TPLP **22** (4): 523–537, 2022.
doi:10.1017/S1471068422000126 First published online 22 July 2022

*Analyzing Semantics of Aggregate Answer Set Programming Using Approximation Fixpoint Theory**

LINDE VANBESIEN, MAURICE BRUYNOOGHE and MARC DENECKER
Department of Computer Science, KU Leuven, Leuven, Belgium
(*e-mails:* linde.vanbesien@cs.kuleuven.be, maurice.bruynooghe@cs.kuleuven.be, marc.denecker@cs.kuleuven.be)

submitted 17 May 2022; accepted 8 June 2022

Abstract

Aggregates provide a concise way to express complex knowledge. The problem of selecting an appropriate formalization of aggregates for answer set programming (ASP) remains unsettled. This paper revisits it from the viewpoint of Approximation Fixpoint Theory (AFT). We introduce an AFT formalization equivalent with the Gelfond–Lifschitz reduct for basic ASP programs and we extend it to handle aggregates. We analyze how existing approaches relate to our framework. We hope this work sheds some new light on the issue of a proper formalization of aggregates.

KEYWORDS: aggregates, Approximation Fixpoint Theory, Answer set Programming

1 Introduction

Aggregate expressions are very useful and have been added to classical logic, query languages, constraint languages, and also to logic programming (LP) and answer set programming (ASP). The effort it takes to add aggregates to (syntax and semantics of) a logic is very language dependent. For example, to extend first order logic (FO) with a *Count* aggregate, we extend the definition of "term" with a new inductive rule: "*If* $a_1, \ldots, a_n$ *are variables, and* ψ *a formula, then* $Count(\{(a_1, \ldots, a_n), \psi\})$ *is a term*" and the definition of "interpretation of a term t in structure I" (used in the definition of $\models$) with: "*If* $t = Count(\{(a_1, .., a_n), \psi\})$ *then* t^I *is* $\#\{(d_1, \ldots, d_n) \in Dom(I)^n | I[a_1 : d_1, \ldots, a_n : d_n] \models \psi\}$*, that is, the number of tuples that satisfy* ψ *in* I*.*". The method is simple and follows Frege's compositionality principle.

In LP and ASP, it is much more difficult. Research into extensions with aggregates (well-founded and stable semantics) started with the work of Kemp and Stuckey (1991). Many approaches exist, but so far no consensus on how to handle aggregates in those logics has been reached.

* This research received funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme and from the FWO (Research Foundation Flanders) for project G0B2221N.

We propose a framework for defining semantics for extensions of LP and ASP based on Approximation Fixpoint Theory (AFT) introduced by Denecker *et al.* (2000) and (2004). AFT is an abstract lattice theoretic formalization of constructive methods for non-monotonic operators. It defines different types of constructions and fixpoints to a lattice operator in an approximation space, including supported, Kripke–Kleene (KK), stable and well-founded (WF) fixpoints. The theory has been applied to a range of non-monotonic logics to characterize existing as well as new semantics: for example, LP and ASP as shown in the paper by Denecker *et al.* (2012), autoepistemic and default logic as shown in the paper by Denecker *et al.* (2003), higher order LP as shown in the paper by Charalambidis *et al.* (2018), argumentation frameworks and abstract dialectal frameworks as shown in the papers by Strass (2013) and Bogaerts (2019). AFT has been applied to aggregate LP and ASP resulting in the ultimate stable and well-founded semantics in the work by Denecker *et al.* (2001) and the broader framework in the paper by Pelov *et al.* (2007) where stable and well-founded semantics are induced by a choice of a 3-valued truth function.

Here, we clarify and expand this work. First, to make the AFT framework more accessible to the ASP community, we show how each approximation truth assignment can be broken up in a lower and an upper ternary satisfaction relation which, in the context of ASP can be easily related to the reduct approach originally used by Gelfond and Lifschitz (1988) to define stable semantics. Then we focus on aggregate ASP using examples from the literature. Where possible, we consider aggregate atoms with positive and negative literals as conditions. However, the semantics described by Gelfond and Zhang (2019) does not allow *negation by default* inside an aggregate atom, therefore we only consider positive conditions for this specific case. It is shown that not only the semantics of Denecker *et al.* (2001), Pelov *et al.* (2007) but also those of Liu *et al.* (2010) and Gelfond and Zhang (2019) are instances of our framework. But not all proposed semantics for aggregate ASP semantics belong to our framework; for example those of Ferraris (2011), Marek and Remmel (2004), Faber *et al.* (2011). We investigate the reason for this. The paper contributes to the discussion about semantics for Aggregate ASP by clarifying some important principles of NMR and by showing where they are applied and where other principles are applied.

