Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning
Main Track

Operator-Based Semantics for Choice Programs: Is Choosing Losing?

Jesse Heyninck
Open Universiteit, the Netherlands
University of Cape Town, South-Africa
jesse.heyninck@ou.nl

Abstract

Choice constructs are an important part of the language of logic programming, yet the study of their semantics has been a challenging task. So far, only two-valued semantics have been studied, and the different proposals for such semantics have not been compared in a principled way. In this paper, an operator-based framework allow for the definition and comparison of different semantics in a principled way is proposed.

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 constructs* (Simons, Niemelä, and Soininen 2002), that allow to state e.g. set constraints in the body or head of rules, and are, among others, part of the ASP-Core-2 standard (Calimeri et al. 2012). For example, the rule $1\{p,q,r\}2 \leftarrow s.$ expresses that if s is true, between 1 and 2 atoms among p, q and r are true. Choice constructs are non-deterministic, in the sense that there is more than one way to satisfy them. For example, $1\{p,q,r\}2$ can be satisfied by $\{p\}$, $\{p,q\}$, $\{r\}$, $\ldots$. Formulating semantics for such non-deterministic rules has proven a challenging task (Liu et al. 2010; Marek and Remmel 2004; Faber, Leone, and Pfeifer 2004; Son and Pontelli 2007): several semantics have been proposed but no unifying framework for defining and comparing these semantics exists. Furthermore, attention has been restricted to two-valued semantics, in contradistinction to many other dialects for logic programming for which three- or four-valued semantics have been proposed. Moreover, many proposed semantics only allow for choice constructs in the body, but not in the head. Finally, relations with the non-deterministic construct disjunction remain unclear.

In this paper, a unifying framework for the definition and study of semantics for logic programs with choice constructs in the head and body is provided. This framework is based on immediate consequence operators, which are also useful for the design of explanatory tools and provide foundations for solvers (Kaminski and Schaub 2023; Eiter and Geibinger 2023). The contributions of this paper are the following: (1) we show how the famework of non-deterministic approximation fixpoint theory (AFT) (Heyninck, Arieli, and Bogaerts 2024) can be used to define a wide variety of supported and stable semantics for choice logic programs, (2) we introduce the constructive stable fixpoints, allowing to (3) generalize many existing semantics for choice programs, (4) compare these semantics by introducing postulates, (5) provide a principled comparison with disjunctive logic programs.

Outline of the Paper: In Section 2, the background on choice programs and non-deterministic approximation fixpoint theory is given. In Section 3, approximation operators for choice programs are defined. In Section 4, we study the resulting supported semantics. In Section 5, we define stable semantics and show representation results, while giving a postulate-based study in Section 6. In Section 7, we relate choice constructs and disjunctions. Related work and a conclusion follows in Sections 8 and 9.

Relation with Previous Work: This paper extends this previous work (Heyninck 2023) by also considering choice construct in the body, which gives rise to different possible approximators, and comparing these operators using the notion of groundedness. A full version of this paper, including proofs of all results, is available online (Heyninck 2024).

2 Background and Preliminaries

We recall choice programs and non-deterministic AFT.

2.1 Choice Rules and Programs

A *choice atom* (relative to a set of atoms $\mathcal{A}$) is 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 $1\{p,q,r\}2$ 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). For such choice atoms, we assume the domain and satisfiers are clear and can be left implicit (and similarly for constructs such as $\{p,q\} = 1$ or $\{p,q\} \neq 1$).

A *choice rule* has the form $C \leftarrow C_1, \ldots, C_n$ where $C, C_1, \ldots, C_n$ are choice atoms. C is called the *head* (denoted $\mathsf{hd}(r)$) and $C_1, \ldots, C_n$ the *body*. If C_i is a literal (i.e.

$C_i = (\{\alpha\},\{\{\alpha\}\})$, abbreviated by α, or $C_i = (\{\alpha\},\{\emptyset\})$, abbreviated by $\neg\alpha$) for every $i = 1,\dots,n$, we call it a *normal choice rule*. If C is an atom, we call r an *aggregate rule*. A *choice program* is a set of choice rules, and is called normal[aggregate] if all of the rules are so. A choice atom C is *monotone* if $\mathsf{dom}(C) \cap x \in \mathsf{sat}(C)$ implies $\mathsf{dom}(C) \cap x' \in \mathsf{sat}(C)$ for any $x \subseteq x' \subseteq \mathcal{A}$, and it is *convex* if for any s.t. $x \subseteq y \subseteq \mathcal{A}$, $\mathsf{dom}(C) \cap x \in \mathsf{sat}(C)$ and $\mathsf{dom}(C) \cap y \in \mathsf{sat}(C)$, $\mathsf{dom}(C) \cap z \in \mathsf{sat}(C)$ for any $x \subseteq z \subseteq y$.

Following (Liu et al. 2010), a set $x \subseteq \mathcal{A}$ *satisfies* a choice atom C if $\mathsf{dom}(C) \cap x \in \mathsf{sat}(C)$. An interpretation x satisfies a rule r if x satisfies the head of r or does not satisfy some choice atom 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 $\mathcal{P}(x)$. $x \subseteq \mathcal{A}$ is a *supported model* of $\mathcal{P}$ if it is a model and $x \subseteq \bigcup_{r \in \mathcal{P}(x)} \mathsf{dom}(\mathsf{hd}(r))$. For some $x \subseteq \mathcal{A}$, let $HD_{\mathcal{P}}(x) = \{\mathsf{hd}(r) \mid r \in \mathcal{P}(x)\}$ and $IC_{\mathcal{P}}(x) =$

$$\{z \subseteq \bigcup_{C \in HD_{\mathcal{P}}(x)} \mathsf{dom}(C) \mid \forall C \in HD_{\mathcal{P}}(x) : z(C) = \mathsf{T}\}$$

Example 1. *Consider the program $\mathcal{P} = \{1\{p,q\}2 \leftarrow \{p,q\} \neq 1\}$. The choice atoms behave as follows:*

	$\emptyset$	$\{p\}$	$\{q\}$	$\{p,q\}$
$1\{p,q\}2$	F	T	T	T
$\{p,q\} \neq 1$	T	F	F	T

This means that $IC_{\mathcal{P}}(\emptyset) = IC_{\mathcal{P}}(\{p,q\}) = \{\{p\},\{q\},\{p,q\}\}$, whereas $IC_{\mathcal{P}}(\{p\}) = IC_{\mathcal{P}}(\{q\}) = \{\emptyset\}$. The models of $\mathcal{P}$ are $\{p\}$, $\{q\}$ and $\{p,q\}$, and only $\{p,q\}$ is supported.

2.2 Approximation Fixpoint Theory

We first recall some basic algebraic notions. A *lattice* is a partially ordered set (poset) $\langle \mathcal{L}, \leq \rangle$ s.t. for every $x,y \in \mathcal{L}$, a least upper bound $x \sqcup y$ and a greatest lower bound $x \sqcap y$ exist. A lattice is *complete* if every $X \subseteq \mathcal{L}$ has a least upper bound $\bigsqcup X$ and a greatest lower bound $\bigsqcap X$. $\bigsqcup \mathcal{L}$ is denoted by $\top$ and $\bigsqcap \mathcal{L}$ is denoted by $\bot$. A function $f : X \to Y$ from a poset $\langle X, \leq_1 \rangle$ to a poset $\langle Y, \leq_2 \rangle$ is *monotonic* if $x_1 \leq_1 x_2$ implies $f(x_1) \leq_2 f(x_2)$, and $x \in X$ is a *fixpoint* of f if $x = f(x)$ We define the ordinal powers of a function $f : X \to X$ as $f^0(x) = x$, $f^{\alpha+1}(x) = f(f^{\alpha}(x))$ for a successor ordinal α, and $f^{\alpha}(x) = \bigsqcup_{\beta<\alpha} f^{\beta}(x)$ for a limit ordinal α. The following notation will be used: $[x,y] = \{z \mid x \leq z \leq y\}$. We say a pair (x,y) is consistent if $x \leq y$ and total if $x = y$.

We now recall basic notions from non-deterministic approximation fixpoint theory (AFT) by (Heyninck, Arieli, and Bogaerts 2024), which generalizes approximation fixpoint theory as introduced by (Denecker, Marek, and Truszczyński 2000) to non-deterministic operators. We refer to the original paper (Heyninck, Arieli, and Bogaerts 2024) for more details.

