Semantics for Logic Programs with Choice Constructs on the Basis of Approximation Fixpoint Theory (Preliminary Report)

Jesse Heyninck

Open Universiteit, the Netherlands
University of Cape Town, South Africa

Abstract
Choice constructs are an important addition to the language of logic programming that greatly increase its modeling capabilities. Their semantics are non-deterministic, in the sense that their might be several interpretations that satisfy a choice construct. In this paper, the semantics of logic programs with choice operators are studied using the recently proposed *non-deterministic approximation fixpoint theory*. We show that this allows to represent the semantics of Liu, Pontenelli, Son and Trusczczyński and generalize these semantics to the three-valued case. Furthermore, the framework allows us to give a principled account of the difference and similarities between stable model semantics of choice programs and disjunctive logic programs.

Keywords
Approximation fixpoint theory, choice constructs, logic programming

1. Introduction

Logic programming is one of the most popular declarative formalisms, as it offers an expressive, rule-based modelling language and efficient solvers for knowledge representation. An important part of this expressiveness comes from *choice construct*, that allow to state e.g. set constraints in the head of rules. For example, the rule $1 \leq \{p, q, r\} \leq 2 \leftarrow s$ expresses that if s is true, between 1 and 2 atoms among p, q and r can be true. Choice constructs are non-deterministic, in the sense that there is more than one way to satisfy their head. For example, $1 \leq \{p, q, r\} \leq 2$ can be satisfied by $\{p\}$, $\{p, q\}$, $\{r\}$, ... Formulating semantics for such non-deterministic rules has proven a challenging task [1, 2, 3, 4], and, to the best of our knowledge, attention has been restricted to two-valued semantics. Furthermore, the relation with a related non-deterministic construct, namely disjunction, is not clear.

Approximation fixpoint theory (AFT) [5] is a purely algebraic theory which was shown to unify the semantics of, among others, logic programming default logic and autoepistemic logic. The central objects of study of AFT are *(approximating) operators* and their *fixpoints*. For logic programming for instance, it was shown that Fitting's [6] three-valued immediate consequence operator is an approximating operator of Van Emden and Kowalski's [7] two-valued immediate consequence operator and that all major semantics of (normal) logic programming can be derived directly from this approximating operator. Recently, AFT was generalized to also capture *non-deterministic operators* [8] which allow for different options or choices in their output. It was shown that many major semantics of disjunctive logic programming (specifically the weakly supported, (partial) stable, and well-founded semantics [9]) are captured by non-deterministic AFT.

In this paper, we commence an operator-based study of the semantics of logic programs with choice constructs in the head using the framework of non-deterministic AFT. This has several advantages: (1) it brings the semantics of these programs in a principle-based framework for the definition of semantics; (2) it gives immediately rise to a wide variety of semantics, such as the three-valued fixpoint and stable semantics, and the state semantics; (3) it allows to compare these semantics with semantics for other, related formalisms, notably, disjunctive logic programs; and (4) allows to use concepts investigated for AFT such as stratification [10].

This study of choice constructs brings to light that the stable semantics as they are currently defined in non-deterministic AFT are too restrictive. This leads to the generaliziation of the stable semantics to what we call the *constructive stable semantics*. We

21st International Workshop on Nonmonotonic Reasoning, September 2–4, 2023, Rhodes, Greece
jesse.heyninck@ou.nl (J. Heyninck)
https://sites.google.com/view/jesseheyninck (J. Heyninck)
0000-0002-3825-4052 (J. Heyninck)

Preorder	Type	Definition
		Element Orders
$\leq_i$	$\wp(\mathcal{A}_\mathcal{P})^2 \times \wp(\mathcal{A}_\mathcal{P})^2$	$(x_1, y_1) \leq_i (x_2, y_2)$ iff $x_1 \subseteq x_2$ and $y_1 \supseteq y_2$
$\leq_t$	$\wp(\mathcal{A}_\mathcal{P})^2 \times \wp(\mathcal{A}_\mathcal{P})^2$	$(x_1, y_1) \leq_t (x_2, y_2)$ iff $x_1 \subseteq x_2$ and $y_1 \subseteq y_2$
		Set-based Orders
$\preceq_L^S$	$\wp(\wp(\mathcal{A}_\mathcal{P})) \times \wp(\wp(\mathcal{A}_\mathcal{P}))$	$X \preceq_L^S Y$ iff for every $y \in Y$ there is an $x \in X$ s.t. $x \subseteq y$
$\preceq_L^H$	$\wp(\wp(\mathcal{A}_\mathcal{P})) \times \wp(\wp(\mathcal{A}_\mathcal{P}))$	$X \preceq_L^H Y$ iff for every $x \in X$ there is an $y \in Y$ s.t. $x \subseteq y$
$\preceq_i^A$	$\wp(\wp(\mathcal{A}_\mathcal{P}))^2 \times \wp(\wp(\mathcal{A}_\mathcal{P}))^2$	$(X_1, Y_1) \preceq_i^A (X_2, Y_2)$ iff $X_1 \preceq_L^S X_2$ and $Y_2 \preceq_L^H Y_1$

Table 1
List of the preorders used in this paper (instantiated for the lattice $\langle \mathcal{A}_\mathcal{P}, \subseteq \rangle$).

can show that these semantics generalize existing semantics for choice programs [1] to the three-valued case. Furthermore, we are able to show that the differences between semantics for choice logic programs and disjunctive logic programs can be explained exactly by these two notions of stable semantics, which coincide for normal logic programs. As such, our study also sheds more light on the foundations of current implementations of logic programming.

Outline of the Paper: This paper is constructed as follows. In Section 2, the necessary background on disjunctive logic programs and non-deterministic approximation fixpoint theory is given. In Section 3, we formulate a non-deterministic approximation operator for choice programs. In Section 4, we show that this captures existing supported model semantics and leads to a natural, three-valued generalization. In Section 5, we turn to the stable semantics, and show that, when generalizing the stable semantics as defined for non-deterministic AFT, answer set semantics from the literature can be represented. In Section 6, we make some observations on the relation between choice constructs and disjunctions. Related work is discussed and the paper is concluded in Section 7.

Notational Conventions and Defintiions: To increase the readability of this paper, we have provided a summary of the main notationa conventions and definitions in Tables 1 and 2.

2. Background and Preliminaries

In this section, we recall disjunctive logic programming and choice programs (Sec. 2.1) and non-deterministic operators (Sec. 2.2).

2.1. Disjunctive Logic Programming and Choice Rules

Disjunctive Logic Programs In what follows we consider a propositional[1] language $\mathfrak{L}$, whose atomic formulas are denoted by p, q, r (possibly indexed), and that contains the propositional constants T (representing truth), F (falsity), U (unknown), and C (contradictory information). The connectives in $\mathfrak{L}$ include negation $\neg$, conjunction $\wedge$ and disjunction $\vee$. Formulas are denoted by ϕ, ψ, δ (again, possibly indexed). Logic programs in $\mathfrak{L}$ may be divided to different kinds as follows: a (propositional) *disjunctive logic program* $\mathcal{P}$ in $\mathfrak{L}$ (a dlp in short) is a finite set of rules of the form $\bigvee_{i=1}^n p_i \leftarrow \psi$, where the head $\bigvee_{i=1}^n p_i$ is a non-empty disjunction of atoms, and the body ψ is a formula in $\mathfrak{L}$. A rule is called *normal*, if its body is a conjunction of literals (i.e., atomic formulas or negated atoms), and its head is atomic. A rule is *disjunctively normal* if its body is a conjunction of literals and its head is a non-empty disjunction of atoms. We will use these denominations for programs if all rules in the program satisfy the denomination, e.g. a program is normal if all its rules are normal. The set of atoms occurring in a logic program $\mathcal{P}$ is denoted $\mathcal{A}_\mathcal{P}$.

The semantics of dlps are given in terms of *four-valued interpretations*. A *four-valued interpretation* of a program $\mathcal{P}$ is a pair (x, y), where $x \subseteq \mathcal{A}_\mathcal{P}$ is the set of the atoms that are assigned a value in $\{\mathsf{T}, \mathsf{C}\}$ and $y \subseteq \mathcal{A}_\mathcal{P}$ is the set of atoms assigned a value in $\{\mathsf{T}, \mathsf{U}\}$. Furthermore, let the involution $-$ is defined by $-\mathsf{T} = \mathsf{F}$, $-\mathsf{F} = \mathsf{T}$, $-\mathsf{U} = \mathsf{U}$ and $-\mathsf{C} = \mathsf{C}$. Truth assignments to complex formulas are then recursively defined as follows:

- $(x, y)(p) = \begin{cases} \mathsf{T} & \text{if } p \in x \text{ and } p \in y, \\ \mathsf{U} & \text{if } p \notin x \text{ and } p \in y, \\ \mathsf{F} & \text{if } p \notin x \text{ and } p \notin y, \\ \mathsf{C} & \text{if } p \in x \text{ and } p \notin y. \end{cases}$

[1] For simplicity we restrict ourselves to the propositional case.