2 Approximation fixpoint theory

Here, we recall the basics of AFT from the work by Denecker *et al.* (2000). In many non-monotonic languages, a theory defines a semantic lattice[1] operator $O : L \to L$. If O is monotone, its least fixpoint is often taken as the semantics of the theory. Otherwise, AFT can be applied. The first step is to associate the approximation space L^2 to L. A pair $(x, y) \in L^2$ is an approximation of any $z \in [x, y]$. With $x \leq y$, the interval is non-empty and the pair is *consistent*. L^c is the subspace of consistent pairs. L^2 and L^c possess (i) a precision order, $(x, y) \leq_p (u, v)$ if $x \leq u$ and $y \geq v$ and (ii) the embedding

[1] A lattice $\langle L, \leq\rangle$ is a partially ordered set where each subset S has a least upper bound $lub(S)$ and a greatest lower bound $glb(S)$.

https://doi.org/10.1017/S1471068422000126 Published online by Cambridge University Press

of L, namely the set of *exact pairs* (x, x) which approximate only x. The least precise point is $(\bot, \top)$, with $\bot = glb(L), \top = lub(L)$.

The second step is to assign an approximating operator A to O: a $\leq_p$-monotone operator on L^2 or L^c such that if (x, y) approximates z then $A(x, y)$ approximates $O(z)$; an approximator on L^2 also has to be symmetric: $A(x, y) = (u, v)$ iff $A(y, x) = (v, u)$. Thus, increasing the precision of the input to an approximator A increases the precision of the output. With an approximator A, several types of fixpoints are definable. The least fixpoint (lfp) construction $(\bot, \top)$, $A(\bot, \top)$, $\ldots$, $A^\alpha(\bot, \top)$, $\ldots$ produces the KK fixpoint $KK(A) = lfp(A)$. The KK fixpoint is a pair $(x, y) \in L^2$. If exact, $x(= y)$ is the only fixpoint of O. Otherwise, it approximates all fixpoints of O, including "self-supported" ones that are often not minimal in L. Stable and WF fixpoint definitions contain mechanisms to reduce self-support.[2] A stable fixpoint $x \in L$ is one such that $x = lfp(\lambda z : A(z, x)_1)$, where $A(z, x)_1$ is the first component of the pair $A(z, x)$. The WF fixpoint $WF(A)$ is the least precise pair (x, y) with $x = lfp(\lambda z : A(z, y)_1)$ and $y = lfp(\lambda z : A(x, z)_2)$. It is the least precise fixpoint of the monotone operator $(x, y) \mapsto (lfp(\lambda z : A(z, y)_1), lfp(\lambda z : A(x, z)_2))$. We have that $KK(A)$ and $WF(A)$ are consistent and for each stable fixpoint x, $KK(A) \leq_p WF(A) \leq_p x$ and x is a minimal fixpoint of O.[3]

AFT induces relationships between fixpoints of different approximators of O. If approximator A is pointwise less precise than B, then $KK(A) \leq_p KK(B)$, $WF(A) \leq_p WF(B)$ and any stable fixpoint of A is a stable fixpoint of B. Thus, with increasing precision of the approximator, KK and WF fixpoints increase in precision, and the set of stable fixpoints grows. There exists a most precise approximator Ult_O of O, with the most precise KK and WF fixpoint and the largest set of stable fixpoints. For consistent pairs (x, y), $Ult_O(x, y)$ is the most precise pair approximating $O([x, y])$, that is, $(glb(O([x, y]), lub(O([x, y])))$. It follows, perhaps surprisingly, that if an even moderately precise approximator A has a stable fixpoint x, then x is approximated by the most precise WF fixpoint $WF(Ult_O)$ associated with O (but keep in mind that unprecise approximators are unlikely to have stable fixpoints). This counter-intuitive fact tells us that in all cases, if stable fixpoints exist, they are in the proximity of the most precise WF fixpoint that can be associated to O. This may explain the good quality of stable semantics in capturing intuitions. But while KK and WF are constructive, stable semantics is not really. It only has a constructive test: testing if x is stable is by testing if x is the limit of the *lfp* construction of $\lambda z : A(z, x)_1$.

We now sketch how to use AFT to define constructive semantics for programs based on some logic $\mathcal{L}$.[4] We assume $\mathcal{L}$ is a first-order logic with a (Herbrand) model semantics

[2] The intuition of self-support is not easily explained in an algebraic setting but shows intuitively in the logic program $\{p \leftarrow p.\ \ q \leftarrow \neg p\}$. It has two minimal fixpoints $\{p\}$ and $\{q\}$, but $\{p\}$ is self-supported (in p) while $\{q\}$ is not. The second is the unique stable and well-founded fixpoint.