The basic idea behind non-deterministic approximation operators is that we approximate a non-deterministic operator O by generating, for a given lower bound x and upper bound y that approximates z, a set of lower bounds $\{x_1,x_2,\dots\}$ and a set of upper bounds $\{y_1,y_2,\dots\}$ that under- respectively over-approximate $O(z) = \{z_1,z_2,\dots\}$. Formally, a *non-deterministic operator on $\mathcal{L}$* is a function $O : \mathcal{L} \to \wp(\mathcal{L}) \setminus \{\emptyset\}$. For example, the operator $IC_{\mathcal{P}}$ is a non-deterministic operator on the lattice $\langle \wp(\mathcal{A}), \subseteq \rangle$. 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: $X \preceq_L^S Y$ if for every $y \in Y$ there is an $x \in X$ such that $x \leq y$; and $X \preceq_L^H Y$ if for every $x \in X$ there is a $y \in Y$ such that $x \leq y$. Given some $X_1,X_2,Y_1,Y_2 \subseteq \mathcal{L}$, $X_1 \times Y_1 \preceq_i^A X_2 \times Y_2$ iff $X_1 \preceq_L^S X_2$ and $Y_2 \preceq_L^H Y_1$; and $X_1 \times Y_1 \preceq_t^S X_2 \times Y_2$ iff $X_1 \preceq_L^S X_2$ and $Y_2 \preceq_L^S Y_1$. The main orders, instantiated for the lattice of interest for this paper, $\langle \mathcal{A}, \subseteq \rangle$ are recalled in Table 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 $\mathcal{O} : \mathcal{L}^2 \to \wp(\mathcal{L})\backslash\emptyset \times \wp(\mathcal{L})\backslash\emptyset$ is called a *non-deterministic approximating operator* (ndao, for short), if it is $\preceq_i^A$-monotonic (i.e. $(x_1,y_1) \leq_i (x_2,y_2)$ implies $\mathcal{O}(x_1,y_1) \preceq_i^A \mathcal{O}(x_2,y_2)$), and is *exact* (i.e., for every $x \in \mathcal{L}$, $\mathcal{O}(x,x) = (\mathcal{O}_l(x,x), \mathcal{O}_l(x,x))$). (x,y) is a fixpoint of $\mathcal{O}$ if $x \in \mathcal{O}_l(x,y)$ and $y \in \mathcal{O}_u(x,y)$. A non-deterministic operator $O : \mathcal{L} \to \wp(\mathcal{L})$ is *downward closed* if for every sequence $X = \{x_\varepsilon\}_{\varepsilon<\alpha}$ of elements in $\mathcal{L}$ such that: (1) for every $\varepsilon < \alpha$, $O(x_\varepsilon) \preceq_L^S \{x_\varepsilon\}$, and (2) for every $\varepsilon < \varepsilon' < \alpha$, $x_{\varepsilon'} < x_\varepsilon$, it holds that $O(\bigsqcap X) \preceq_L^S \bigsqcap X$. We also introduce the following generalisation of an approximation operator, as it will allow us to capture some further existing semantics.

Definition 1. *An operator $\mathcal{O} : \mathcal{L}^2 \to \wp(\mathcal{L})\backslash\emptyset \times \wp(\mathcal{L})\backslash\emptyset$ is a semi-ndao iff: (1) it is exact, (2) $\mathcal{O}_l(\cdot,y)$ is $\preceq_L^S$-monotonic for any $y \in \mathcal{L}$, and (3) $\mathcal{O}_u(x,\cdot)$ is $\preceq_L^H$-anti-monotonic for any $x \in \mathcal{L}$.*

Any ndao is a semi-ndao (cf. Lemma 1 by (Heyninck, Arieli, and Bogaerts 2024)). A (semi-)ndao $\mathcal{O}$ *approximates* an operator O if $\mathcal{O}_l(x,x) = O(x,x)$ for every $x \in \mathcal{L}$.

We recall now stable operators (given an ndao $\mathcal{O}$). 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* is given by: $S(\mathcal{O})(x,y) = (C(\mathcal{O}_l)(y), C(\mathcal{O}_u)(x))$. (x,y) is a *stable fixpoint* of $\mathcal{O}$ if $(x,y) \in S(\mathcal{O})(x,y)$.

Other semantics (e.g. well-founded state, Kripke-Kleene fixpoints and state and semi-equilibrium semantics) have been defined ((Heyninck, Arieli, and Bogaerts 2024; Heyninck and Bogaerts 2023)) and can be immediately obtained once an ndao is formulated. Due to space limitations, they are not discussed or studied here.

3 Approximation Operators for Choice Programs

The central task is to define non-deterministic approximations of the immediate consequence operator $IC_{\mathcal{P}}$. As is usual, we conceive of pairs of sets of atoms (x,y) as *four-valued interpretations*, where atoms in x are *true* whereas those in y are *not false*. Thus, assuming $x \subseteq y$, (x,y) represents an approximation of some set $z \in [x,y]$. The basic idea behind all the operators defined below is the same: given an input interpretation (x,y), we determine a set of rules that are to be taken into account when constructing the new lower (respectively upper bound), and then take as new lower (respectively upper bounds) the interpretations that make true the heads of all these rules. As is well-known already in the case for aggregate programs, there are various ways to give formal substance to this idea. We will consider four operators inspired by previous work on aggregate or choice programs, namely $\mathcal{IC}^{\mathsf{GZ}}_{\mathcal{P}}$ (Gelfond and Zhang 2014), $\mathcal{IC}^{\mathsf{MR}}_{\mathcal{P}}$ (Marek and Remmel 2004), $\mathcal{IC}^{\mathsf{LPST}}_{\mathcal{P}}$ (Liu et al. 2010), and $\mathcal{IC}^{\mathcal{U}}_{\mathcal{P}}$ (Heyninck and Bogaerts 2023). The study of further operators is left for future work. Intuitively, the GZ-operator takes into account only rules of satisfied bodies whose domain remains unchanged in x and y. The lower bound of the LPST-operator takes into account heads of rules whose body is true in every interpretation between x and y, whereas the upper bound collects the results of applying $IC_{\mathcal{P}}$ to every "non-false" interpretation (i.e. every member of $[x,y]$). The lower bound of the MR-operator looks at all rules whose body is satisfied by the upper bound y and by some subset of the lower bound x, whereas its upper bound is identical to the LPST-operator upper bound. The $\mathcal{U}$-operator, finally, has as a lower and upper bound the LPST-operator upper bound. Given a choice program $\mathcal{P}$ and pair of sets of atoms (x,y), let:

$$\mathcal{HD}^{\mathsf{GZ},l}_{\mathcal{P}}(x,y) = \{C \mid \exists C \leftarrow C_1,\ldots,C_i \in \mathcal{P}, \forall i = 1\ldots n : x \cap \mathsf{dom}(C_i) = y \cap \mathsf{dom}(C_i) \in \mathsf{sat}(C_i)\},$$

$$\mathcal{HD}^{\mathsf{LPST},l}_{\mathcal{P}}(x,y) = \{C \mid \exists C \leftarrow C_1,\ldots,C_n \in \mathcal{P}, \forall i = 1\ldots n : \forall z \in [x,y] : z(C_i) = \mathsf{T}\},$$

$$\mathcal{HD}^{\mathsf{MR},l}_{\mathcal{P}}(x,y) = \{C \mid \exists C \leftarrow C_1,\ldots,C_n \in \mathcal{P}, \exists z \subseteq x : \forall i = 1\ldots n : y(C_i) = \mathsf{T} \text{ and } z(C_i) = \mathsf{T}\},$$

We then define (for $\mathsf{x} \in \{\mathsf{LPST}, \mathsf{MR}, \mathsf{GZ}\}$) the lower bound operators by $\mathcal{IC}^{\mathsf{x},l}(x,y) =$

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

Furthermore, we define (for $\dagger \in \{\mathsf{MR}, \mathsf{LPST}, \mathcal{U}\}$):

$$\mathcal{IC}^{\mathcal{U},l}_{\mathcal{P}}(x,y) \quad = \mathcal{IC}^{\dagger,u}_{\mathcal{P}}(x,y) = \bigcup_{x \subseteq z \subseteq y} IC_{\mathcal{P}}(z)$$

whereas $\mathcal{IC}^{\mathsf{GZ},u}_{\mathcal{P}}(x,y) = \mathcal{IC}^{\mathsf{GZ},l}_{\mathcal{P}}(x,y)$. Finally, we define (for $\mathsf{x} \in \{\mathsf{LPST}, \mathsf{MR}, \mathsf{GZ}, \mathcal{U}\}$):

$$\mathcal{IC}^{\mathsf{x}}(x,y) = (\mathcal{IC}^{\mathsf{x},l}(x,y), \mathcal{IC}^{\mathsf{x},u}(x,y)).$$

We also provide a handy summary of all operators in Table 2.

We first illustrate the behaviour of these operators:

Example 2. *Consider again $\mathcal{P}$ from Example 1. We will consider the three-valued interpretation $(\{p\},\{p,q\})$. We observe that:*

- $\mathcal{IC}^{\mathsf{MR},l}_{\mathcal{P}}(\{p\},\{p,q\}) = \{\{p\},\{q\},\{p,q\}\}$,
- $\mathcal{IC}^{\mathsf{LPST},l}_{\mathcal{P}}(\{p\},\{p,q\}) = \mathcal{IC}^{\mathsf{GZ},\dagger}_{\mathcal{P}}(\{p\},\{p,q\}) = \{\emptyset\}$ *(for $\dagger = l,u$), and*
- $\mathcal{IC}^{\mathcal{U},l}_{\mathcal{P}}(\{p\},\{p,q\}) = \mathcal{IC}^{\mathcal{U},u}_{\mathcal{P}}(\{p\},\{p,q\} = \{\emptyset,\{p\},\{q\},\{p,q\}\}$.