Immediate Consequence Operator for dlps	
$HD_{\mathcal{P}}(x)$	$= \{\Delta \mid \bigvee\Delta \leftarrow \psi \in \mathcal{P}$ and $(x,x)(\psi) = \mathsf{T}\}$,
$IC_{\mathcal{P}}(x)$	$= \{y \subseteq \bigcup HD_{\mathcal{P}}(x) \mid \forall\Delta \in HD_{\mathcal{P}}(x),\ y \cap \Delta \neq \emptyset\}$.
Immediate Consequence Operator for choice programs	
$IC^c_{\mathcal{P}}(x) =$	$\{z \subseteq \bigcup_{r \in \mathcal{P}(x)} \mathsf{dom}(\mathsf{hd}(r)) \mid \forall r \in \mathcal{P}(x) : z \models \mathsf{hd}(r)\}$
Ndao $\mathcal{IC}_{\mathcal{P}}$ for disjunctive logic programs	
$\mathcal{HD}^l_{\mathcal{P}}(x,y)$	$= \{\Delta \mid \bigvee\Delta \leftarrow \phi \in \mathcal{P}, (x,y)(\phi) \geq_t \mathsf{C}\}$,
$\mathcal{HD}^u_{\mathcal{P}}(x,y)$	$= \{\Delta \mid \bigvee\Delta \leftarrow \phi \in \mathcal{P}, (x,y)(\phi) \geq_t \mathsf{U}\}$,
$\mathcal{IC}^\dagger_{\mathcal{P}}(x,y)$	$= \{x_1 \subseteq \bigcup \mathcal{HD}^\dagger_{\mathcal{P}}(x,y) \mid \forall\Delta \in \mathcal{HD}^\dagger_{\mathcal{P}}(x,y),\ x_1 \cap \Delta \neq \emptyset\}$ $(\dagger \in \{l,u\})$,
$\mathcal{IC}_{\mathcal{P}}(x,y)$	$= (\mathcal{IC}^l_{\mathcal{P}}(x,y), \mathcal{IC}^u_{\mathcal{P}}(x,y))$.
Ndao $\mathcal{IC}^c_{\mathcal{P}}$ for choice programs	
$\mathcal{HD}^{c,l}_{\mathcal{P}}(x,y)$	$= \{C \mid C \leftarrow \phi \in \mathcal{P}, (x,y)(\phi) \in \{\mathsf{T},\mathsf{C}\}\}$,
$\mathcal{HD}^{c,u}_{\mathcal{P}}(x,y)$	$= \{C \mid C \leftarrow \phi \in \mathcal{P}, (x,y)(\phi) \in \{\mathsf{T},\mathsf{U}\}\}$,
$\mathcal{IC}^{c,\dagger}_{\mathcal{P}}(x,y)$	$= \{z \subseteq \bigcup_{C \in \mathcal{HD}^{c,\dagger}_{\mathcal{P}}(x,y)} \mathsf{dom}(C) \mid \forall C \in \mathcal{HD}^{c,l}_{\mathcal{P}}(x,y), z \cap \mathsf{dom}(C) \in \mathsf{sat}(C)\}$ $(\dagger \in \{l,u\})$,
$\mathcal{IC}^c_{\mathcal{P}}(x,y)$	$= (\mathcal{IC}^{c,l}_{\mathcal{P}}(x,y), \mathcal{IC}^{c,u}_{\mathcal{P}}(x,y))$.

Table 2
Concrete Operators for dlps and choice programs

- $(x,y)(\neg\phi) = -(x,y)(\phi)$,
- $(x,y)(\psi \wedge \phi) = lub_{\leq_t}\{(x,y)(\phi),(x,y)(\psi)\}$,
- $(x,y)(\psi \vee \phi) = glb_{\leq_t}\{(x,y)(\phi),(x,y)(\psi)\}$.

A four-valued interpretation of the form (x,x) may be associated with a *two-valued* (or *total*) interpretation x. (x,y) is a *three-valued* (or *consistent*) interpretation, if $x \subseteq y$. Interpretations are compared by two order relations: the *information order*, which is defined as $(x,y) \leq_i (w,z)$ iff $x \subseteq w$ and $z \subseteq y$, and the *truth order*, where $(x,y) \leq_t (w,z)$ iff $x \subseteq w$ and $y \subseteq z$.

An extension to dlps of the immediate consequence operator for normal programs [11] is defined as follows:

Definition 1 (Immediate Consquence operator for dlps)**.** *Given a dlp $\mathcal{P}$ and a two-valued interpretation x, we define:*

- $HD_{\mathcal{P}}(x) = \{\Delta \mid \bigvee\Delta \leftarrow \psi \in \mathcal{P}$ *and* $(x,x)(\psi) = \mathsf{T}\}$.
- $IC_{\mathcal{P}}(x) = \{y \subseteq \bigcup HD_{\mathcal{P}}(x) \mid \forall\Delta \in HD_{\mathcal{P}}(x),\ y \cap \Delta \neq \emptyset\}$.

Thus, $IC_{\mathcal{P}}(x)$ consists of sets of atoms that occur in activated rule heads, each sets contains at least one representative from every disjuncts of a rule in $\mathcal{P}$ whose body is satisfied by x. Denoting by $\wp(\mathcal{S})$ the powerset of $\mathcal{S}$, $IC_{\mathcal{P}}$ is an operator on the lattice $\langle\wp(\mathcal{A}_{\mathcal{P}}),\subseteq\rangle$.

Given a dlp $\mathcal{P}$ a consistent interpretation (x,y) is a *(three–valued) model* of $\mathcal{P}$, if for every $\phi \leftarrow \psi \in \mathcal{P}$, $(x,y)(\phi) \geq_t (x,y)(\psi)$. The GL-transformation $\frac{\mathcal{P}}{(x,y)}$ of a disjunctively normal dlp $\mathcal{P}$ with respect to a consistent interpretation (x,y), is the positive program obtained by replacing in every rule in $\mathcal{P}$ of the form $p_1 \vee \ldots \vee p_n \leftarrow \bigwedge_{i=1}^m q_i \wedge \bigwedge_{j=1}^n \neg r_j$ a negated literal $\neg r_i$ $(1 \leq i \leq k)$ by $(x,y)(\neg r_i)$. (x,y) is a *three-valued stable model* of $\mathcal{P}$ iff it is a $\leq_t$-minimal model of $\frac{\mathcal{P}}{(x,y)}$. [2]

Choice Rules Choice constructs have been studied in several works on logic programming [1], and are, among others, part of the ASP-Core-2 standard [12]. We define a *choice atom* relative to a set of atoms $\mathcal{A}$ as an expression $C = (\mathsf{dom},\mathsf{sat})$ where $\mathsf{dom} \subseteq \mathcal{A}$ and $\mathsf{sat} \subseteq \wp(\mathsf{dom})$. Intuitively, dom denotes the *domain* of C, i.e. the atoms relevant for the evaluation of C, whereas sat is the set of *satisfiers* of C. We also denote, for $C = (\mathsf{dom},\mathsf{sat})$, dom by $\mathsf{dom}(C)$ and sat by $\mathsf{sat}(C)$. For a concrete example, consider $1 \leq \{p,q,r\} \leq 2$ which intuitively states that between 1 and 2 of the atoms p, q and r have to be true, corresponds to the choice atom $(\{p,q,r\},\{\{p\},\{q\},\{r\},\{p,q\},\{p,r\},\{q,r\}\})$ (notice that $\{p,q,r\}$ is the domain and not a satisfier of the choice atom). For such a set constraint, we assume the domain and satisfiers are clear and can be left implicit.

Where C is a choice atom and ϕ is a formula in $\mathcal{L}$, rule $C \leftarrow \phi$ is a *choice rule*. C is called the head (denoted $\mathsf{hd}(r)$) and ϕ the body of a choice rule $C \leftarrow \phi$. If ϕ is a conjunction of literals, we call it a *normal choice rule*. A choice atom C is *monotone* if $\mathsf{dom}(C) \cap x \in \mathsf{sat}(C)$ implies $\mathsf{dom}(C) \cap x' \in \mathsf{sat}(C)$

[2] An overview of other semantics for dlps can be found in previous work on non-deterministic AFT [8].

for any $x \subseteq x' \subseteq \mathcal{A}_\mathcal{P}$. A choice program is a set of choice rules, and it is normal respectively monotone if all of the rules are normal respectively all the heads of rules are monotone.

