-3-642-40564-8_11.

[3] N. Pelov, M. Truszczynski, Semantics of disjunctive programs with monotone aggregates-an operator-based approach, in: Proceedings of the 10th International Workshop on Non-Monotonic Reasoning, 2004, pp. 327–334.

[4] J. Heyninck, O. Arieli, B. Bogaerts, Non-deterministic approximation fixpoint theory and its application in disjunctive logic programming, Artificial Intelligence 331 (2024) 104110.

[5] J. Heyninck, B. Bogaerts, Non-deterministic approximation operators: Ultimate operators, semi-equilibrium semantics, and aggregates, Theory Pract. Log. Program. 23 (2023) 632–647. URL: https://doi.org/10.1017/s1471068423000236. doi:10.1017/S1471068423000236.

[6] J. Heyninck, Operator-based semantics for choice programs: Is choosing losing?, in: P. Marquis, M. Ortiz, M. Pagnucco (Eds.), Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning, KR 2024, Hanoi, Vietnam. November 2-8, 2024, 2024. URL: https://doi.org/10.24963/kr.2024/42. doi:10.24963/KR.2024/42.

[7] S. Killen, J.-H. You, J. Heyninck, An alternative theory of stable revision for nondeterministic approximation fixpoint theory and the relationships, in: Thirty-Ninth AAAI Conference on Artificial Intelligence, AAAI 2025, AAAI Press, 2025.

[8] Y. Shen, T. Eiter, Determining inference semantics for disjunctive logic programs, Artif. Intell. 277 (2019). URL: https://doi.org/10.1016/j.artint.2019.103165. doi:10.1016/J.ARTINT.2019.103165.

[9] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: R. Kowalski, Bowen, Kenneth (Eds.), Proceedings of International Logic Programming Conference and Symposium, MIT Press, 1988, pp. 1070–1080. URL: http://www.cs.utexas.edu/users/ai-lab?gel88.

[10] T. Eiter, G. Gottlob, On the computational cost of disjunctive logic programming: Propositional case, Ann. Math. Artif. Intell. 15 (1995) 289–323. doi:10.1007/BF01536399.

[11] T. C. Przymusinski, The well-founded semantics coincides with the three-valued stable semantics, Fundam. Inform. 13 (1990) 445–463.

[12] N. D. Belnap, A Useful Four-Valued Logic, Springer Netherlands, Dordrecht, 1977, pp. 5–37. doi:10.1007/978-94-010-1161-7_2.

[13] J. Lee, V. Lifschitz, Loop formulas for disjunctive logic programs, in: C. Palamidessi (Ed.), Logic Programming, 19th International Conference, ICLP 2003, Mumbai, India, December 9-13, 2003, Proceedings, volume 2916 of *Lecture Notes in Computer Science*, Springer, 2003, pp. 451–465. doi:10.1007/978-3-540-24599-5_31.

[14] T. C. Przymusinski, Stable semantics for disjunctive programs, New Gener. Comput. 9 (1991) 401–424. doi:10.1007/BF03037171.

[15] M. Gelfond, V. Lifschitz, Classical negation in logic programs and disjunctive databases, New Gener. Comput. 9 (1991) 365–386. doi:10.1007/BF03037169.

[16] S. Roman, Lattices and Ordered Sets, Springer New York, 2008. doi:10.1007/978-0-387-78901-9.

[17] A. Tarski, A lattice-theoretical fixpoint theorem and its applications., Pacific Journal of Mathematics 5 (1955) 285 – 309. doi:10.2140/pjm.1955.5.285.

[18] M. Knorr, J. J. Alferes, P. Hitzler, Local closed world reasoning with description logics under the well-founded semantics, Artif. Intell. 175 (2011) 1528–1554. doi:10.1016/j.artint.2011.01.007.

[19] S. Killen, J. You, A fixpoint characterization of three-valued disjunctive hybrid MKNF knowledge bases, in: Y. Lierler, J. F. Morales, C. Dodaro, V. Dahl, M. Gebser, T. Tekle (Eds.), Proceedings 38th International Conference on Logic Programming, ICLP 2022 Technical Communications / Doctoral

Consortium, Haifa, Israel, 31st July 2022 - 6th August 2022, volume 364 of *EPTCS*, 2022, pp. 51–64. doi:10.4204/EPTCS.364.6.