[3] In case of an L^c-operator, the domain of $\lambda z : A(z, y)_1$ is $[\bot, y]$ and that of $\lambda z : A(x, z)_2$ is $[x, \top]$ while the range of both operators is L. So, it is possible that the iterated lfp construction of one of these operators terminates in a point outside the operator domain in which case the operator has neither a fixpoint nor a lfp. To accommodate, we call x a stable fixpoint of A if the lfp of $\lambda z : A(z, y)_1$ exists and is equal to x. In the paper by Denecker *et al.* (2004), it was proven that every L^c-approximator A is expandable to L^2 approximators, and that each such an expansion has the same KK, WF and stable models as A. Therefore, we spend little attention to the difference between L^2 and L^c.

[4] For ease of discussion, this work only considers Herbrand interpretations and ground programs.

https://doi.org/10.1017/S1471068422000126 Published online by Cambridge University Press

defined using a 2-valued truth function $\mathcal{H}^2$, or equivalently, a satisfaction relation $\models_2$ (where $I \models_2 \phi$ iff $\mathcal{H}^2_I(\phi) = \mathbf{t}$). An $\mathcal{L}$-program is then defined as follows:

Definition 1 (*$\mathcal{L}$-program*)
An $\mathcal{L}$-program is a set of rules r of the form $p \leftarrow \psi$ such that the body ψ is a formula of $\mathcal{L}$ and the head p is a propositional atom.

Importantly, $\leftarrow$ is not a connective of $\mathcal{L}$ but AFT defines its meaning as a *construction operator*. LP and (non-disjunctive) ASP are instances of this where $\mathcal{L}$ is simply the logic of conjunctions of literals p or $\neg p$ under standard interpretation. Given an $\mathcal{L}$-program P, the corresponding lattice $L, \leq$ is the set of P's Herbrand interpretations ordered by the truth order ($\mathbf{f} < \mathbf{t}$). The sets L^c and L^2 correspond to 3- and 4-valued interpretations. Any 3- or 4-valued interpretation $\mathcal{I}$ can be split into a pair (I, J) by splitting truth values as in $\mathbf{t} \sim (\mathbf{t}, \mathbf{t}), \mathbf{f} \sim (\mathbf{f}, \mathbf{f}), \mathbf{u} \sim (\mathbf{f}, \mathbf{t})$ and $\mathbf{i} \sim (\mathbf{t}, \mathbf{f})$. If $\mathcal{I}$ is 3-valued, then $I \leq J$, and I is a lower bound, J an upper bound of $\mathcal{I}$. The 3- and 4-valued structures are equipped with a truth order, which isomorphically corresponds to the product order $\leq$ of L^2 and L^c, and with a precision order $\leq_p$ ($\mathbf{u} <_p \mathbf{t} <_p \mathbf{i}, \mathbf{u} <_p \mathbf{f} <_p \mathbf{i}$ which corresponds to the precision order of L^2 and L^c. Any truth or precision monotone operator Γ on 2-, 3- or 4-valued structures corresponds to a truth or precision monotone operator on L, L^c, L^2.

An $\mathcal{L}$-program P is characterised by the immediate consequence operator $T_P : L \to L$. This operator fulfills the role of O, the approximated operator. The immediate consequence operator $T_P : L \to L$ for a program P is such that $T_P(I) = J$ if for every ground atom p, $\mathcal{H}^2_J(p) = lub_{\leq}(\{\mathcal{H}^2_I(\psi) | (p \leftarrow \psi) \in P\})$. There are two ways to define AFT semantics using an approximator $A(= A_P)$ for T_P.

One way is to use $A_P = Ult(T_P)$, the most precise approximator of T_P in L^c leading to ultimate versions of the family of AFT semantics as described by Denecker *et al.* (2004) and used by Denecker *et al.* (2001) for defining semantics of Aggregate LP. This is the most precise approach, but computationally costly. The other way is to extend $\mathcal{L}$'s truth assignment to 3- or 4-valued interpretations and by interpreting T_P's definition in this broader context. For example, a 3-valued truth assignment $\mathcal{H}^3$ induces a 3-valued immediate consequence operator Φ_P where $\Phi_P(\mathcal{I}) = \mathcal{I}'$ if for every atom p, $\mathcal{H}^3_{\mathcal{I}'}(p) = lub_{\leq}\{\mathcal{H}^3_{\mathcal{I}}(\psi) | (p \leftarrow \psi) \in P\}$. The operator Φ_P corresponds isomorphically to an L^c approximator A_P on pairs $I < J$ of 2-valued interpretations. But for A_P to be an approximator of T_P, the 3-valued truth assignment $\mathcal{H}^3$ should satisfy a condition introduced for 3-valued logic by Kleene (1952):