We now recall the *supported model*-semantics [1] for choice programs. A set $x \subseteq \mathcal{A}_\mathcal{P}$ *satisfies* a constraint if $\mathsf{dom}(C) \cap x \in \mathsf{sat}(C)$. x satisfies a rule r if x satisfies the head of r or does not satisfy some literal in the body of r. x is a model of $\mathcal{P}$ if it satisfies every rule in $\mathcal{P}$. A rule $r \in \mathcal{P}$ is *x-applicable* if x satisfies the body of r, and the set of x-applicable rules in $\mathcal{P}$ is denoted by $\mathcal{P}(x)$. x is a *supported model* of $\mathcal{P}$ if it is a model and $x \subseteq \bigcup_{r\in\mathcal{P}(x)} \mathsf{dom}(\mathsf{hd}(r))$.

We now recall the *answer set semantics* for choice programs by Liu, Pontelli, Son and Truszczyński [1], which we call the LPST-answer set semantics. Let $x \subseteq \mathcal{A}_\mathcal{P}$ be given. A set z is *non-deterministically one-step provable from x by means of $\mathcal{P}$,* if $z \subseteq \bigcup_{r\in\mathcal{P}(x)} \mathsf{dom}(\mathsf{hd}(r))$ and $z \models \mathsf{hd}(r)$ for every $r \in \mathcal{P}(x)$. The operator $IC^c_\mathcal{P} : \wp(\mathcal{A}_\mathcal{P}) \to \wp(\wp(\mathcal{A}_\mathcal{P}))$ is defined as: $IC^c_\mathcal{P}(x) =$

$$\{z \subseteq \bigcup_{r\in\mathcal{P}(x)} \mathsf{dom}(\mathsf{hd}(r)) \mid \forall r \in \mathcal{P}(x) : z \models \mathsf{hd}(r)\}$$

We now recall the definition of a *computation*:

Definition 2. *A sequence of sets of atoms $\langle x_i\rangle_{i=0}^\infty$ is a* computation *for $\mathcal{P}$ if $x_0 = \emptyset$ and the sequence satisfies the following principles:*

Convergence *: $x_\infty \in IC^c_\mathcal{P}(x_\infty)$*

Persistence of reasons *: There is a sequence of programs $\langle P_i\rangle_{i=0}^\infty$ such that for every $i \geq 0$, $P_i \subseteq P_{i+1}$, $P_i \subseteq P(x_i)$ and $x_{i+1} \in IC^c_{\mathcal{P}_i}(x_i)$.*

Revision *For every $i > 0$, there is some $\mathcal{P}' \subseteq \mathcal{P}(x_{i-1})$ with $x_i \in IC^c_{\mathcal{P}'}(x_{i-1})$.*

Persistence of beliefs *$x_i \subseteq x_{i+1}$ for every $i \geq 0$.*

A set x is a LPST-*answer set* if there is a computation for $\mathcal{P}$ whose result is x.

Example 1. *Consider the program $\mathcal{P} = \{1 \leq \{p, r\} \leq 2 \leftarrow \neg r; p \leftarrow q; q \leftarrow p\}$. We see that $\{p, q\}$ is a supported model, as it is a model of $\mathcal{P}$ and $\{p, q\} \subseteq \bigcup_{r\in\mathcal{P}(\{p,q\})} \mathsf{dom}(\mathsf{hd}(r))$ (since $\mathcal{P}(\{p, q\}) = \mathcal{P}$). $\{p, q\}$ is also a* LPST-*answer set of $\mathcal{P}$. This is seen by observing that the following is a computation for $\mathcal{P}$: $\langle\emptyset, \{p\}, \{p, q\}\rangle$.*

2.2. Non-Deterministic Approximation Fixpoint Theory

We now recall basic notions from non-deterministic approximation fixpoint theory (AFT) by Heyinck, Arieli and Bogaerts 2022, which generalizes approximation fixpoint theory as introduced by Denecker, Marek and Truszczyński (2000) to non-deterministic operators, i.e. operators which map elements of a lattice to a set of elements of that lattice (like the operator $IC_\mathcal{P}$ over $\langle\mathcal{A}_\mathcal{P}, \subseteq\rangle$ for dlps). We recall the necessary details on non-deterministic AFT, referring to the original paper [8] for more details and explanations.

A *non-deterministic operator on $\mathcal{L}$* is a function $O : \mathcal{L} \to \wp(\mathcal{L})\setminus\{\emptyset\}$. For example, the operator $IC_\mathcal{P}$ from Definition 1 is a non-deterministic operator on the lattice $\langle\wp(\mathcal{A}_\mathcal{P}), \subseteq\rangle$.

As the ranges of non-deterministic operators are *sets* of lattice elements, one needs a way to compare them, such as the *Smyth order* and the *Hoare order*. Let $L = \langle\mathcal{L}, \leq\rangle$ be a lattice, and let $X, Y \in \wp(\mathcal{L})$. Then: $X \preceq^S_L Y$ if for every $y \in Y$ there is an $x \in X$ such that $x \leq y$; and $X \preceq^H_L Y$ if for every $x \in X$ there is a $y \in Y$ such that $x \leq y$. Given some $X_1, X_2, Y_1, Y_2 \subseteq \mathcal{L}$, $X_1 \times Y_1 \preceq^A_i X_2 \times Y_2$ iff $X_1 \preceq^S_L X_2$ and $Y_2 \preceq^H_L Y_1$; and $X_1 \times Y_1 \preceq^S_t X_2 \times Y_2$ iff $X_1 \preceq^S_L X_2$ and $Y_2 \preceq^S_L Y_1$

Let $L = \langle\mathcal{L}, \leq\rangle$ be a lattice. Given an operator $\mathcal{O} : \mathcal{L}^2 \to \mathcal{L}^2$, we denote by $\mathcal{O}_l$ the projection operator defined by $\mathcal{O}_l(x, y) = \mathcal{O}(x, y)_1$, and similarly for $\mathcal{O}_u(x, y) = \mathcal{O}(x, y)_2$. An operator $\mathcal{O} : \mathcal{L}^2 \to \wp(\mathcal{L})\setminus\emptyset \times \wp(\mathcal{L})\setminus\emptyset$ is called a *non-deterministic approximating operator* (ndao, for short), if it is $\preceq^A_i$-monotonic (i.e. $(x_1, y_1) \leq_i (x_2, y_2)$ implies $\mathcal{O}(x_1, y_1) \preceq^A_i \mathcal{O}(x_2, y_2)$), and is *exact* (i.e., for every $x \in \mathcal{L}$, $\mathcal{O}(x, x) = \mathcal{O}_l(x, x) \times \mathcal{O}_l(x, x)$). We restrict ourselves to ndaos ranging over consistent pairs (x, y). A non-deterministic operator $O : \mathcal{L} \to \wp(\mathcal{L})$ is *downward closed* if for every sequence $X = \{x_\epsilon\}_{\epsilon<\alpha}$ of elements in $\mathcal{L}$ such that: (1) for every $\epsilon < \alpha$, $O(x_\epsilon) \preceq^S_L \{x_\epsilon\}$, and (2) for every $\epsilon < \epsilon' < \alpha$, $x_{\epsilon'} < x_\epsilon$, it holds that $O(glb(X)) \preceq^S_L glb(X)$.

The stable operator (given an ndao $\mathcal{O}$) is defined as follows. The *complete lower stable operator* is defined by (for any $y \in \mathcal{L}$) $C(\mathcal{O}_l)(y) = \{x \in \mathcal{L} \mid x \in \mathcal{O}_l(x, y)$ and $\neg\exists x' < x : x' \in \mathcal{O}_l(x', y)\}$. The *complete upper stable operator* is defined by (for any $x \in \mathcal{L}$) $C(\mathcal{O}_u)(x) = \{y \in \mathcal{L} \mid y \in \mathcal{O}_u(x, y)$ and $\neg\exists y' < y : y' \in \mathcal{O}_u(x, y')\}$. The *stable operator*: $S(\mathcal{O})(x, y) = C(\mathcal{O}_l)(y) \times C(\mathcal{O}_u)(x)$. (x, y) is a *stable fixpoint* of $\mathcal{O}$ if $(x, y) \in S(\mathcal{O})(x, y)$.[3]

Other semantics, e.g. the well-founded state and the Kripke-Kleene fixpoints and state are defined

[3] Notice that we slightly abuse notation and write $(x, y) \in S(\mathcal{O})(x, y)$ to abbreviate $x \in (S(\mathcal{O})(x, y))_1$ and $y \in (S(\mathcal{O})(x, y))_2$, i.e. x is a lower bound generated by $S(\mathcal{O})(x, y)$ and y is an upper bound generated by $S(\mathcal{O})(x, y)$.

by Heyninck et al (2022) and can be immediately obtained once an ndao is formulated. Due to space limitations, these semantics are not discussed here.