Thus, $(\{p\},\{p,q\})$ is a fixpoint of the MR- *and* $\mathcal{U}$*-operators, but not of the* LPST *and* GZ*-operators.*

Unsurprisingly, these ndaos are not well-defined for every program (inevitably so, as this already holds for $IC_{\mathcal{P}}$):

Example 3. *Let $\mathcal{P} = \{\{p,q\} \neq 2 \leftarrow; \{p,q\} = 2 \leftarrow\}$. For this program, no set satisfying both heads exists, and thus $\mathcal{IC}^{\mathsf{x}}_{\mathcal{P}}(x,y)$ is not defined (for any $x,y \subseteq \mathcal{A}$).*

We therefore making the following

Assumption: We restrict attention to programs for which $\mathcal{IC}^{\mathsf{x},\dagger}_{\mathcal{P}}(x,y) \neq \emptyset$ for any $x,y \subseteq \mathcal{A}$, $\mathsf{x} \in \{\mathsf{LPST}, \mathsf{MR}, \mathsf{GZ}, \mathcal{U}\}$ and $\dagger \in \{l,u\}$.

All operators are (semi-)approximators of $IC_{\mathcal{P}}$:[1]

Proposition 1. *$\mathcal{IC}^{\mathsf{LPST}}_{\mathcal{P}}$ and $\mathcal{IC}^{\mathsf{MR}}_{\mathcal{P}}$ are ndaos for $IC_{\mathcal{P}}$. $\mathcal{IC}^{\mathsf{MR}}_{\mathcal{P}}$ is a semi-ndao for $IC_{\mathcal{P}}$. $\mathcal{IC}^{\mathsf{GZ}}_{\mathcal{P}}$ is an ndao for $IC_{\mathcal{P}}$ when restricted to consistent inputs.*

The following example shows that $\preceq^{A}_{i}$-monotonicity of $\mathcal{IC}^{\mathsf{GZ}}_{\mathcal{P}}$ is not guaranteed when considering inconsistent interpretations:

Example 4. *Consider $\mathcal{P} = \{p \leftarrow \{p,q\} \neq 0\}$. We see that*

$$(\{p\},\{p\}) \leq_i (\{p,q\},\{p\}) \quad \textit{yet}$$

$$\mathcal{IC}^{\mathsf{GZ},l}_{\mathcal{P}}(\{p\},\{p\}) = \{\{p\}\} \not\preceq^{S}_{L} \mathcal{IC}^{\mathsf{GZ},l}_{\mathcal{P}}(\{p,q\},\{p\}) = \{\emptyset\}$$

To see that $\mathcal{IC}^{\mathsf{GZ},l}_{\mathcal{P}}(\{p,q\},\{p\}) = \{\emptyset\}$, observe that $\mathsf{dom}(\{p,r\} \neq 0) \cap \{p\} \neq \mathsf{dom}(\{p,r\} \neq 0) \cap \{p,q\}$.

Furthermore, the LPST and MR-operators coincide for normal choice programs:

Proposition 2. *For any normal choice program, $\mathcal{IC}^{\mathsf{MR}}_{\mathcal{P}}(x,y) = \mathcal{IC}^{\mathsf{LPST}}_{\mathcal{P}}(x,y)$.*

Remark 1. *It can be easily verified that the* LPST *and* MR-*operators coincide with the four-valued Przymusinski operator for normal logic programs (Przymusinski 1990), which implies, with Proposition 3, Proposition 11 and Theorem 6 by (Heyninck, Arieli, and Bogaerts 2024), that the semantics based on these operators faithfully generalize the (partial) supported, (partial) stable and well-founded semantics of normal logic programs.*

4 Supported Model Semantics

In this section, we look at the fixpoints of $\mathcal{IC}^{\mathsf{x}}_{\mathcal{P}}$, which give a three-valued generalisation of the supported model semantics by (Liu et al. 2010) (and the three-valued model semantics of normal logic programs). Intuitively, fixpoints of $\mathcal{IC}^{\mathsf{x}}_{\mathcal{P}}$ generalise the idea that only atoms supported by activated rules can be accepted.

A first insight is that the total fixpoints of all operators coincide with *supported models* (Liu et al. 2010).

[1]Proof of all results in the paper, as well as additional results can be foud in the full version of this paper (Heyninck 2024).

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

Immediate Consequence Operator for choice programs	
$\mathcal{P}(x) =$	$\{C \leftarrow C_1, \ldots, C_i \in \mathcal{P} \mid \forall i = 1 \ldots n : x \cap \mathsf{dom}(C_i) \in \mathsf{sat}(C_i)\}$
$HD_{\mathcal{P}}(x) =$	$\{\mathsf{hd}(r) \mid r \in \mathcal{P}(x)\}$
$IC_{\mathcal{P}} =$	$\{z \subseteq \bigcup_{C \in HD_{\mathcal{P}}(x)} \mathsf{dom}(C) \mid \forall C \in HD_{\mathcal{P}}(x) : z(\mathsf{hd}(r)) = \mathsf{T}\}$
Ndao based on (Gelfond and Zhang 2014)	
$\mathcal{HD}_{\mathcal{P}}^{\mathsf{GZ},l}(x,y) =$	$\{C \mid \exists C \leftarrow C_1, \ldots, C_i \in \mathcal{P}, \forall i = 1 \ldots n : x \cap \mathsf{dom}(C_i) = y \cap \mathsf{dom}(C_i) \in \mathsf{sat}(C_i)\}$
$\mathcal{IC}^{\mathsf{GZ},l}(x,y) =$	$\{z \subseteq \bigcup_{C \in \mathcal{HD}_{\mathcal{P}}^{\mathsf{GZ},l}(x,y)} \mathsf{dom}(C) \mid \forall C \in \mathcal{HD}_{\mathcal{P}}^{\mathsf{GZ},l}(x,y) : z \cap \mathsf{dom}(C) \in \mathsf{sat}(C)\}$
$\mathcal{IC}^{\mathsf{GZ},u}(x,y) =$	$\mathcal{IC}^{\mathsf{GZ},l}(x,y)$
$\mathcal{IC}^{\mathsf{GZ}}(x,y) =$	$(\mathcal{IC}^{\mathsf{GZ},l}(x,y), \mathcal{IC}^{\mathsf{GZ},u}(x,y))$
Ultimate Ndao for choice programs	
$\mathcal{IC}_{\mathcal{P}}^{\mathcal{U},l}(x,y) =$	$\bigcup_{x \subseteq z \subseteq z} IC_{\mathcal{P}}(z)$
$\mathcal{IC}^{\mathcal{U}}(x,y) =$	$(\mathcal{IC}^{\mathcal{U},l}(x,y), \mathcal{IC}^{\mathcal{U},l}(x,y))$
Ndao based on (Liu et al. 2010)	
$\mathcal{HD}_{\mathcal{P}}^{\mathsf{LPST},l}(x,y) =$	$\{C \mid \exists C \leftarrow C_1, \ldots, C_n \in \mathcal{P}, i = 1 \ldots n : \forall z \in [x,y] : z(C_i) = \mathsf{T}\}$,
$\mathcal{IC}^{\mathsf{LPST},l}(x,y) =$	$\{z \subseteq \bigcup_{C \in \mathcal{HD}_{\mathcal{P}}^{\mathsf{LPST},l}(x,y)} \mathsf{dom}(C) \mid \forall C \in \mathcal{HD}_{\mathcal{P}}^{\mathsf{LPST},l}(x,y) : z \cap \mathsf{dom}(C) \in \mathsf{sat}(C)\}$
$\mathcal{IC}^{\mathsf{LPST},u}(x,y) =$	$\mathcal{IC}^{\mathcal{U},l}(x,y)$
$\mathcal{IC}^{\mathsf{LPST}}(x,y) =$	$(\mathcal{IC}^{\mathsf{LPST},l}(x,y), \mathcal{IC}^{\mathsf{LPST},u}(x,y))$
Ndao based on (Marek and Remmel 2004)	
$\mathcal{HD}_{\mathcal{P}}^{\mathsf{MR},l}(x,y) =$	$\{C \mid \exists C \leftarrow C_1, \ldots, C_n \in \mathcal{P}, \exists z \subseteq x : \forall i = 1 \ldots n : y(C_i) = \mathsf{T}$ and $z(C_i) = \mathsf{T}\}$
$\mathcal{IC}^{\mathsf{MR},l}(x,y) =$	$\{z \subseteq \bigcup_{C \in \mathcal{HD}_{\mathcal{P}}^{\mathsf{MR},l}(x,y)} \mathsf{dom}(C) \mid \forall C \in \mathcal{HD}_{\mathcal{P}}^{\mathsf{MR},l}(x,y) : z \cap \mathsf{dom}(C) \in \mathsf{sat}(C)\}$
$\mathcal{IC}^{\mathsf{MR},u}(x,y) =$	$\mathcal{IC}^{\mathcal{U},l}(x,y)$
$\mathcal{IC}^{\mathsf{MR}}(x,y) =$	$(\mathcal{IC}^{\mathsf{MR},l}(x,y), \mathcal{IC}^{\mathsf{MR},u}(x,y))$