Definition 2 (*Regular truth assignment*)
A 3-valued truth assignment $\mathcal{H}^3$ of $\mathcal{L}$ is *regular* iff for all formulas ψ, for all 3-valued structures $\mathcal{I}$ interpretating ψ: (1) (extension of $\mathcal{H}^2$) if $\mathcal{I}$ is 2-valued, then $\mathcal{H}^2_{\mathcal{I}}(\psi) = \mathcal{H}^3_{\mathcal{I}}(\psi)$ and (ii) (precision monotonocity) if $\mathcal{I} \leq_p \mathcal{I}'$ then $\mathcal{H}^3_{\mathcal{I}}(\psi) \leq_p \mathcal{H}^3_{\mathcal{I}'}(\psi)$.[5]

For example, Kleene's strong 3-valued truth assignment $\mathcal{H}^{SK}$ of FO (introduced by Kleene 1952) and Belnap's 4-valued extension (introduced by Belnap 1977) are regular. They induce multi-valued extensions Φ_P of T_P first introduced by Fitting (1985). Later, Φ_P was found to correspond to an approximator A_P of T_P whose KK, WF and stable fixpoints correspond to the semantics of the same name.

[5] For the 4-valued case, an additional condition is symmetry.

https://doi.org/10.1017/S1471068422000126 Published online by Cambridge University Press

In LP and ASP, it is often taken for granted that LP's nonmonotonicity is due to its non-classical negation *not*. But it is evident in AFT-based semantics, that the main non-classical connective is the rule operator: its semantics is defined via operators and constructive processes while negation in bodies is treated like the other FO connectives, using three-valued logic only for approximation of the standard classical connectives.

The AFT road is quite unlike other semantic techniques in ASP. In the next section, we reformulate the framework in more accessible terms for the ASP community.

3 Ternary satisfaction relations

Ternary satisfaction relations were used originally by Liu *et al.* (2010) in the context of Aggregate ASP semantics where they were called *sub-satisfiability relations.*

We still assume a base logic $\mathcal{L}$ equipped with satisfaction relation $\models_2$. Below, we restrict ourselves to 3-valued interpretations corresponding to pairs (I, J) of 2-valued Herbrand interpretations $I \subseteq J$ (but extension to non-Herbrand interpretations is possible).

Definition 3 (*Ternary satisfaction relations*)
A *ternary satisfaction relation* (*TSR*) $\models_3$ of $\mathcal{L}$ is a relation between pairs (I, J) of interpretations such that $I \subseteq J$, and formulas ψ of $\mathcal{L}$ such that $I \models_2 \psi$ iff $(I, I) \models_3 \psi$. It is *lower-monotone* if $(I, J) \models_3 \psi$ implies $(I', J) \models_3 \psi$ when $I \subseteq I' \subseteq J$. It is *lower-regular* (*upper-regular*) if $(I, J) \models_3 \psi$ implies $(I', J') \models_3 \psi$ when $(I, J) \leq_p (I', J')$ $((I', J') \leq_p (I, J))$.

For any pair of a lower-regular *TSR* $\models_3$ and an upper-regular *TSR* $\models_3^{\uparrow}$, it always holds that $\models_3 \subseteq \models_3^{\uparrow}$ since $(I, J) \models_3 \psi$ implies $(I, I) \models_2 \psi$ which implies $(I, J) \models_3^{\uparrow} \psi$. Also, a three-valued truth-function $\mathcal{H}^3$ corresponds one to one to pairs $(\models_3, \models_3^{\uparrow})$ of *TSRs* satisfying $\models_3 \subseteq \models_3^{\uparrow}$. The correspondence is: (i) $(I, J) \models_3 \psi$ iff $\mathcal{H}^3_{(I,J)}(\psi) = \mathbf{t}$ and (ii) $(I, J) \models_3^{\uparrow} \psi$ iff $\mathcal{H}^3_{(I,J)}(\psi) \in \{\mathbf{t}, \mathbf{u}\}$.

Proposition 1
$\mathcal{H}^3$ is regular iff $\models_3$ is a lower- and $\models_3^{\uparrow}$ an upper-regular *TSR*.[6]

Taking $\mathcal{L}$ as FO and $\mathcal{H}^3$ as the strong Kleene truth assignment, it is a folk result that $(I, J) \models_3 \psi$ if ψ evaluates to true when interpreting all positively occurring atoms in I and all negatively occurring ones in J. For $(I, J) \models_3^{\uparrow} \psi$, exchange the roles of I and J.