Example 2. *An example of an ndao approximating $IC_\mathcal{P}$ (Definition 1) is defined as follows (given a dlp $\mathcal{P}$ and an interpretation (x,y)):*

- $\mathcal{HD}^l_\mathcal{P}(x,y) = \{\Delta \mid \bigvee\Delta \leftarrow \phi \in \mathcal{P}, (x,y)(\phi) \geq_t \mathsf{C}\}$,
- $\mathcal{HD}^u_\mathcal{P}(x,y) = \{\Delta \mid \bigvee\Delta \leftarrow \phi \in \mathcal{P}, (x,y)(\phi) \geq_t \mathsf{U}\}$,
- $\mathcal{IC}^\dagger_\mathcal{P}(x,y) = \{x_1 \subseteq \bigcup\mathcal{HD}^\dagger_\mathcal{P}(x,y) \mid \forall\Delta \in \mathcal{HD}^\dagger_\mathcal{P}(x,y),\ x_1 \cap \Delta \neq \emptyset\}$ *(for $\dagger \in \{l,u\}$),*
- $\mathcal{IC}_\mathcal{P}(x,y) = (\mathcal{IC}^l_\mathcal{P}(x,y), \mathcal{IC}^u_\mathcal{P}(x,y))$.

$\mathcal{IC}^l_\mathcal{P}$ behaves as follows for $\mathcal{P} = \{p \vee q \leftarrow \neg q\}$:

- *For any interpretation (x,y) for which $q \in x$, $\mathcal{HD}^l_\mathcal{P}(x,y) = \emptyset$ and thus $\mathcal{IC}^l_\mathcal{P}(x,y) = \{\emptyset\}$.*
- *For any interpretation (x,y) for which $q \notin x$, $HD^l_\mathcal{P}(x,y) = \{\{p,q\}\}$ and thus $\mathcal{IC}^l_\mathcal{P}(x,y) = \{\{p\},\{q\},\{p,q\}\}$.*

Since $\mathcal{IC}^l_\mathcal{P}(x,y) = \mathcal{IC}^u_\mathcal{P}(y,x)$ (see [8, Lemma 1]), $\mathcal{IC}_\mathcal{P}$ behaves as follows:

- *For any (x,y) with $q \notin x$ and $q \notin y$, $\mathcal{IC}_\mathcal{P}(x,y) = \{\{p\},\{q\},\{p,q\}\} \times \{\{p\},\{q\},\{p,q\}\}$,*
- *For any (x,y) with $q \notin x$ and $q \in y$, $\mathcal{IC}_\mathcal{P}(x,y) = \{\emptyset\} \times \{\{\{p\},\{q\},\{p,q\}\}$,*
- *For any (x,y) with $q \in x$ and $q \notin y$, $\mathcal{IC}_\mathcal{P}(x,y) = \{\{p\},\{q\},\{p,q\}\} \times \{\emptyset\}$, and*
- *For any (x,y) with $q \in x$ and $q \in y$, $\mathcal{IC}_\mathcal{P}(x,y) = \{(\emptyset,\emptyset)\}$.*

We see e.g. that $C(\mathcal{IC}^l_\mathcal{P})(\{p\}) = \{\{p\},\{q\}\}$ and thus $(\{p\},\{p\})$ is a stable fixpoint of $\mathcal{IC}_\mathcal{P}$. $(\emptyset,\{q\})$ is the second stable fixpoint of $\mathcal{IC}_\mathcal{P}$. $(\emptyset,\{p,q\})$ is a fixpoint of $\mathcal{IC}_\mathcal{P}$ that is not stable.

The operator $\mathcal{IC}_\mathcal{P}$ faithfully represents the semantics of dlps: In general, (total) stable fixpoints of $\mathcal{IC}_\mathcal{P}$ correspond to (total) stable models of $\mathcal{P}$ [13], and weakly supported models of $\mathcal{IC}_\mathcal{P}$ [14] correspond to fixpoints of $\mathcal{IC}_\mathcal{P}$ [8].

3. Approximation Operator for Choice Programs

In order to apply non-deterministic AFT to obtain semantics for choice programs, the first task is to formulate an *non-deterministic approximation operator* $\mathcal{IC}^c_\mathcal{P}$ that approximates the immediate consequence operator $IC^c_\mathcal{P}$. The idea of designing approximation operators for rules using choice constructs in rules is quite similar to that of disjunctions in disjunctive logic programs: given an interpretation (x,y), $\mathcal{IC}^{c,l}_\mathcal{P}(x,y)$ should satisfy all choice constructs C for which a rule $C \leftarrow \phi \in \mathcal{P}$ has a body ϕ that is made true by (x,y) (and similarly for $\mathcal{IC}^{u,c}_\mathcal{P}(x,y)$). In more formal detail, first, we define the activated heads $\mathcal{HD}^{c,l}_\mathcal{P}$ simple as the heads of rules whose body is true or inconsistent.

$$\mathcal{HD}^{c,l}_\mathcal{P}(x,y) = \quad \{C \mid C \leftarrow \phi \in \mathcal{P}, (x,y)(\phi) \in \{\mathsf{T},\mathsf{C}\}\}$$

The lower bound of the immediate consequence relation is now defined as the subsets of atoms occuring in satisfiers of activated heads that contain a satisfier for every every head rule, i.e. $\mathcal{IC}^{c,l}_\mathcal{P}(x,y) =$

$$\{z \subseteq \bigcup_{C \in \mathcal{HD}^{c,l}_\mathcal{P}(x,y)} \mathsf{dom}(C) \mid \quad \forall C \in \mathcal{HD}^{c,l}_\mathcal{P}(x,y),\ z \cap \mathsf{dom}(C) \in \mathsf{sat}(C)\}$$

The upper bound of the operator can be defined symmetrically as $\mathcal{IC}^{c,u}_\mathcal{P}(x,y) = \mathcal{IC}^{c,l}_\mathcal{P}(y,x)$. It can easily be checked that this corresponds to defining an $\mathcal{IC}^{c,u}_\mathcal{P}$-operator based on heads that occur in rules whose body is U or T:

$$\mathcal{HD}^{c,u}_\mathcal{P}(x,y) = \quad \{C \mid C \leftarrow \phi \in \mathcal{P}, (x,y)(\phi) \in \{\mathsf{U},\mathsf{T}\}\}$$

Finally, $IC^c_\mathcal{P}(x,y)$ is defined by combining the lower and upper bound operators:

$$\mathcal{IC}^c_\mathcal{P}(x,y) = (\mathcal{IC}^{c,l}_\mathcal{P}(x,y), \mathcal{IC}^{c,u}_\mathcal{P}(x,y))$$

Example 3. *Consider again the program $\mathcal{P}$ from Example 1. We see e.g. that $\mathcal{IC}^c_\mathcal{P}(\emptyset,\{r\}) = \{\emptyset\} \times \{\{p\},\{q\},\{p,q\}\}$ as $\mathcal{HD}^{c,l}_\mathcal{P}(\emptyset,\{r\}) = \emptyset$ whereas $HDc^{c,u}_\mathcal{P}(\emptyset,\{r\}) = \{\{p,q\}\}$. On the other hand, $\mathcal{IC}^c_\mathcal{P}(\{p\},\{p\}) = \{\{q\},\{p,q\}\} \times \{\{q\},\{p,q\}\}$ as $\mathcal{HD}^{c,l}_\mathcal{P}(\{p\},\{p\}) = \{\{q\},\{p,q\}\}$.*

It should be noticed that this operator is not well-defined for every program:

Example 4. *Let $\mathcal{P} = \{1 < \{p,q\} < 2 \leftarrow; 2 < \{p,q\} \leftarrow\}$. For this program, no sets satisfying both choice constructs exist, and thus $\mathcal{IC}^c_\mathcal{P}(x,y)$ is not defined (for any $x,y \subseteq \mathcal{A}_\mathcal{P}$).*

In what follows, for simplicity, we restrict attention to programs for which $\mathcal{IC}^c_\mathcal{P}(x,y) \neq \emptyset$ for any $x,y \subseteq \mathcal{A}_\mathcal{P}$.