Table 2: Concrete Operators for dlps and choice programs

Proposition 3. *Let a choice program $\mathcal{P}$ and $\mathsf{x} \in \{\mathsf{LPST}, \mathsf{MR}, \mathsf{GZ}, \mathcal{U}\}$ be given. Then (x,x) is a fixpoint of $\mathcal{IC}^{\mathsf{x}}_{\mathcal{P}}$ iff x is a supported model of $\mathcal{P}$.*

However, our extension to three-valued semantics allows to give semantics to a wider class of programs:

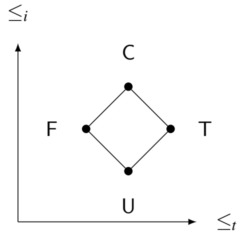
Example 5. *Let $\mathcal{P} = \{\{p,q\} = 1 \leftarrow \{p\} \neq 1; p \leftarrow q.\}$. It can be easily checked that this program has no two-valued supported model. However, $(\emptyset, \{p\})$ is a fixpoint of $\mathcal{IC}^{\mathsf{x}}_{\mathcal{P}}$ for $x \in \{\mathsf{LPST}, \mathsf{MR}, \mathcal{U}\}$.*

A more detailed investigation of conditions for the existence of fixpoints is left for future work.

We now show that fixpoints of all four operators satisfy a notion of *supportedness*, relative to how rule bodies are evaluated according to their definition. For example, C can be said to be true in (x,y) according to the LPST-operator if $z(C) = \mathsf{T}$ for every $z \in [x,y]$. We formalize this as follows. Given (x,y) and $\mathcal{IC}^{\mathsf{x}}_{\mathcal{P}}$, $a \in x$ supported if a occurs in the domain of the head of a rule whose body is true, i.e. there is some $C \leftarrow C_1, \ldots, C_n$ s.t. $a \in \mathsf{dom}(C)$, and if would replace C by the dummy atom p, p is the only consequence of $\mathcal{IC}^{\mathsf{x},l}_{\{p \leftarrow C_1,\ldots,C_n\}}(x,y)$ (and similarly for y).

Proposition 4. *Let a choice program $\mathcal{P}$ and $\mathsf{x} \in \{\mathsf{LPST}, \mathsf{MR}, \mathsf{GZ}, \mathcal{U}\}$ be given. Then $(x,y) \in \mathcal{IC}^{\mathsf{x}}_{\mathcal{P}}(x,y)$ implies that for every $a \in y$, there is some $C \leftarrow C_1, \ldots, C_n$ with $a \in \mathsf{dom}(C)$ s.t. $\mathcal{IC}^{\mathsf{x}}_{\{p \leftarrow C_1,\ldots,C_n\}}(x,y) = (\{\{p\} \cap x\}, \{\{p\} \cap y\})$.*

Normal choice programs We can simplify the definition of supported models for normal choice programs. 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. We first recall the bilattice FOUR, consisting of the elements T (true), F (false), U (undecided) and C (contradictory) and two order relations $\leq_i$ and $\leq_t$:



Definition 2. *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* $x \cap \mathsf{dom}(c) \in \mathsf{sat}(C)$ *and* $y \cap \mathsf{dom}(c) \notin \mathsf{sat}(C)$,
- $(x,y)(C) = \mathsf{U}$ *if* $x \cap \mathsf{dom}(c) \notin \mathsf{sat}(C)$ *and* $y \cap \mathsf{dom}(c) \in \mathsf{sat}(C)$.

We define *three-valued models* of a normal choice program $\mathcal{P}$ as consistent interpretations (x,y) for which $\bigsqcap_{\leq_t}\{(x,y)(C_i) \mid i = 1 \ldots n\} \leq_t (x,y)(C)$ for every $C \leftarrow C_1, \ldots, C_n \in \mathcal{P}$, and *supported*[2] models as models (x,y) of $\mathcal{P}$ s.t. for every $p \in y$, there is a rule $C \leftarrow C_1, \ldots, C_n \in \mathcal{P}$ with $p \in \mathsf{dom}(C)$ and $\bigsqcap_{\leq_t}\{(x,y)(C_i) \mid i = 1 \ldots n\} \geq_t (x,y)(p)$. I.e., a model is supported if for every non-false atom p, we have a reason in the form of an activated rule for accepting (or not rejecting) that atom.

Three-valued supported models of $\mathcal{P}$ coincide with fixpoints of $\mathcal{IC}^{\mathsf{LPST}}_{\mathcal{P}}$ and $\mathcal{IC}^{\mathsf{MR}}_{\mathcal{P}}$, whereas those of $\mathcal{IC}^{\mathsf{GZ}}_{\mathcal{P}}$ are a subset of the three-valued supported models:

Proposition 5. *Let some normal choice program $\mathcal{P}$ and $\mathsf{x} \in \{\mathsf{LPST}, \mathsf{MR}\}$ be given. Then (x,y) is a three-valued supported model of $\mathcal{P}$ iff (x,y) is a fixpoint of $\mathcal{IC}^{\mathsf{x}}_{\mathcal{P}}$. Furthermore, if (x,y) is a fixpoint of $\mathcal{IC}^{\mathsf{GZ}}_{\mathcal{P}}$ then (x,y) is supported, but not always vice-versa. Supported models might not be fixpoints of $\mathcal{IC}^{\mathcal{U}}_{\mathcal{P}}$ and vice-versa.*

Models as pre-fixpoints It is well-known that for many non-monotonic formalisms, pre-fixpoints of an operator can characterise models of the corresponding knowledge base. For the general case of choice constructs, this correspondence does not hold:

Example 6. *Consider $\mathcal{P} = \{\{p,q\} = 1 \leftarrow\}$. Then $(\{p,q\}, \{p,q\})$ is a pre-fixpoint of $\mathcal{IC}_{\mathcal{P}}$ (as $\mathcal{IC}_{\mathcal{P}}(\{p,q\}, \{p,q\}) = \{\{p\}, \{q\}\} \times \{\{p\}, \{q\}\} \preceq^S_t (\{p,q\}, \{p,q\})$) yet it is not a model (as the only models are $\{p\}$ and $\{q\}$): there is no way to prove p and q.*

For choice programs with monotone heads, this correspondence *does* hold:

Proposition 6. *Let some choice program $\mathcal{P}$ s.t. for every $C \leftarrow C_1, \ldots, C_n$, C is monotone and some $\mathsf{x} \in \{\mathsf{LPST}, \mathsf{MR}, \mathsf{GZ}, \mathcal{U}\}$ be given. Then x is a model of $\mathcal{P}$ iff $\mathcal{IC}^{l,\mathsf{x}}_{\mathcal{P}}(x,x) \preceq^S_L x$.*

5 Stable Semantics

We now move to the stable semantics, whose main aim is favvoding the acceptance of self-supporting cycles, e.g. accepting $(\{p\}, \{p\})$ as a reasonable model of the program $\{p \leftarrow p\}$. For the deterministic case, this is done by looking at fixpoints of the *stable operator*, obtained by calculating a new lower bound as the least fixpoint of $\mathcal{O}_l(\cdot, y)$, where y is the input upper bound (and similarly for the upper bound). Intuitively, we take the least information the upper bound obliges use to derive. In contradistinction to the deterministic case, there are divergent options for how to define stable semantics for non-deterministic operators. We first consider the minimality-based stable semantics known from non-deterministic AFT (Heyninck, Arieli, and Bogaerts 2024), defined as the $\preceq$-minimal fixpoints of

[2]These models have been called *weakly supported models* (Brass and Dix 1995) for disjunctive logic programs.

$\mathcal{O}_l(.,y)$ (cf. Section 2.2), which where shown to generalize the (partial) stable model semantics for disjunctive (aggregate) programs (Heyninck, Arieli, and Bogaerts 2024; Heyninck and Bogaerts 2023). For choice constructs, this construction is overly strong:

Example 7. *Consider the program $\mathcal{P} = \{1\{p,q\}2 \leftarrow\}$. Intuitively, this rule allows to choose between one and two among p and q. The stable version of $\mathcal{IC}^{\mathsf{x}}_{\mathcal{P}}$ (cf. Section 2.1) behaves as follows (for any* $\mathsf{x} = \mathsf{LPST}, \mathsf{MR}, \mathsf{GZ}$*):*

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

$\{p\}$ and $\{q\}$ are the two stable fixpoints of $\mathcal{IC}^{\mathsf{x}}_{\mathcal{P}}$. This is undesirable, as according to the intuitive reading of $\mathcal{P}$, $\{p,q\}$ should also be allowed as a stable interpretation.