For each $\mathcal{L}$-program P, the lower- and upper-regular *TSR* induce two distinct operators on consistent pairs $I \subseteq J$: $A_P^{\models_3}(I, J) = \{p | \exists (p \leftarrow \psi) \in P : (I, J) \models_3 \psi\}$ and $A_P^{\models_3^{\uparrow}}(I, J) = \{p | \exists (p \leftarrow \psi) \in P : (I, J) \models_3^{\uparrow} \psi\}$. Now, we define $A_P(I, J) = (A_P^{\models_3}(I, J), A_P^{\models_3^{\uparrow}}(I, J))$.

Proposition 2
If $\models_3$ and $\models_3^{\uparrow}$ are lower- and upper-regular *TSRs*, then A_P is an L^c approximator. Moreover it is isomorphic to the 3-valued Φ_P induced by the 3-valued truth assignment $\mathcal{H}^3$ combining $\models_3$ and $\models_3^{\uparrow}$.

Now, supported, KK, WF and stable models of the $\mathcal{L}$-programs P can be defined in terms of $\models_3$ and $\models_3^{\uparrow}$. Interestingly, J is a stable model of P iff J is the least fixpoint

[6] All proofs are in the supplementary material corresponding to this paper at the TPLP archives.

of $A_P^{\models_3}(I,J) = \lambda I \in [\bot, J] : \{p | \exists (p \leftarrow \psi) \in P : (I,J) \models_3 \psi\}$. No need of $\models_3^{\uparrow}$! Clearly there exists an asymmetry between truth and falsity in stable models. While information about the truth of formulas, encoded by $\models_3$, is essential to determine the stable fixpoints of a program, information about their falsity, given by $\models_3^{\uparrow}$, is disregarded.

4 Generalizing the concept of answer set

Here, we generalize stable models of $\mathcal{L}$-programs to answer sets. Now, $\mathcal{L}$, the logic of the rule bodies, has, besides a satisfaction relation $\models_2$ also a *TSR* $\models_3$.

Definition 4
I is an answer set of P if (1) for every $(p \leftarrow \psi) \in P$, if $I \models_2 \psi$ then $I \models_2 p$; (2) there is no $J \subset I$ such that for every $(p \leftarrow \psi) \in P$, if $(J,I) \models_3 \psi$ then $(J,I) \models_3 p$.

Proposition 3 (semi-constructive answer sets)
If $\models_3$ is lower-monotone, then for $\mathcal{L}$-programs P, I is an answer set of P iff I is the limit of the increasing sequence $\langle I_\alpha \rangle_{\alpha \geq 0}$ where (1) $I_0 = \emptyset$, (2) $I_{\alpha+1} = A_P^{\models_3}(I_\alpha, I)$ if $I_\alpha \subseteq I$, (3) $I_\lambda = \bigcup_{\alpha<\lambda} I_\alpha$ for limit ordinal λ.

In general, the (transfinite) fixpoint sequence may leave $[\emptyset, J]$, or end up with a fixpoint $I \subseteq J$. In case it is J, it is an answer set. Let $\models_3$ be lower-regular. Combining it with *any* upper-regular *TSR* $\models_3^{\uparrow}$ induces a regular truth assignment $\mathcal{H}^3$, as well as an entire family of AFT fixpoints and models of $\mathcal{L}$-programs: KK models, WF models and AFT-stable models approximated by the WF models. The AFT-stable models in this framework depend only on $\models_3$ and they correspond exactly to answer sets of $\models_3$, since a lower-regular *TSR* is also lower-monotone.

Proposition 4
Answer sets and AFT-stable models coincide for $\mathcal{L}$ programs with a lower regular *TSR*.

To finish this section, we analyze the link with the original definition of answer set given by Gelfond and Lifschitz (1988). There, $\mathcal{L}$ is the logic of ground sets/conjunctions of literals with FO's standard satisfaction relation $\models_2$. Let P be an $\mathcal{L}$-program.

Definition 5 (Gelfond-Lifschitz reduct and answer set)
The Gelfond-Lifschitz reduct (defined by Gelfond and Lifschitz 1988) P^J of P for an interpretation J is obtained from P by deleting

- all rules with a negative literal $\neg l$ such that $l \in J$.
- all negative literals in the bodies of the remaining rules.

J is a GL-answer set of P if $J \models_2 P^J$ and there is no $I \subset J$ such that $I \models_2 P^J$.