The operator $\mathcal{IC}^c_\mathcal{P}$ is a symmetric approximation of the immediate consequence operator $IC^c_\mathcal{P}$ known from the literature [1]:

Proposition 1. *$\mathcal{IC}^c_{\mathcal{P}}$ is a symmetric non-deterministic approximation operator approximating $IC^c_{\mathcal{P}}$.*

Proof. Symmetry follows by the definition of the operator. For $\preceq^A_i$*-monotonicity*, consider some $(x_1, y_1) \leq_i (x_2, y_2)$. We first show $\preceq^S_L$-monotonicity of $\mathcal{IC}^{c,l}_{\mathcal{P}}$. By [8, Proof of Proposition 1], $(x_1, y_1)(\phi) \geq_t \mathsf{T}$ implies $(x_2, y_2)(\phi) \geq_t \mathsf{T}$ and thus $\mathcal{HD}^{c,l}_{\mathcal{P}}(x_1, y_1) \subseteq \mathcal{HD}^{c,l}_{\mathcal{P}}(x_2, y_2)$. Consider some $z_2 \in IC^{c,l}_{\mathcal{P}}(x_2, y_2)$. We show that $z_2 \cap \bigcup\{\mathsf{dom}(C) \mid C \leftarrow \phi \in \mathcal{P}, (x_1, y_1)(\phi) \in \{\mathsf{C}, \mathsf{T}\} \in IC^{c,l}_{\mathcal{P}}(x_1, y_1)$. Indeed, consider some $C \in \mathcal{HD}^{c,l}_{\mathcal{P}}(x_1, y_1)$. Then $C \in \mathcal{HD}^{c,l}_{\mathcal{P}}(x_2, y_2)$ and thus $\mathsf{dom}(C) \cap Z_2 \in \mathsf{sat}(C)$. As $z_2 \cap \bigcup\{\mathsf{dom}(C) \mid C \leftarrow \phi \in \mathcal{P}, (x_1, y_1)(\phi) \in \{\mathsf{C}, \mathsf{T}\} \subseteq z_2$, this concludes the proof of $\preceq^S_L$-monotonicity. $\preceq^H_L$-monotonicity follows from symmetry and [8, Lemma 3]. *Exactness* is immediate, as $\mathcal{IC}^{c,l}_{\mathcal{P}}(x, x) = \mathcal{IC}^{c,u}_{\mathcal{P}}(x, x)$ for any $x \subseteq \mathcal{A}_{\mathcal{P}}$. That *$\mathcal{IC}^c_{\mathcal{P}}$ approximates $IC^c_{\mathcal{P}}$* is immediate. □

4. Supported Model Semantics

We now turn to the study of the semantics for choice programs obtained on the basis of our approximation operator $\mathcal{IC}^c_{\mathcal{P}}$. In this section, we look at the fixpoints of $\mathcal{IC}^c_{\mathcal{P}}$, which we will see give a natural four-valued generalisation of the supported model semantics by [1]. Stable semantics are studied in Section 5.

A first insight is that exact fixpoints of $\mathcal{IC}^c_{\mathcal{P}}$ coincide with the *supported models* of Liu et al [1].

Proposition 2. *Let a normal constraint program $\mathcal{P}$ be given. Then $x \in \mathcal{IC}^{c,l}_{\mathcal{P}}(x, x)$ iff x is a supported model of $\mathcal{P}$.*

Proof. For the $\Rightarrow$-direction, suppose that $x \in IC^{c,l}_{\mathcal{P}}(x, x)$. We first show x is a model of $\mathcal{P}$. Indeed, consider a rule $C \leftarrow C_1, \ldots, C_n$ s.t. x satisfies C_i for $i = 1, \ldots, n$. Then $(x, x)(C_i) = \mathsf{T}$ and thus $C \in \mathcal{HD}^{c,l}_{\mathcal{P}}(x, x)$, which implies, since $x \in IC^{c,l}_{\mathcal{P}}(x, x)$, that $x \cap \mathsf{dom}(C) \in \mathsf{sat}(C)$. That x is supported follows immediately from the fact that $x \in IC^{c,l}_{\mathcal{P}}(x, x)$ implies $z \subseteq \bigcup_{C \in \mathcal{HD}^{c,l}_{\mathcal{P}}(x,x)} \mathsf{dom}(C)$. The $\Leftarrow$-direction is analogous. □

We now generalize this result to the three-valued case. In order to do this, we first have to generalize the four-valued truth-assignments to choice constructs. The following forms a generalization of assignment of truth-values to choice constructs that forms a natural generalization of the assignment of atoms to choice constructs. Other notions will be investigated in future work.

Definition 3. *Given a choice construct C and an interpretation (x, y), we say that:*

- $(x, y)(C) = \mathsf{T}$ *if* $x \cap \mathsf{dom}(c) \in \mathsf{sat}(C)$ *and* $y \cap \mathsf{dom}(c) \in \mathsf{sat}(C)$,
- $(x, y)(C) = \mathsf{F}$ *if* $x \cap \mathsf{dom}(c) \notin \mathsf{sat}(C)$ *and* $y \cap \mathsf{dom}(c) \notin \mathsf{sat}(C)$,
- $(x, y)(C) = \mathsf{C}$ *if* $x \cap \mathsf{dom}(c) \notin \mathsf{sat}(C)$ *and* $y \cap \mathsf{dom}(c) \in \mathsf{sat}(C)$,
- $(x, y)(C) = \mathsf{U}$ *if* $x \cap \mathsf{dom}(c) \in \mathsf{sat}(C)$ *and* $y \cap \mathsf{dom}(c) \notin \mathsf{sat}(C)$.

We can now define *three-valued models* of $\mathcal{P}$ as consistent interpretations (x, y) for which $(x, y)(\phi) \geq_t (x, y)(C)$ for every $C \leftarrow \phi \in \mathcal{P}$, and *supported*[4] models as models (x, y) of $\mathcal{P}$ s.t. for every $p \in y$, there is a rule $C \leftarrow \phi \in \mathcal{P}$ such that $p \in \mathsf{dom}(C) \cap x$ and $(x, y)(\phi) \geq_t (x, y)(p)$. In other words, a model is supported if for every atom p that is not false, we have a reason in the form of an activated rule for accepting (or not rejecting) that atom.

We can now show that three-valued supported models of $\mathcal{P}$ coincide with pre-fixpoints (respectively fixpoints) of $\mathcal{IC}^c_{\mathcal{P}}$:

Proposition 3. *Let some normal choice program $\mathcal{P}$ be given. Then (x, y) is a three-valued supported model of $\mathcal{P}$ iff $(x, y) \in \mathcal{IC}^c_{\mathcal{P}}(x, y)$.*

Proof. For the $\Rightarrow$-direction, suppose that (x, y) is a three-valued supported model of $\mathcal{P}$. As for every $p \in y$, there is a rule $C \leftarrow \phi \in \mathcal{P}$ such that $p \in \mathsf{dom}(C) \cap x$ and $(x, y)(\phi) \geq_t (x, y)(p)$, $x \subseteq \bigcup_{C \in \mathcal{HD}^{c,l}_{\mathcal{P}}(x,y)} \mathsf{dom}(C)$ (and similarly for y). That for every $C \leftarrow \phi \in \mathcal{HD}^{c,l}_{\mathcal{P}}(x, y)$, $\mathsf{dom}(C) \cap x \in \mathsf{sat}(C)$ follows from (x, y) being a model of $\mathcal{P}$ (and similarly for y). For the $\Leftarrow$-direction, suppose $(x, y) \in IC^c_{\mathcal{P}}(x, y)$. That (x, y) is a model follows straightforwardly from the definition of $IC^c_{\mathcal{P}}(x, y)$. Consider now some $p \in y$. As $y \subseteq \bigcup_{C \in \mathcal{HD}^{c,u}_{\mathcal{P}}(x,y)} \mathsf{dom}(C)$, there is some $C \leftarrow \phi \in \mathcal{P}$ s.t. $p \in \mathsf{dom}(C) \cap y$ (and similarly for x). □

For many other non-monotonic formalisms, one can obtain an additional characterisation results, namely a correspondence between pre-fixpoints of an operator and models of the corresponding knowledge base. For the general case of choice constructs, this correspondence does not hold:

[4] In some works, these models have been called *weakly supported models* [14] for disjunctive logic programs.

Example 5. *Consider the program* $\mathcal{P} = \{\{p,q\} = 1 \leftarrow\}$. *Then* $(\{p,q\},\{p,q\})$ *is a pre-fixpoint of* $\mathcal{IC}^c_\mathcal{P}$ *(as* $\mathcal{IC}^c_\mathcal{P}(\{p,q\},\{p,q\}) = \{\{p\},\{q\}\}\times\{\{p\},\{q\}\} \preceq^S_t (\{p,q\},\{p,q\})$*) yet it is not a model (as the only models are* $\{p\}$ *and* $\{q\}$*).*

This non-correspondence between pre-fixpoints and models does not hold due to the non-monotonic nature of choice constructs: for monotone choice constructs the correspondence holds:

Proposition 4. *Let some normal choice program* $\mathcal{P}$ *with monotone choice constructs be given. Then* (x,y) *is a model of* $\mathcal{P}$ *iff* $\mathcal{IC}^c_\mathcal{P}(x,y) \preceq^S_t (x,y)$.