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 (Heyninck, Arieli, and Bogaerts 2024) as follows. For deterministic operators, the stable version of an approximation operator $\mathcal{O}$ is defined as the greatest lower bound (glb) of fixpoints of $\mathcal{O}_l(.,y)$. For deterministic operators over finite lattices, the minimal fixpoint of $\mathcal{O}_l(.,y)$ are 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. $\bigcup_{i=1}^{\infty} \mathcal{O}_l^i(\bot,y)$). For non-deterministic operators, this correspondence does not hold. Indeed, the glb of fixpoints of $\mathcal{O}_l(.,x)$ is often too weak (e.g. for the program $\mathcal{P}$ from Example 7 we get $\{p\} \cap \{q\} \cap \{p,q\} = \emptyset$ as the glb of fixpoints). However, this still leaves a third choice: namely looking at fixpoints reachable by applications of $\mathcal{O}_l(.,y)$ starting from $\bot$. E.g. for choice programs, we are interested in the fixpoints of $\mathcal{IC}^{\mathsf{x},l}_{\mathcal{P}}(.,y)$ that can built them up from the ground up (i.e. from $\emptyset$) by a sequence of applications of $\mathcal{IC}^{\mathsf{x},l}_{\mathcal{P}}(.,y)$. We first generalize the notion of a well-founded sequence by (Denecker and Vennekens 2014):

Definition 3. *Given a non-deterministic operator $O : \mathcal{L} \to \wp(\mathcal{L})$ over a complete lattice, a sequence $x_0,\ldots,x_n \subseteq \mathcal{L}$ is* well-founded relative to O *if: (1) $x_0 = \bot$; (2) $x_i \leq x_{i+1}$ and $x_{i+1} \in O(x_i)$ for every successor ordinal $i \geq 0$; and (3) $x_\lambda = (\bigsqcap\{x_i\}_{i<\lambda})$ for a limit ordinal λ. The well-founded sequences relative to O are denoted by* $\mathsf{wfs}(O)$.

Remark 2. *The assumption of a complete lattice in Definition 3 is needed since the greatest lower bound is used in point 3 of Definition 3.*

Notice that, in contradistinction to the deterministic version of a well-founded sequence (Denecker and Vennekens 2014), 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 $x_{i+1} \preceq^S_L O(x_i)$:

Example 8. *Let $\mathcal{P} = \{\{p,q\} = 2 \leftarrow\}$. If we would allow for $x_{i+1} \preceq^S_L O(x_i)$ in Definition 3.(2), $\emptyset,\{p\}$ would be a well-founded sequence according to $\mathcal{IC}^l_{\mathcal{P}}(\cdot,y)$ (for any $y \subseteq \{p,q\}$) as $\{p\} \preceq^S \mathcal{IC}^l_{\mathcal{P}}(\{p\},y) = \{\{p,q\}\}$. However, we have no way of deriving just p from the program $\mathcal{P}$.*

We now define the *constructive stable operator*:

Definition 4. *Given an semi-ndao $\mathcal{O}$ over a complete lattice $\mathcal{L}$ with $y \in \mathcal{L}$, the* c(onstructive)-complete 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 c-complete upper bound operator *$C^c(\mathcal{O}_u)$ is defined analogously, and the* c-stable operator *is defined as $S^c(\mathcal{O})(x,y) = (C^c(\mathcal{O}_l)(y), C^c(\mathcal{O}_u)(x))$. (x,y) is a* c-stable fixpoint *iff $(x,y) \in S^c(\mathcal{O})(x,y)$.*

Example 9 (Example 7 continued)**.** *Consider again the program from Example 7. We see that (for any $y \subseteq \{p,q\}$), $S^c(\mathcal{IC}^{\mathsf{x},l}_{\mathcal{P}})(y) = \{\{p\},\{q\},\{p,q\}\}$, as $\emptyset,\{p\}$, $\emptyset,\{q\}$ and $\emptyset,\{p,q\}$ are all well-founded sequences. Thus, the total c-stable fixpoints of $IC^{\mathsf{x}}_{\mathcal{P}}$ are $(\{p\},\{p\})$, $(\{q\},\{q\})$ and $(\{p,q\},\{p,q\})$ (for $\mathsf{x} = \mathsf{LPST}, \mathsf{MR}, \mathsf{GZ}, \mathcal{U}$).*

It is easy to see that for a deterministic operator over a complete lattice, the c-stable operator coincides with the stable operator known from deterministic AFT. The c-stable operator generalizes the minimality-based one:

Proposition 7. *Let an (semi-)ndao $\mathcal{O}$ over a complete lattice $L = \langle \mathcal{L}, \leq \rangle$ be given s.t. $\mathcal{O}_u(x,\cdot)$ is $\preceq^S_L$-monotonic for any $x \in \mathcal{L}$. Then $C^c(\mathcal{O}_\dagger)(y) \supseteq C(\mathcal{O}_\dagger)(y)$ for any $y \in \mathcal{L}$ and $\dagger = l,u$. Furthermore, if $(x,y) \in S(\mathcal{O})(x,y)$ then $(x,y) \in S^c(\mathcal{O})(x,y)$.*

Thus, what might appear to be a change to the previously formulated stable semantics for non-deterministic approximation operators (Heyninck, Arieli, and Bogaerts 2024), is a mere generalization of these semantics.

In general, non-deterministic operators might not admit a fixpoint, which means that the c-complete operator is not always well-defined (e.g. $C^c(O_l)(x) = \emptyset$ or $C^c(O_u)(y) = \emptyset$). We illustrate this for $C^c(O_u)(y)$, by giving an example of a $\preceq^H_L$-monotonic operator that does not admit a well-founded sequence.

Example 10. *Let $O : \mathbb{N} \cup \{\infty\}$ where $O(i) = \{i+1\}$ for any $i \in \mathbb{N}$ and $\infty = \mathbb{N}$. Observe that O is $\preceq^H_L$-monotonic, as for any finite i,j s.t. $i \leq j$, $O(i) = \{i+1\}$ and $i+1 \leq j+1 \in O(j)$, and for any finite i, $i \in O(\infty)$. However, it is clear that O admits no fixpoint and thus no well-founded sequence.*

To ensure well-definedness of the c-stable operator we will assume that the lower bound is downwards closed and the upper bound satisfies the following, analogous, notion, called *upwards closedness*:

Definition 5. *An operator O is* upwards closed *if for every sequence $X = \{x_\varepsilon\}_{\varepsilon<\alpha}$ of elements in $\mathcal{L}$ s.t.*

1. *for every $\varepsilon < \alpha$, $x_\varepsilon \preceq^H_L O(x_\varepsilon)$, and*
2. *for every $\varepsilon' < \varepsilon < \alpha$, $x_{\varepsilon'} < x_\varepsilon$,*

it holds that $\bigsqcup X \preceq^H_L O(\bigsqcup X)$.

It can be easily observed that the operator from Example 10 is not upwards closed. Downwards closedness of $\mathcal{O}_l(\cdot,y)$ and upwards closedness of $\mathcal{O}_l(x,\cdot)$ guarantee well-definedness of the c-stable operator:

Proposition 8. *For any (semi-)ndao $\mathcal{O}$ over a complete lattice s.t. $\mathcal{O}_l(.,y)$ is downwards closed and $\mathcal{O}_u(x,.)$ is upwards closed for any $x,y \in \mathcal{L}$, $C^c(\mathcal{O}_l)(y) \neq \emptyset$ and $C^c(\mathcal{O}_y)(x) \neq \emptyset$.*

We now show that the c-stable operator is well-defined for all of the ndao's considered in this paper for choice programs $\mathcal{P}$ built up from choice atoms whose domain is finite. This is a sufficient condition: other conditions might also guarantee well-definedness. For most applications, this assumption seems warranted.

Proposition 9. *Let a choice program $\mathcal{P}$ s.t. for every $C_1 \leftarrow C_2,\ldots,C_n \in \mathcal{P}$, $\mathsf{dom}(C_i)$ is finite for $i = 1\ldots n$, and $\mathsf{x} \in \{\mathsf{MR},\mathsf{LPST},\mathcal{U}\}$ and $x \subseteq y \subseteq \mathcal{A}$ be given. Then $S^c(\mathcal{IC}^{\mathsf{x}}_{\mathcal{P}})(x,y) \neq \emptyset$. Furthermore, $C^c(\mathcal{IC}^{\mathsf{GZ},l}_{\mathcal{P}})(y) \neq \emptyset$.*[3]

Upwards closedness is not guaranteed when allowing choice constructs with infinite domains:

Example 11. *Consider the set of atoms $x = \{a_i \mid i \in \mathbb{N}\}$ and the choice atom $C = (A, \{A' \subseteq A \mid A' \text{ is infinite}\})$. We let $\mathcal{P} = \{C \leftarrow\}$. Let $x_j = A \setminus \{a_j \mid j \leq i\}$ for any $i \in \mathbb{N}$. Then $\{x_j\}_{j<\infty}$ is a $\subseteq$-descending chain of infinite sets. We see that for any $j < \infty$, $IC_{\mathcal{P}}(x_j) = \{A' \subseteq A \mid A' \text{ is infinite}\} \preceq^S_L \{x_j\}$ as $x_j \in \{A' \subseteq A \mid A' \text{ is infinite}\}$. However, $\bigcap_{j<\infty}\{x_j\} = \emptyset$, whereas $IC_{\mathcal{P}}(\emptyset) = \{A' \subseteq A \mid A' \text{ is infinite}\} \preceq^S_L \{x_j\}$ and thus $IC_{\mathcal{P}}(\emptyset) \not\preceq^S_L \{\emptyset\}$.*