We now identify the lower-regular *TSR* $\models_{GL}$ such that answer sets of programs induced by $\models_{GL}$, coincide with GL-answer sets. The *TSR* that is needed evaluates bodies ψ in pairs (I,J) by interpreting atomic literals of ψ in I and negative literals in J. But as explained in the previous section, that is how the lower-regular *TSR* $\models_{SK}$ of the strong Kleene truth assignment operates: $(I,J) \models_{SK} \psi$ iff atoms in ψ hold in I and negative literals hold in J. Thus, $\models_{GL}$ is the restriction of $\models_{SK}$ to conjunctions of literals. With this in mind, the following proposition is straightforward.

https://doi.org/10.1017/S1471068422000126 Published online by Cambridge University Press

Proposition 5
For $I \in [\bot, J]$, $I \models_2 P^J$ iff for every rule $p \leftarrow \psi \in P$, if $(I, J) \models_{GL} \psi$ then $(I, J) \models_{GL} p$. J is a GL-answer set of P iff J is an AFT-stable model of P under strong Kleene truth assignment.

Thus, this type of answer set fits in the AFT-landscape of KK, WF and stable models.

5 Aggregates programs in the AFT framework

Aggregate Programs For the remainder of the text, we will consider $\mathcal{L}$-programs where $\mathcal{L}$ is the logic of conjunctions of literals and positive aggregate atoms.[7]

An aggregate atom a^{Aggr} is of the form: $Agg(\{a_1 : cond_1, ..., a_n : cond_n\}) * w$ with aggregate symbol Agg (e.g., SUM), comparison connective $*$ (e.g., $\leq, =, \neq, \dots$), numerical value w and multiset $\{a_1 : cond_1, ..., a_n : cond_n\}$ wher each $cond_i$ is a literal and each a_i is a weight. The $\models_2$ relation for $\mathcal{L}$ is naturally extended with a rule for evaluating aggregate atoms, so also T_P is defined. This enables the first approach to apply AFT on this type of programs, using the 3-valued ultimate approximator Ult_{T_P} yielding the ultimate KK, WF and stable semantics. Up to the syntax, this is the semantics of Denecker *et al.* (2001). The second approach to apply AFT is based on defining a regular 3-valued truth assignment for aggregate atoms, leading to a three valued operator Φ_P^{Aggr}. Up to the syntax, this was the approach followed by Pelov *et al.* (2007).

We now start the study of existing approaches for handling aggregates in ASP. Some of these use more extensive programs than the ones analysed here, however our analysis only considers the simpler $\mathcal{L}$-programs. Proposition 3 shows that there is a constructive test for answer sets of non-disjunctive $\mathcal{L}$-programs for semantics with lower-monotone *TSR*'s. A lower-regular *TSR* is lower-monotone by definition. Thus, this constructive test is applicable for all semantics that fit in the AFT-framework. However, not all semantics for ASP programs in the literature have lower-regular or even lower-monotone ternary satisfaction relations.

Before we start, we define the precision order on the *TSRs* analogous to the precision order on truth assignments defined by Pelov *et al.* (2007).

Definition 6 (*Precision relation over lower ternary satisfaction relations*)
A *TSR* $\models_a$ is less precise than a *TSR* $\models_b$, or $\models_a \leq_p \models_b$, iff for every formula ψ and every pair of two-valued interpretations (I, J): $(I, J) \models_a \psi$ implies $(I, J) \models_b \psi$.

Proposition 6
Let $\models_a, \models_b$ be *TSR*s that coincide with $\models_{GL}$ on aggregate free bodies. If $\models_a \leq_p \models_b$ and J is an answer set associated with $\models_a$ (an a-answer set), then J is a b-answer set.

5.1 Approaches that fit in the AFT framework

Pelov et al. (2007) This paper introduces several regular truth assignments for aggregate atoms. This is equivalent with expanding the lower- and upper-regular *TSR* with a rule

[7] In many semantics, including AFT semantics, a negated aggregate atom can always be represented by its dual positive aggregate atom. For example, $\neg(SUM(\{1 : s\}) > 0)$ corresponds to $SUM(\{1 : s\}) \leq 0$.

https://doi.org/10.1017/S1471068422000126 Published online by Cambridge University Press

for aggregate atoms. For defining answer sets for the syntax of this paper, the lower one suffices.[8] The least precise approximation, $\mathcal{H}^{triv}$ assigns to an aggregate atom a^{Aggr} the same value as I and J when I and J agree on the conditions in a^{Aggr} and $\mathbf{u}$ when they disagree. The corresponding lower TSR is:

Definition 7 ($\models_{triv}$)
$\models_{triv}$ extends $\models_{GL}$ with: $(I, J) \models_{triv} a^{Aggr}$ iff $J \models_2 a^{Aggr}$ and ${cond_i}^I = {cond_i}^J$ for every condition $cond_i$ in a^{Aggr}.