Proof. For the $\Rightarrow$-direction, suppose that (x,y) is a model of $\mathcal{P}$. We show that $x \cap \bigcup_{C\in\mathcal{HD}^{c,l}_\mathcal{P}(x,y)} \mathsf{dom}(C) \in \mathcal{IC}^c_\mathcal{P}(x,y)$ (a similar statement can be shown for y), which implies $x \preceq^S_L \mathcal{IC}^{c,l}_\mathcal{P}(x,y)$ (and similarly for y. Indeed, consider some $C \in \mathcal{HD}^{c,l}_\mathcal{P}(x,y)$, i.e. there is some $C \leftarrow \phi \in \mathcal{P}$ with $\phi \in \{\mathsf{T},\mathsf{C}\}$. Then $x \cap \mathsf{dom}(c) \in \mathsf{sat}(C)$. Thus $x \cap \mathsf{dom}(C) \in \mathcal{IC}^c_\mathcal{P}(x,y)$. For the $\Leftarrow$-direction, suppose that $C \leftarrow \phi \in \mathcal{P}$ and $(x,y)(\phi) = \mathsf{T}$. Then $C \in \mathcal{HD}^{c,l}_\mathcal{P}(x,y)$ and thus for every $x_1 \in \mathcal{IC}^{c,l}_\mathcal{P}(x,y)$, $x_1 \cap \mathsf{dom}(C) \in \mathsf{sat}(C)$. As $\mathcal{IC}^{c,l}_\mathcal{P}(x,y) \preceq^S_L x$ and C is monotone, $x_1 \cap \mathsf{dom}(C) \in \mathsf{sat}(C)$. □

5. Stable Semantics

We now move to the stable semantics. We first consider the stable semantics as defined by Heyninck, Arieli and Bogaerts [8]. There, the stable operator based on $\mathcal{O}$ was defined as the $\leq_t$-minimal fixpoints of $\mathcal{O}^l(.,y)$. The usefulness of the stable semantics as defined by Heyninck, Arieli and Bogaerts is demonstrated by the characterisation of the (partial) stable model semantics for disjunctive logic programs. However, for choice constructs, the selection of minimal fixpoints might be overly strong:

Example 6. *Consider the program* $\mathcal{P} = \{1 \leq \{p,q\} \leq 2 \leftarrow\}$. *Intuitively, this rule allows to choose between one and two among p and q. The stable version of* $IC^c_\mathcal{P}$ *behaves as follows:*

$$S(IC^{c,l}_\mathcal{P})(x) = \{\{p\},\{q\}\} \text{ for any } x \subseteq \mathcal{A}_\mathcal{P}$$

This means that this program has two stable fixpoints of $IC^c_\mathcal{P}$*:* $\{p\}$ *and* $\{q\}$*. This is clearly undesirable, as according to the intuitive reading of* $\mathcal{P}$*,* $\{p,q\}$ *should be allowed as a stable interpretation as well (notice this is also an* LPST*-answer set).*

In the context of choice operators, minimality is not a satisfactory generalisation of the ideas underlying the stable semantics for normal logic programs. If we take one step back, we can explain the choice for minimal fixpoints, and their shortcomings in the context of choice constructs, in stable non-deterministic operators as defined by Heyninck, Arieli and Bogaerts [8] as follows. For deterministic operators, the stable version of an approximation operator $\mathcal{O}$ is defined as the glb of fixpoints of $\mathcal{O}_l(.,y)$. For deterministic operators over finite lattices, the minimal fixpoint of $\mathcal{O}_l(.,x)$ is identical to the glb of fixpoints of $\mathcal{O}_l(.,y)$, and it is also identical to the fixpoint obtained by iterating $\mathcal{O}_l(.,y)$ starting from $\bot$ (i.e. $\bigcup_{i=1}^\infty \mathcal{O}^i_l(.,y)$. For non-deterministic operators, this correspondence does not hold. Indeed, as already observed by Heyninck, Arieli and Bogaerts [8], the glb of fixpoints of $\mathcal{O}(.,x)$ is often too weak (e.g. for the program $\mathcal{P}$ from Example 6 we get $\{p\} \cap \{q\} \cap \{p,q\} = \emptyset$ as the glb of fixpoints. However, this still leaves an alternative choice: namely looking at fixpoints reachable by applications of $\mathcal{O}(.,y)$ starting from $\bot$. In other words, we are interested in the fixpoints of $\mathcal{IC}^{c,l}_\mathcal{P}(.,y)$ that are grounded in the sense that we can justify them using a sequence of applications of $\mathcal{IC}^{c,l}_\mathcal{P}(.,y)$, starting from $\emptyset$. More colloquially, the fixpoint should be derivable *from the ground up.* To formalize this, we first generalize the notion of a well-founded sequence from [15].

Definition 4. *Given a non-deterministic operator* $O : \mathcal{L} \to \wp(\mathcal{L})$*, a sequence* $x_0,\ldots,x_n \subseteq \mathcal{L}$ *is* well-founded relative to O *if:*

- $x_0 = \bot$,
- $x_i \leq x_{i+1}$ *and* $x_{i+1} \in O(x_i)$ *for every successor ordinal* $i \geq 0$.
- $x_\lambda = (lub\{x_i\}_{i<\lambda})$ *for a limit ordinal* λ.[5]

We denote the well-founded sequences relative to O by $\mathsf{wfs}(O)$.

Notice that, in contradistinction to the deterministic version of a well-founded sequence, we require not merely that $x_{i+1} \preceq^S_L O(x_i)$ (or, in the case of deterministic operators O, $x_{i+1} < O(x_i)$) but $x_{i+1} \in O(x_i)$. This is to ensure that x_{i+1} can actually be constructed from x_i. For non-deterministic operators, this is not ensured by merely requiring $x_{i+1} \preceq^S_L O(x_i)$:

[5] Even though we assume finite programs, we define here well-founded sequences and the corresponding stable operator for the general case of non-deterministic operators O, and thus consider limit ordinals.

Example 7. *Let $\mathcal{P} = \{\{p,q\} = 2 \leftarrow\}$. If we would allow for $x_{i+1} \preceq_L^S O(x_i)$ in the second condition of Definition 4, $\emptyset, \{p\}$ would be a well-founded sequence according to $\mathcal{IC}_{\mathcal{P}}^{c,l}(.,y)$ (for any $y \subseteq \{p,q\}$) as $\{p\} \preceq^S \mathcal{IC}_{\mathcal{P}}^{c,l}(\{p\},y) = \{\{p,q\}\}$. However, we have no way of deriving just p from the program $\mathcal{P}$.*

We can now define the *constructive complete operator* as follows:

Definition 5. *Given an ndao $\mathcal{O}$, the* complete constructive lower bound operator *is defined as:*

$$C^c(\mathcal{O}_l)(y) = \{x \in \mathcal{O}_l(x,y) \mid \exists x_0,..,x \in \mathsf{wfs}(\mathcal{O}_l(.,y))\}$$

The complete constructive upper bound operator is defined analogously, and the constructive stable operator is defined as $S^c(\mathcal{O})(x,y) = C^c(\mathcal{O}_l)(y) \times C^c(\mathcal{O}_u)(x)$. A pair (x,y) is a constructive stable fixpoint *iff $(x,y) \in S^c(\mathcal{O})(x,y)$.*

Notice that for *deterministic* operators over finite lattices, all notions of complete and stable operator coincide (see also [8, Proposition 11]).