Remark 3. *Another property that stable fixpoints of both deterministic and non-deterministic operators have in common is that they are $\leq_t$-minimal. As is to be expected and as can be seen from the constructive stable fixpoint $(\{p,q\},\{p,q\})$ in Example 9, constructive stable fixpoints are not necessariy $\leq_t$-minimal, in contradistinction to minimiality-based stable fixpoints (Proposition 14 by (Heyninck, Arieli, and Bogaerts 2024)).*

We now show a host of representation results: the LPST-operator allows to generalize the semantics of Liu et al (Liu et al. 2010) from two- to three-valued. The GZ-operator allows to adapt the semantics of (Gelfond and Zhang 2014) (defined for disjunctive programs) to choice programs, and the semantics of (Marek and Remmel 2004) from two- to three-valued. Due to spatial limitations, we cannot recall the definitions of the represented semantics (but give them in the full paper (Heyninck 2024)):

Proposition 10. *Let a choice program $\mathcal{P}$ be given.*

1. *x is a stable model according to (Liu et al. 2010) iff (x,x) is a stable fixpoint of $\mathcal{IC}^{\mathsf{LPST}}_{\mathcal{P}}$.*
2. *x is a stable model according to (Marek and Remmel 2004) iff (x,x) is a stable fixpoint of $\mathcal{IC}^{\mathsf{MR}}_{\mathcal{P}}$.*
3. *If $\mathcal{P}$ is a aggregate program then x is a stable model according to (Gelfond and Zhang 2014) iff $x \in C^c(\mathcal{IC}^{\mathsf{GZ},l}_{\mathcal{P}})(x)$.*

[3]Notice that, as $\mathcal{IC}^{\mathsf{GZ},u}_{\mathcal{P}}$ is only $\preceq^H_L$-monotonic for consistent inputs, the complete operator for the upper-bound will not be well-defined as it starts from $\mathcal{IC}^{\mathsf{GZ},u}_{\mathcal{P}}(x,\emptyset)$. This means that the GZ-operator is only useful for total stable fixpoints.

Furthermore, for normal logic programs, all operators besides the ultimate coincide, and the stable fixpoints respectively partial stable fixpoint coincide with the stable respectively partial stable models (and similarly for the well-founded semantics) (recall Remark 1). Thus, the semantics studied in this paper do not only generalize existing semantics for choice or aggregate programs, but also the well-known semantics for normal logic programs.

6 Groundedness

We introduce several postulates to facilitate a comparison between semantics for choice programs (thus solving an open question in the literature (Alviano, Faber, and Gebser 2023)) formalizing in different ways the idea of *groundedness*. Furthermore, we show that for every notion of groundedness, there exist examples which have been argued in the literature to be counter-intuitive.

Intuitively, the idea behind groundedness is that models should be derivable *from the ground up*, i.e. they should be supported by non-cyclic arguments. For choice programs, what constitutes a cycle becomes less clear:

Example 12 ((Liu et al. 2010)). *Consider the program $\mathcal{P} = \{\{p,q\} = 2 \leftarrow \{p,q\} \neq 1\}$. There are two candidates for stable models: $\emptyset$ and $\{p,q\}$ (as the only satisfier of the head of the only rule is $\{p,q\}$), which we discuss:*
(1) If we choose $\emptyset$, then we see that the only rule in the program is applicable but not applied.
(2) If we choose $\{p,q\}$, we could justify this intuitively by the sequence $\emptyset,\{p,q\}$: at the first step, $\emptyset$ makes $\{p;q\} \neq 1$ true and thus we derive $\{p,q\}$. At the second step, however, we can only justify our choice by a self-supporting justification: $\{p,q\}$ is true since it is derivable using the head of the rule $\{p,q\} = 2 \leftarrow \{p;q\} \neq 1$ and since $\{p,q\}(\{p;q\} \neq 1) = \mathsf{T}$.
We observe that $\{p,q\}$ is stable fixpoint for the $\mathcal{U}$- and MR-operator, but not for the LPST- and GZ-operators.

The benefit of our operator-based framework is that we do not have to choose for a single "best" semantics: the choices outlined above will be obtainable by using different operators. We now carry forward our study of self-support on a more general level by introducing three postulates of decreasing strength. The first notion (inspired by (Gelfond and Zhang 2014)) requires that we can find a stratification of the program (i.e. an assignment κ of natural numbers representing levels) s.t. the entire domain of the head of a rule is strictly higher stratified then the domain of the choice constructs in the body.

Definition 6. *A set x is* d(omain)-grounded *(for $\mathcal{P}$) if there is some $\kappa : x \to \mathbb{N}$ s.t. every $a \in x$ there is some $r = C \leftarrow C_1,\ldots,C_n \in \mathcal{P}$ s.t. $a \in \mathsf{dom}(C)$, $\kappa(a) > \max\{\kappa(b) \mid b \in \bigcup_{i=1}^n \mathsf{dom}(C_i)\}$.*

As already observed by (Alviano, Faber, and Gebser 2023), this requirement might be overly strong:

Example 13. *Consider $\mathcal{P} = \{b \leftarrow 1\{a,b\}; a \leftarrow\}$. Then one might expect $\{a,b\}$ to be a good candidate for a stable model. Indeed, a is a fact and allows to support b. However, $\{a,b\}$ is not d-grounded, as there is no $\kappa : \{a,b\} \to \mathbb{N}$ s.t. $\kappa(b) > \max\{\kappa(a),\kappa(b)\}$.*

The following weaker notion (inspired in name and idea by (Liu et al. 2010)) requires that at some point in a sequence, there is support for the body of a rule, and this support persists in every following step:

Definition 7. *(1) A set x is an y-*trigger *for $C \leftarrow C_1,\ldots,C_n$ if for every $z \in [x,y]$, $z(C_i) = \mathsf{T}$ for every $i = 1,\ldots,n$. (2) A set x is* s(trongly)-grounded *(for $\mathcal{P}$) if there is some $\kappa : x \to \mathbb{N}$ s.t. every $a \in x$, there is some $r = C \leftarrow C_1,\ldots,C_n \in \mathcal{P}$ s.t. $a \in \mathsf{dom}(C)$ and there is an x-trigger z for r s.t. $\kappa(a) > max\{\kappa(b) \mid b \in z\}$.*

Intuitively, x is s-grounded if for every atom $a \in x$, we can find a rule that that has an x-trigger in a strictly lower level.

D-grounded sets are s-grounded, but not vice-versa:

Example 14. *Consider $\mathcal{P}$ as in Example 13. We see that $\{a,b\}$ is strongly grounded. Indeed, if we take $\kappa(a) = 0$ and $\kappa(b) = 1$, we see that $b \leftarrow 1\{a,b\}$ has a κ-lower $\{a,b\}$-trigger in $\{a\}$ as $\{a\}(1\{a,b\}) = \{a,b\}(1\{a,b\}) = \mathsf{T}$.*

The following example, first introduced by (Alviano and Faber 2019), shows that in some circumstances, s-groundedness might be overly strong:

Example 15. *Consider $\mathcal{P} = \{a \leftarrow \{a,b\} \neq 1; b \leftarrow \{a,b\} \neq 1\}$. One might expect the set $\{a,b\}$ to be acceptable. Suppose it is s-grounded. Then there is an $\{a,b\}$-trigger y for a s.t. $y \subseteq \{b\}$. This is impossible as $\{b\} \in [\emptyset,\{a,b\}]$ and $\{b\} \in [\{b\},\{a,b\}]$ and $\{b\} \cap \mathsf{dom}(\{a,b\} \neq 1) \notin \mathsf{sat}(\{a,b\})$. Thus, $\{a,b\}$ cannot be s-grounded.*

We thus introduce an even weaker notion of groundedness which we call *antecedent groundedness*, which requires that for every accepted atom, we can find a rule that is activated by a lower κ-level:

Definition 8. *A set x is* a(ntecedent)-grounded *if there is some $\kappa : x \to \mathbb{N}$ s.t. every $a \in x$, there is some $r = C \leftarrow C_1,\ldots,C_n \in \mathcal{P}$ s.t. $a \in \mathsf{dom}(C)$ and for every $i = 1,\ldots,n$ there is some $z \subseteq \{b \mid \kappa(b) < \kappa(a)\}$ s.t. $z(C_i) = \mathsf{T}$.*

S-grounded sets are a-grounded, but not vice-versa:

Example 16. *Consider again $\mathcal{P}$ from Example 12. We first observe that $\{p,q\}$ is antecedent grounded as $\emptyset(\{p,q\} \neq 1) = \mathsf{T}$ and thus we have found our antecedent justification for $\{p,q\}$. However, $\emptyset$ is* not *a $\{p,q\}$-trigger, as $\{p\} \in [\emptyset,\{p,q\}]$ and $\{p\}(\{p,q\} \neq 1) = \mathsf{F}$. Likewise, it can be seen that $\{a,b\}$ in Example 15 is a-grounded.*

The reader might think that a-groundedness is the most suitable notion of groundedness, as it avoids the behavior (deemed problematic by (Alviano and Faber 2019) in Example 15). However, (Liu et al. 2010) argued that, for $\mathcal{P}$ from Example 12, $\{p,q\}$ is not a feasible candidate for an answer set. Altogether, we see that for every notion of groundedness, one can find examples deemed as counter-intuitive in the literature.