The most precise regular truth assignment $\mathcal{H}^{ult}$ assigns $\mathbf{t}$ ($\mathbf{f}$) to an aggregate expression in (I, J) if it is $\mathbf{t}$ ($\mathbf{f}$) in every $Z \in [I, J]$. Otherwise, it assigns $\mathbf{u}$. The corresponding lower TSR is:

Definition 8 ($\models_{ult}$)
$\models_{ult}$ extends $\models_{GL}$ with: $(I, J) \models_{ult} a^{Aggr}$ iff for each Z such that $I \subseteq Z \subseteq J$: $Z \models_2 a^{Aggr}$.

Pelov (2004) shows that for stratified aggregate programs (where predicates in aggregate expressions are defined at a lower level), the trivial and the ultimate truth assignments lead to the same semantics.

While very precise, Pelov (2004) shows that the complexity of computing KK, WF and stable models under $\models_{ult}$ moves to the next level of the polynomial hierarchy. To avoid this, he offers a less precise alternative called the *bounded* truth assignment.

Phrased in terms of the present aggregate programs, it uses functions $LB_{Agg}, UB_{Agg} : \mathcal{P}(\tilde{D}_1) \rightarrow D_2$ that maps any three-valued multiset $\{ms\}^{\mathcal{I}}$ to respectively the minimum and the maximum of $\{Agg(\{ms\})^{I'} | I' \in [I, J]\}$; that is, LB_{Agg} represents the lower bound for the aggregate function Agg on the possible multisets and UB_{Agg} the upperbound. The truth value for aggregate atoms with sum and product is based on these bounds. The corresponding lower TSR is:

Definition 9 ($\models_{bnd}$)
$\models_{bnd}$ agrees with $\models_{ult}$ except for aggregate atoms of the form $Agg(\{ms\}) * w$ with $Agg \in \{SUM, PROD\}$ and $* \in \{=, \neq\}$

- $(I, J) \models_{bnd} Agg(\{ms\}) = w$ iff $LB_{Agg}(\{ms\}^{(I,J)}) = w = UB_{Agg}(\{ms\}^{(I,J)})$.
- $(I, J) \models_{bnd} Agg(\{ms\}) \neq w$ iff $LB_{Agg}(\{ms\}^{(I,J)}) > w$ or $UB_{Agg}(\{ms\}^{(I,J)}) < w$.

Pelov (2004) lists polynomial algorithms to compute both bounds for all aggregate atoms discussed in his thesis. The same holds for the common aggregate atoms in ASP. Consequently, the complexity of computing the different types of models remains on the same level as for the non-aggregate case. Yet, the bound semantics is precise enough to solve many useful aggregate programs with recursion over the aggregates.

Proposition 7
The $TSRs$ $\models_{triv}$, $\models_{ult}$ and $\models_{bnd}$ are lower-regular. Since $\models_{triv} \leq_p \models_{bnd} \leq_p \models_{ult}$, an answer set of $\models_{triv}$ is one of $\models_{bnd}$, and one of $\models_{bnd}$ is one of $\models_{ult}$.

[8] The formalism of Pelov *et al.* (2007) is much richer including aggregate atoms under negation, and its semantics requires lower- and upper-regular TSRs defined inductively in terms of each other.

Liu et al. (2010). They introduced a kind of *TSR* to define semantics for abstract constraints. While the restrictions imposed on these relations are different and do not necessarily fit into AFT, their main example, the sub-satisfiability relation as proposed by Son *et al.* (2007) does. For any abstract constraint α: $(I, J) \models_{LPST} \alpha$ if and only if for each interpretation Z such that $I \subseteq Z \subseteq J$, it holds that $Z \models_2 \alpha$.

Definition 10 ($\models_{LPST}$)
The *TSR* $\models_{LPST}$ extends $\models_{GL}$ with: If a^{Aggr} is an aggregate atom, then $(I, J) \models_{LPST} a^{Aggr}$ iff for each Z such that $I \subseteq Z \subseteq J$: $Z \models_2 a^{Aggr}$.

This is the same satisfaction relation as $\models_{ult}$ in Definition 8, hence it is lower-regular and defines the same answer sets.