We illustrate the constructive stable semantics with the program from Example 6:

Example 8 (Example 6 continued)**.** *Consider again the program from Example 6. We see that (for any $y \subseteq \{p,q\}$), $S^c(\mathcal{O}_l)(y) = \{\{p\},\{q\},\{p,q\}\}$, as $\emptyset,\{p\}$, $\emptyset,\{q\}$ and $\emptyset,\{p,q\}$ are all well-founded sequences relative to $IC^c_{\mathcal{P}}$. Thus, the total constructive stable fixpoints of $IC^c_{\mathcal{P}}$ are $(\{p\},\{p\})$, $(\{q\},\{q\})$ and $(\{p,q\},\{p,q\})$.*

For finite lattices[6], the constructive stable operator is a generalization of the minimality-based stable operator by Heyninck, Arieli and Bogaerts [8] for finite lattices:

Proposition 5. *Let an ndao $\mathcal{O}$ over a finite lattice $L = \langle \mathcal{L}, \leq \rangle$ be given. Then $C^c(\mathcal{O}_\dagger)(y) \supseteq C(\mathcal{O}_\dagger)(y)$ for any $y \in \mathcal{L}$ and $\dagger = l,u$.*

Proof. We first show that (†): for any $\preceq_L^S$-monotonic operator O, for every $x \in \min_{\leq}\{x \in \mathcal{L} \mid x \in O(x)\}$, there is a well-founded sequence $x_0,\ldots,x_n,x$ relative to O. Indeed, since $\bot \leq x$, $O(\bot) \preceq_L^S O(x)$ and thus (since $x \in O(x)$), there is some $x_1 \in O(\bot)$ s.t. $x_1 \leq x$. Repeating this line of reasoning, we obtain a sequence $\bot = x_0 \leq x_1 \leq \ldots \leq x$ s.t. $x_i \in O(x_{i-1})$ for any $i \geq 0$. That there is some $i \geq 0$ s.t. $x \in O(x_i)$ follows from the assumption that we have a finite lattice L. We can now show the proposition. Indeed, since $\mathcal{O}$ is $\preceq_i^A$-monotonic, with [8, Lemma 3], $\mathcal{O}(.,y)$ is $\preceq_L^S$-monotonic. The Proposition follows from †. □

[6]The generalisation of this result to infinite lattices is left for future work.

Thus, what appears to be a change to the semantics of [8] on first sight, can be seen as a mere generalization of these semantics. From the above Proposition 5, it follows immediately that the set of fixpoints of $S(\mathcal{O})$ is a subset of the set of fixpoints of $S^c(\mathcal{O})$:

Corollary 1. *For any ndao $\mathcal{O}$, if $(x,y) \in S(\mathcal{O})(x,y)$ then $(x,y) \in S^c(\mathcal{O})(x,y)$*

Proof. If $(x,y) \in S(\mathcal{O})(x,y)$ then with Proposition 5, $x \in S^c(\mathcal{O}_l)(y)$ and $y \in S^c(\mathcal{O}_u)(x)$ and thus $(x,y) \in S^c(\mathcal{O})(x,y)$. □

The fact that constructive stable operators generalize (minimality-based) stable operators means we can also obtain results on the well-definedness of constructive stable operators on the basis of the insights on (minimality-based) stabel operators [8].

Proposition 6. *For any ndao and $x,y \in \mathcal{L}$ s.t. $\mathcal{O}_l(.,y)$ and $\mathcal{O}_u(x,.)$ are downward closed and $\preceq_L^S$-monotonic, $C^c(\mathcal{O}_l)(y) \neq \emptyset$ and $S^c(\mathcal{O}_y)(x) \neq \emptyset$.*

Proof. Immediate as $C(\mathcal{O}_l)(y) \neq \emptyset$ and $C(\mathcal{O}_y)(x) \neq \emptyset$ in view of [8, Proposition 13] and $C^c(\mathcal{O}_l)(y) \supseteq S_l(\mathcal{O})(y)$ and $C^c(\mathcal{O}_u)(x) \supseteq S_u(\mathcal{O})(x)$ (Proposition 5). □

A property that (unsurprisingly) does *not* generalize from the (minimality-based) stable operator to the constructive stable operator is the $\leq_t$-minimality of stable fixpoints [8, Proposition 14]. This can be seen by observing that in Example 8, $(\{p\},\{p\})$, $(\{q\},\{q\})$ and $(\{p,q\},\{p,q\})$ are stable fixpoints of $\mathcal{IC}^c$, which demonstrates that there might be non-$\leq_t$-minimal fixpoints of $S^c(\mathcal{IC}^{c,l}_{\mathcal{P}})$.

We now show that the total constructive stable fixpoints of $\mathcal{IC}^c_{\mathcal{P}}$ coincide with the stable models according to Liu et al [1].

Proposition 7. *Let a normal choice program $\mathcal{P}$ be given. Then (x,x) is a total constructive stable fixpoint of $\mathcal{IC}^c_{\mathcal{P}}$ iff x is a* LPST*-answer set of $\mathcal{P}$.*

Proof. For the $\Rightarrow$-direction, suppose that $x \in S^c(\mathcal{IC}^c_{\mathcal{P}})(x)$. This means that there is a well-founded sequence $x_0,\ldots,x_n$ relative to $\mathcal{IC}^c_{\mathcal{P}}(.,x)$ s.t. $x_n = x$. We show that $x_0,\ldots,x_n$ is a computation. We first show *convergence*. Indeed, since $(x,x) \in S(\mathcal{IC}^c_{\mathcal{P}})(x,x)$ implies $x \in \mathcal{IC}^c_{\mathcal{P}}(x,x) = IC^c_{\mathcal{P}}(x_\infty)$ this is immediate. The latter follows from the fact that $x_\infty = x$ and the definition of $\mathcal{IC}^c_{\mathcal{P}}$. We now show *persistence of reasons*. We define $\mathcal{P}_i = \{C \leftarrow \phi \in \mathcal{P} \mid (x_i,x)(\phi) = \mathsf{T}, \mathsf{dom}(C) \cap x_{+1} \in \mathsf{sat}(C)\}$. We first show that $\mathcal{P}_i \subseteq \mathcal{P}(x_i)$. Consider some $C \leftarrow \bigwedge_{j=1}^m \alpha_j \wedge$

$\bigwedge \neg_{j'=1}^{m'} \beta_{j'} \in \mathcal{P}_i$. $(x_i, x)(\bigwedge_{j=1}^{m} \alpha_j \wedge \bigwedge \neg_{j'=1}^{m'} \beta_{j'}) = \mathsf{T}$ implies $\alpha_j \in x_i$ for every $j = 1, \ldots, m$ and $\beta_{j'} \notin x$ for every $j' = 1, \ldots, m'$, which implies $\beta_{j'} \notin x_i$ (as $x_i \subseteq x$) for every $j' = 1, \ldots, m'$ and thus $C \leftarrow \bigwedge_{j=1}^{m} \alpha_j \wedge \bigwedge \neg_{j'=1}^{m'} \beta_{j'} \in \mathcal{P}(x_i)$. We now show that $x_{i+1} \in IC^c_{\mathcal{P}_i}(x_i)$. Indeed, consider some $C \leftarrow \bigwedge_{j=1}^{m} \alpha_j \wedge \bigwedge \neg_{j'=1}^{m'} \beta_{j'} \in \mathcal{P}_i$ s.t. $\alpha_j \in x_i$ for $j = 1, \ldots, m$ and $\beta_{j'} \notin x_i$ for $j' = 1, \ldots, m'$. By construction of $\mathcal{P}_i$, $(x_i, x)(\bigwedge_{j=1}^{m} \alpha_j \wedge \bigwedge \neg_{j'=1}^{m'} \beta_{j'}) = \mathsf{T}$ and $\mathsf{dom}(C) \cap x_{+1} \in \mathsf{sat}(C)$. Thus, $x_{i+1} \in IC^c_{\mathcal{P}_i}(x_i)$. We now show that $\mathcal{P}_i \subseteq \mathcal{P}_{i+1}$ for every $i \geq 0$. Consider some $C \leftarrow \bigwedge_{j=1}^{m} \alpha_j \wedge \bigwedge_{j'=1}^{m'} \neg \beta_{j'} \in \mathcal{P}_i$. Clearly, $(x_{i+1}, x)(\bigwedge_{j=1}^{m} \alpha_j \wedge \bigwedge_{j'=1}^{m'} \neg \beta_{j'}) = \mathsf{T}$ as well. What remains to be shown is that $x_{i+1} \cap \mathsf{dom}(C) \in \mathsf{sat}(C)$, which follows from $x_{i+1} \in \mathcal{IC}^c_{\mathcal{P}}(x_i, x)$. *Persistence of beliefs* is immediate. *Revision* follows from the fact that $x_i \in \mathcal{IC}^c_{\mathcal{P}}(x_{i-1}, x)$ for every $i > 0$ and thus (as $\mathcal{P}$ is a normal choice program) also $x_i \in IC^c_{\mathcal{P}}(x_{i-1})$.