Notice that all notions of groundedness generalize a well-known notion of groundedness proposed for normal logic programs (Erdem and Lifschitz 2003):

Proposition 11. *Let $\mathcal{P}$ a normal logic program $\mathcal{P}$. Then x is a-grounded $\mathcal{P}$ then x is grounded according to (Erdem and Lifschitz 2003), namely: there is a ranking $\kappa : x \to \mathbb{N}$ s.t. for every $a \in x$, there is some $a \leftarrow a_1,\ldots,a_n,\neg b_1,\ldots,\neg b_m \in \mathcal{P}$ s.t. for every $i = 1,\ldots,n$, $\kappa(a) > \kappa(a_i)$.*

The $\mathcal{U}$-operator does not satisfy any notion of groundedness, whereas all other operators give rise to a-grounded stable models, but only the LPST-operator and GZ-operator give rise to s-grounded stable models and only the GZ-operator gives rise to d-grounded stable models:

Proposition 12. *Let $\mathcal{P}$ be a choice program and $\dagger \in \{\mathsf{GZ},\mathsf{LPST},\mathsf{MR}\}$. Then if $(x,y) \in S^c(\mathcal{IC}^{\dagger}_{\mathcal{P}})(x,y)$:*

1. *x and y are a-grounded (for $\mathcal{P}$), and*
2. *x is s-grounded (for $\mathcal{P}$) if $\dagger \in \{\mathsf{GZ},\mathsf{LPST}\}$.*
3. *x and y are d-grounded (for $\mathcal{P}$) if $\dagger = \mathsf{GZ}$.*

There are programs $\mathcal{P}$ s.t. $(x,y) \in S^c(\mathcal{IC}^{\mathcal{U}}_{\mathcal{P}})(x,y)$ yet x and y are not antecedent grounded.

Summing up, we see that the different semantics satisfy different notions of groundedness, and that no notion of groundedness is uncontested. It seems that it depends on the application at hand which notion of groundedness is the most suitable. The results of this section are summarized in Table 3.

7 Disjunctions are Choice Constructs

Our study allows us to give a principled account of the relation between stable semantics for disjunctive logic programs (DLPs) and choice programs. Indeed, in this section, we show that for DLPs (Heyninck, Arieli, and Bogaerts 2024; Heyninck and Bogaerts 2023) are a special case of the operator for choice programs. This means that all semantics obtained on the basis of these operators coincide, thus explaining the difference between semantics for stable models of disjunctive logic programs and choice logic programs in terms of the choice of stable operator (minimality-based versus constructive). In the rest of this section, we show this claim in more detail, first providing the necessary background on DLPs and then showing DLP-operators are a special case of the operators studied in this paper.

Preliminaries on disjunctive logic programs We first recall some necessary preliminaries on disjunctive logic programming. For simplicity, we restrict attention to disjunctive logic programs whose bodies consist solely of literals, leaving the generalization of these results to stronger languages for future work.

In what follows we consider a propositional[4] 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). A (propositional) *disjunctively normal logic program* $\mathcal{P}$ in $\mathfrak{L}$ (a dlp in short) is a finite set of rules of the form $\bigvee_{i=1}^{n} p_i \leftarrow \bigwedge_{i=1}^{n} a_i \wedge \bigwedge_{i=1}^{m} \neg b_i$, where the head $\bigvee_{i=1}^{n} p_i$ is a non-empty disjunction of atoms, and the body ψ is a formula in $\mathfrak{L}$. We furthermore recall

[4]We restrict ourselves to the propositional case.

Operator	d-ground.	s-ground.	a-ground.
GZ	$\vee$	$\vee$	$\vee$
LPST	$\times$	$\vee$	$\vee$
MR	$\times$	$\times$	$\vee$
$\mathcal{U}$	$\times$	$\times$	$\times$
Example of violation	Ex. 13	Ex. 15	Ex. 12
Counter-intuitive according to	(Alviano, Faber, and Gebser 2023)	(Alviano and Faber 2019)	(Liu et al. 2010)

Table 3: Results on Postulates

that a formula of the form $\bigwedge_{i=1}^{n} a_i \wedge \bigwedge_{i=1}^{m} \neg b_i$ can be evaluated over the bilattice FOUR relative to (x,y) by assuming the involution $-$ defined by $-\mathsf{T} = \mathsf{F}$, $-\mathsf{F} = \mathsf{T}$, $-\mathsf{U} = \mathsf{U}$ and $-\mathsf{C} = \mathsf{C}$, and by defining truth assignments to complex formulas recursively as follows:

- $(x,y)(p) = \begin{cases} \mathsf{T} & \text{if } p \in x \text{ and } p \in y, \\ \mathsf{U} & \text{if } p \notin x \text{ and } p \in y, \\ \mathsf{F} & \text{if } p \notin x \text{ and } p \notin y, \\ \mathsf{C} & \text{if } p \in x \text{ and } p \notin y. \end{cases}$
- $(x,y)(\neg\phi) = -(x,y)(\phi)$,
- $(x,y)(\psi \wedge \phi) = \sqcap_{\leq_t}\{(x,y)(\phi),(x,y)(\psi)\}$,
- $(x,y)(\psi \vee \phi) = \sqcup_{\leq_t}\{(x,y)(\phi),(x,y)(\psi)\}$.

Notice that this is equivalent to the evaluations introduced in Section 4.

We recall the following ndao $\mathcal{IC}^d_{\mathcal{P}}$ introduced by (Heyninck, Arieli, and Bogaerts 2024) and defined as follows (given a dlp $\mathcal{P}$ and an interpretation (x,y)):

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

The operator $\mathcal{IC}^d_{\mathcal{P}}$ faithfully represents the semantics of dlps: In general, total stable fixpoints of $\mathcal{P}$ correspond to stable models of $\mathcal{P}$ (Gelfond and Lifschitz 1991), and weakly supported models of $\mathcal{P}$ (Brass and Dix 1995) correspond to fixpoints of $\mathcal{IC}^d_{\mathcal{P}}$ (Heyninck, Arieli, and Bogaerts 2024).

Disjunctions as Choice Constructs We now show how the operator $\mathcal{IC}^d_{\mathcal{P}}$ can be seen as a special case of the operators for choice constructs. In more detail, for a DLP $\mathcal{P}$, we define $\mathtt{D2C}(\mathcal{P}) = \{1\Delta \leftarrow \phi \mid \bigvee\Delta \leftarrow \phi \in \mathcal{P}\}$. E.g. $\mathtt{D2C}(\{p \vee q \leftarrow\}) = \{1\{p,q\} \leftarrow\}$. In other words, we replace every disjunction by the choice atom that requires at least one element of Δ is true. We will show here that the operator defined for disjunctive logic programs (Heyninck, Arieli, and Bogaerts 2024) then coincides with the operator $\mathcal{IC}^{\mathsf{x}}_{\mathtt{D2C}(\mathcal{P})}$ (for $\mathsf{x} = \mathsf{MR}, \mathsf{LPST}$). This implies that all AFT-based semantics coincide for DLPs and their conversion into choice rules. We can now point in a very exact way to the difference between DLPs and choice programs: it lies 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 defined: for disjunctions, typically (e.g. in the most popular solvers (Gebser et al. 2016; Eiter et al. 2012)), a minimality-based stable operator is used, whereas for choice constructs, the c-stable operator is more apt. Thus, disjunctive and choice programs use the same (approximation) operators, but differ in how the corresponding stable operator is constructed.

We now show that the operator $\mathcal{IC}^d_{\mathcal{P}}$ is a special case of the MR- and LPST-operators from this paper, when applied to the choice program $\mathtt{D2C}(\mathcal{P})$.