For the $\Leftarrow$-direction, suppose that x is a stable model of $\mathcal{P}$ according to [1], i.e. there is a computation $\langle x_i \rangle_{i=0}^{\infty}$ whose result is x. We show there is a well-founded sequence $x_0, z_1, \ldots, x$ relative to $\mathcal{IC}^{c,l}_{\mathcal{P}}(., x)$. With Persistence of reasons, for every x_i, there is some $\mathcal{P}' \subseteq \mathcal{P}$ s.t. $x_i \in \mathcal{IC}^{c,l}_{\mathcal{P}'}(x_{i-1}, x)$. We let $z_i = x_i \cup \bigcup \{x \cap \mathsf{dom}(C) \mid C \leftarrow \phi \in \mathcal{P} \setminus \mathcal{P}', (x_i, x)(\phi) = \mathsf{T}\}$. Notice that $z_i \subseteq x$. We now show: (1) $z_i \subseteq z_{i+1}$ for every $i \geq 0$; and (2) $z_{i+1} \in \mathcal{IC}^{c,l}_{\mathcal{P}}(z_i, x)$ for every $i \geq 0$. For (1): immediate as $x_i \subseteq x_{i+1}$ and $(x_i, x)(\phi) = \mathsf{T}$ implies $(x_{i+1}, x)(\phi) = \mathsf{T}$ for any $\phi \in \mathcal{L}$. For (2): observe first that $x_{i+1} \in \mathcal{IC}^{c,l}_{\mathcal{P}'}(x_i, x)$ for some $\mathcal{P}' \subseteq \mathcal{P}(x)$ according to persistence of reasons. Thus, for every $C \leftarrow \phi \in \mathcal{P}'$, $(x_i, x)(\phi) = \mathsf{T}$ implies $(x, x)(\phi) = \mathsf{T}$, which means also $(z_i, x)(\phi) = \mathsf{T}$. Furthermore, by definition of z_{i+1}, for every $C \leftarrow \phi \in \mathcal{P} \setminus \mathcal{P}'$, C is satisfied by z_{i+1}. Thus, $z_{i+1} \in \mathcal{IC}^{c,l}_{\mathcal{P}}(z_i, x)$. □

6. Disjunctions are Choice Constructs

Our study allows us to give a principled account of the relation between stable semantics for disjunctive logic programs and choice programs. We first show that the operator for disjunctive logic programs is a special case of the operator for choice programs. In more technical detail, for a disjunctive logic program $\mathcal{P}$, we define $\mathtt{D2C}(\mathcal{P}) = \{1 \leq \Delta \leftarrow \phi \mid \bigvee \Delta \leftarrow \phi \in \mathcal{P}\}$. In other words, we replace every disjunction by the constraint that requires at least one element of Δ is true (recall $\mathcal{IC}_{\mathcal{P}}$ is defined in Example 2).

Proposition 8. *For any disjunctive logic program $\mathcal{P}$, $\mathcal{IC}_{\mathcal{P}} = \mathcal{IC}^c_{\mathtt{D2C}(\mathcal{P})}$.*

Proof. It suffices to observe that $x \cap \mathsf{dom}(1 \leq \Delta) \in \mathsf{sat}(1 \leq \Delta)$ iff $\Delta \cap x \neq \emptyset$. □

From this, it immediately follows that all semantics coincide for disjunctive logic programs and their conversion into choice rules. In other words, the operator-based perspective of AFT allows us to point in a very exact way to the difference between disjunctive logic programs and choice programs: the difference is not in how the constructs of disjunction and choice atoms are treated (i.e. when they should be made true or false), but rather in how the stable semantics is constructed: for disjunctions, typically (e.g. in the most popular solvers [16, 17]), the minimality-based stable operator is used, whereas for choice constructs, the constructive stable operator is more apt (in order not to exclude perfectly fine candidate answer sets).

This also gives an answer to the question of how to combine disjunctions and choice constructs in logic programs: one can well combine both constructs, but one has to make a choice as to which stable semantics are used: either one preserves the minimality of answer sets as in disjunctive logic programs and loses some reasonable potential stable models, or one gives up the minimality requirement by using the constructive stable semantics. In this context, it is perhaps interesting to note that the constructive stable semantics still coincides with the standard stable semantics for normal logic programs. In that case, all stable models are minimal.

7. Conclusion, in view of Related Work

In this paper, we studied the semantics of choice programs in the framework of non-deterministic approximation fixpoint theory. One of the main insights was that stable operators based on minimality rule out intuitive acceptable models, which lead us to formulate the *constructive stable semantics*. It was shown that the supported and answer set semantics defined by Liu et al [1] can be represented in our framework, which means we obtain three-valued generalizations of these semantics. An additional benefit of the work done in this paper is the modularity of the AFT-framework: if one is interested in using another approximation operator (several of which have been proposed in the literature on logic programming [18, 18, 19]), one only needs to show the operator is actually an ndao, and the AFT-framework immediately defines a whole

family of semantics. In future work, we plan to look at other such operators, based on existing operators for normal [18], disjunctive [8] or aggregate programs [19]. Another main result in this paper is that we get a principled view on the relation between disjunction and choice programs.

To the best of our knowledge, this is the first application of AFT to the semantics of choice programs. As our semantics basically generalize the semantics of Liu et al [1], the relations between the LPST-answer sets and other semantics for choice programs [2, 3, 4] hold for our framework as well.

The study undertaken in this paper is subject to several restrictions: we assume finite programs, do not allow for choice constructs or aggregates in the body, and assume $\mathcal{IC}^c_{\mathcal{P}}(x, y) \neq \emptyset$ for every consistent interpretation. In future work, we will generalize our results beyond these assumptions.

References

[1] L. Liu, E. Pontelli, T. C. Son, M. Truszczyński, Logic programs with abstract constraint atoms: The role of computations, Artificial Intelligence 174 (2010) 295–315.

[2] V. W. Marek, J. B. Remmel, Set constraints in logic programming, in: Logic Programming and Nonmonotonic Reasoning: 7th International Conference, LPNMR 2004 Fort Lauderdale, FL, USA, January 6-8, 2004 Proceedings 7, Springer, 2004, pp. 167–179.

[3] W. Faber, N. Leone, G. Pfeifer, Recursive aggregates in disjunctive logic programs: Semantics and complexity, in: Proceedings of JELIA'04, volume 3229 of *Lecture Notes in Computer Science*, Springer, 2004, pp. 200–212.

[4] T. C. Son, E. Pontelli, A constructive semantic characterization of aggregates in answer set programming, Theory and Practice of Logic Programming 7 (2007) 355–375.

[5] M. Denecker, V. Marek, M. Truszczyński, Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning, in: Logic-based Artificial Intelligence, volume 597 of *Engineering and Computer Science*, Springer, 2000, pp. 127–144.

[6] M. Fitting, A kripke-kleene semantics for logic programs, The Journal of Logic Programming 2 (1985) 295–312.

[7] M. H. Van Emden, R. A. Kowalski, The semantics of predicate logic as a programming language, Journal of the ACM (JACM) 23 (1976) 733–742.

[8] J. Heyninck, O. Arieli, B. Bogaerts, Nondeterministic approximation fixpoint theory and its application in disjunctive logic programming, arXiv preprint arXiv:2211.17262 (2022).

[9] J. Alcântara, C. V. Damásio, L. M. Pereira, A well-founded semantics with disjunction, in: Proceedings of ICLP'05, Springer, 2005, pp. 341–355.

[10] J. Vennekens, D. Gilis, M. Denecker, Splitting an operator: Algebraic modularity results for logics with fixpoint semantics, ACM Transactions on Computational Logic 7 (2006) 765–797.

[11] M. H. van Emden, R. A. Kowalski, The semantics of predicate logic as a programming language, Journal of the ACM 23 (1976) 733–742.

[12] F. Calimeri, W. Faber, M. Gebser, G. Ianni, R. Kaminski, T. Krennwallner, N. Leone, F. Ricca, T. Schaub, Asp-core-2: Input language format, ASP Standardization Working Group (2012).

[13] M. Gelfond, V. Lifschitz, Classical negation in logic programs and disjunctive databases, New generation computing 9 (1991) 365–385.

[14] S. Brass, J. Dix, Characterizations of the stable semantics by partial evaluation, in: Proceedings of LPNMR'95, Springer, 1995, pp. 85–98.

[15] M. Denecker, J. Vennekens, The well-founded semantics is the principle of inductive definition, revisited, in: Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning, 2014.

[16] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, P. Wanko, Theory solving made easy with clingo 5, in: Technical Communications of the 32nd International Conference on Logic Programming (ICLP 2016), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[17] T. Eiter, M. Fink, T. Krennwallner, C. Redl, Conflict-driven asp solving with external sources, Theory and Practice of Logic Programming 12 (2012) 659–679.

[18] M. Denecker, M. Bruynooghe, J. Vennekens, Approximation fixpoint theory and the semantics of logic and answers set programs, in: Correct reasoning, Springer, 2012, pp. 178–194.

[19] L. Vanbesien, M. Bruynooghe, M. Denecker, Analyzing semantics of aggregate answer set programming using approximation fixpoint theory, Theory and Practice of Logic Programming 22 (2022) 523–537.