Proposition 13. *For any disjunctive normal logic program $\mathcal{P}$, $\mathcal{IC}_{\mathcal{P}} = \mathcal{IC}^{\mathsf{MR}}_{\mathtt{D2C}(\mathcal{P})} = \mathcal{IC}^{\mathsf{LPST}}_{\mathtt{D2C}(\mathcal{P})}$.*

From this, we immediately obtain that all the major semantics for disjunctive logic programming are special cases of the semantics introduced in this paper. Namely, weakly supported models of $\mathcal{P}$ coincide with fixpoints of $\mathcal{IC}^{\mathsf{MR}}_{\mathtt{D2C}(\mathcal{P})}$ whereas minimality-based stable fixpoints of $\mathcal{P}$ coincide with stable models of $\mathcal{P}$. Since, the c-stable semantics does not enforce minimality, it will in general *not* correspond to the stable models semantics:

Example 17. *Let $\mathcal{P} = \{p \vee q \leftarrow\}$. Then $(\{p,q\},\{p,q\})$ is a c-stable fixpoint of $\mathcal{IC}^{\mathsf{MR}}_{\mathtt{D2C}(\mathcal{P})}$ but it is not a stable model of $\mathcal{P}$ according to (Gelfond and Lifschitz 1991).*

This also gives an answer to the question of how to combine disjunctions and choice constructs in logic programs: a choice as to which stable semantics are used has to be made: either preserve the minimality of answer sets as in DLPs and loses some reasonable 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. On the other hand, we can now also use minimality-based stable semantics on more complicated choice construct than simple disjunctions.

8 Related Work

To the best of our knowledge, this is the first application of AFT to the semantics of choice programs. We have shown how major semantics for choice programs (Marek and Remmel 2004; Liu et al. 2010) can be characterized in our framework. Other well-known semantics for (disjunctive) aggregate programs are those by (Faber, Leone, and Pfeifer 2004; Alviano and Faber 2019; Ferraris 2011), which are the ones used in the solvers DLV respectively clingo (Ferraris 2011; Alviano et al. 2017).

Another semantics, the FLP-semantics (Faber, Leone, and Pfeifer 2004), were originally not defined for programs with choice constructs in the head, but were generalized to allow choice constructs in the head by (Eiter and Geibinger 2023). Nevertheless, these semantics only allow for two-valued stable models, and were shown to differ from the semantics by (Gelfond and Zhang 2014), (Liu et al. 2010) and (Marek and Remmel 2004) already for aggregate programs (Alviano, Faber, and Gebser 2023), which means, in view of Proposition 10, that the stable semantics induced by the LPST-, MR- and GZ-operators also differ from the FLP-semantics. This comparison also holds for the semantics by (Ferraris 2011) as the latter coincide with the FLP-semantics for aggregates with positive atoms (Alviano, Faber, and Gebser 2023).

Another line of work that is relevant for this paper is the application of AFT to logic programs with aggregates. (Pelov, Denecker, and Bruynooghe 2007) introduce several approximation operators for aggregate programs, whereas (Pelov, Denecker, and Bruynooghe 2007) introduces operator-based semantics for disjunctive aggregate programs. These semantics were generalized by (Heyninck and Bogaerts 2023) in the framework of non-deterministic AFT. Generalizing the operator by (Pelov, Denecker, and Bruynooghe 2007) to choice programs is left for future work. An overview of semantics that are (non-)representable in the deterministic AFT-framework is given by (Vanbesien, Bruynooghe, and Denecker 2022). This work severed as an important foundation of our paper, as the operators $\mathcal{IC}_{\mathcal{P}}^{\mathsf{GZ}}$ and $\mathcal{IC}_{\mathcal{P}}^{\mathsf{MR}}$ are generalizations of the operator-based characterisations by (Vanbesien, Bruynooghe, and Denecker 2022) of the corresponding semantics.

9 Conclusion

The main contributions of this paper are the definition of several ndaos for choice programs, the definition of the constructive stable operator, the characterisation of several existing semantics for various dialects of logic and choice programs and the introduction and study of postulates for choice programs. We provide a principled view of choice programs versus disjunctive programs.

This paper is subject to one restriction: we assume $\mathcal{IC}^c_{\mathcal{P}}(x,y) \neq \emptyset$ for every interpretation (x,y). In future work, we will generalize our results beyond this assumption. We also plan to study the complexity of the resulting semantics, device implementations and study other AFT-based semantics, such as the Kripke-Kleene and well-founded states and semi-equilibrium semantics (Heyninck and Bogaerts 2023; Heyninck, Arieli, and Bogaerts 2024) and study other operators, e.g. inspired by (Pelov, Denecker, and Bruynooghe 2007).

Acknowledgements

I thank Bart Bogaerts and Wolfgang Faber for interesting discussions on this topic, and the reviewers for their dilligent reviews.

References

Alviano, M., and Faber, W. 2019. Chain answer sets for logic programs with generalized atoms. In Calimeri, F.; Leone, N.; and Manna, M., eds., *Logics in Artificial Intelligence - 16th European Conference, JELIA2019, Rende, Italy, May 7-11, 2019, Proceedings*, volume 11468 of *Lecture Notes in Computer Science*, 462–478. Springer.

Alviano, M.; Calimeri, F.; Dodaro, C.; Fuscà, D.; Leone, N.; Perri, S.; Ricca, F.; Veltri, P.; and Zangari, J. 2017. The asp system dlv2. In *Logic Programming and Nonmonotonic Reasoning: 14th International Conference, LPNMR 2017, Espoo, Finland, July 3-6, 2017, Proceedings 14*, 215–221. Springer.

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

Brass, S., and Dix, J. 1995. Characterizations of the stable semantics by partial evaluation. In *Proceedings of LPNMR'95*, 85–98. Springer.

Calimeri, F.; Faber, W.; Gebser, M.; Ianni, G.; Kaminski, R.; Krennwallner, T.; Leone, N.; Ricca, F.; and Schaub, T. 2012. Asp-core-2: Input language format. *ASP Standardization Working Group*.

Cousot, P., and Cousot, R. 1979. Constructive versions of tarski's fixed point theorems. *Pacific journal of Mathematics* 82(1):43–57.

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

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

Eiter, T., and Geibinger, T. 2023. Explaining answer-set programs with abstract constraint atoms. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI 2023, to appear)*.

Eiter, T.; Fink, M.; Krennwallner, T.; and Redl, C. 2012. Conflict-driven asp solving with external sources. *Theory and Practice of Logic Programming* 12(4-5):659–679.

Erdem, E., and Lifschitz, V. 2003. Tight logic programs. *Theory and Practice of Logic Programming* 3(4-5):499–518.

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

Ferraris, P. 2011. Logic programs with propositional connectives and aggregates. *ACM Transactions on Computational Logic (TOCL)* 12(4):1–40.

Gebser, M.; Kaminski, R.; Kaufmann, B.; Ostrowski, M.; Schaub, T.; and Wanko, P. 2016. 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.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New generation computing* 9(3-4):365–385.

Gelfond, M., and Zhang, Y. 2014. Vicious circle principle and logic programs with aggregates. *Theory and Practice of Logic Programming* 14(4-5):587–601.

Heyninck, J., and Bogaerts, B. 2023. Non-deterministic approximation operators: Ultimate operators, semi-equilibrium semantics, and aggregates. *Theory Pract. Log. Program.* 23(4):632–647.

Heyninck, J.; Arieli, O.; and Bogaerts, B. 2024. Non-deterministic approximation fixpoint theory and its application in disjunctive logic programming. *Artif. Intell.* 331:104110.

Heyninck, J. 2023. Semantics for logic programs with choice constructs on the basis of approximation fixpoint theory (preliminary report). In Sauerwald, K., and Thimm, M., eds., *Proceedings of the 21st International Workshop on Non-Monotonic Reasoning co-located with the 20th International Conference on Principles of Knowledge Representation and Reasoning (KR 2023) and co-located with the 36th International Workshop on Description Logics (DL 2023), Rhodes, Greece, September 2-4, 2023*, volume 3464 of *CEUR Workshop Proceedings*, 74–83. CEUR-WS.org.

Heyninck, J. 2024. Operator-based semantics for choice programs: is choosing losing? (full version). `https://arxiv.org/abs/2407.21556`.

Kaminski, R., and Schaub, T. 2023. On the foundations of grounding in answer set programming. *Theory and Practice of Logic Programming* 23(6):1138–1197.

Kuratowski, K. 1922. Une méthode d'élimination des nombres transfinis des raisonnements mathématiques. *Fundamenta mathematicae* 3(1):76–108.

Liu, L.; Pontelli, E.; Son, T. C.; and Truszczyński, M. 2010. Logic programs with abstract constraint atoms: The role of computations. *Artificial Intelligence* 174(3-4):295–315.

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

Pelov, N.; Denecker, M.; and Bruynooghe, M. 2007. Well-founded and stable semantics of logic programs with aggregates. *Theory and Practice of Logic Programming* 7(3):301–353.

Przymusinski, T. C. 1990. The well-founded semantics coincides with the three-valued stable semantics. *Fundamenta Informaticae* 13(4):445–463.

Simons, P.; Niemelä, I.; and Soininen, T. 2002. Extending and implementing the stable model semantics. *Artificial Intelligence* 138(1-2):181–234.

Son, T. C., and Pontelli, E. 2007. A constructive semantic characterization of aggregates in answer set programming. *Theory and Practice of Logic Programming* 7(3):355–375.

Vanbesien, L.; Bruynooghe, M.; and Denecker, M. 2022. Analyzing semantics of aggregate answer set programming using approximation fixpoint theory. *Theory and Practice of Logic Programming* 22(4):523–537.

Zorn, M. 1935. A remark on method in transfinite algebra. *Bulletin of the American Mathematical Society* 41(10):667–670.