Gelfond and Zhang (2019). They construct a reduct for aggregate programs with respect to a three-valued interpretation. We only consider the case where the interpretation is two-valued. Gelfond and Zhang (2019) allow two kinds of negation: *negation by default*, which corresponds to negation as presented in this paper, and *explicit negation*, which is not a part of the syntax of the programs considered here but can be simulated by the well-known translation of explicitly negated atoms of a predicate p into atoms of a newly introduced predicate p^*. Since Gelfond and Zhang (2019) do not allow default negation within an aggregate atom, here it suffices to consider programs with positive conditions inside an aggregate atom. The reduction process is split into two main parts. The first part constructs a reduct regarding the aggregate atoms. It consists of two steps:

1. Removing all rules with aggregate atoms that evaluate to **f** in the interpretation.
2. Replacing every remaining aggregate atom by the conjunction of the subset of its conditions that are **t** in the interpretation.

In other words, given a rule r in a program P: $p \leftarrow l_1 \wedge ... \wedge l_n$, such that for an l_i it holds that $l_i = Agg(\{a_1 : cond_1, ..., a_n : cond_n\}) * w$, then the rule is deleted in the reduct P^J if l_i evaluates to **f** in J. Otherwise the rule is replaced by $p \leftarrow l_1 \wedge ... \wedge l_{i-1} \wedge l_{i+1} \wedge ... \wedge l_n \wedge (\bigwedge\{cond_j \in \{cond_1, \ldots, cond_n\} | J \models_2 cond_j\}$.

The second part transforms the preliminary reduct after the first phase to its Gelfond-Lifschitz reduct. In this way, it preserves the capability to deal with ordinary propositional atoms. From this reduct one can inductively define the *TSR* $\models_{GZ}$:

Definition 11 ($\models_{GZ}$)
$\models_{GZ}$ extends $\models_{GL}$ with: Let $a^{Aggr} = Agg(\{a_1 : cond_1, ..., a_n : cond_n\}) * w$. $(I, J) \models_{GZ} a^{Aggr}$ iff $J \models_2 a^{Aggr}$ and $(I, J) \models_{GZ} \bigwedge\{cond_j \in \{cond_1, \ldots, cond_n\} | J \models_2 cond_j\}$.

Proposition 8
For aggregate programs containing only positive conditions in aggregate atoms, the *TSR* $\models_{GZ}$ is identical to the *TSR* $\models_{triv}$ and lower-regular for consistent pairs, that is, with (I, J) a consistent pair, $(I, J) \models_{GZ} a^{Aggr}$ iff $(I, J) \models_{triv} a^{Aggr}$.

Precision Complexity Trade-off. One expects more effort gives more precise approximations. From Theorem 7.4 in the paper by Pelov *et al.* (2007), it follows that if the evaluation of an expression with respect to a lower-regular ternary satisfaction relation $\models_3$ is polynomially computable, then checking whether or not a model is an answer set is in P and deciding whether an answer set for a program exists, is in NP. This is the case for $\models_{triv}$ and $\models_{bnd}$ and for instances of the framework that coincide with $\models_{triv}$; for instances that coincide with $\models_{ult}$ the check problem is in NP and the exists-problem is in Σ_2^P.

5.2 Other ternary satisfaction relations

In the AFT framework, semantics of ASP aggregate atoms are based on lower regular *TSRs*. As we show in this section, also other well-known semantics can be characterized using *TSRs*, however, they are not regular. This is no coincidence. The definition of answer set semantics in terms of *TSRs* strongly resembles another well-known semantic method of ASP, namely using the logic of here-and-there (HT) (for an overview of HT and its applications to ASP, see the work by Cabalar *et al.* 2017). Due to very different points of view on answer sets, the two frameworks obtain different requirements for the *TSRs*. AFT treats answer sets as the result of constructive processes; the rule operator serves to produce them. A production is safe if the *TSR* is lower-regular. In contrast, HT takes a non-constructive take on answer sets. In HT, extensions build on the inherently non-regular *TSRs* derived from the three-valued logic *G3* introduced by Gödel (1932) where the rule operator is treated as HT-material implication.

Marek and Remmel (2004). They study Set Constraints (SC) Programming. It builds an *NSS-reduct* for an SC-program. Liu *et al.* (2010) prove that this semantics for set constraints is also obtained by the following satisfaction rule for a set constraint α: $(I, J) \models_{MR} \alpha$, if $J \models_2 \alpha$ and there exists an interpretation $Z \subseteq I$ such that $Z \models_2 \alpha$. This leads to the *TSR* $\models_{MR}$:

Definition 12 ($\models_{MR}$)
$\models_{MR}$ extends $\models_{GL}$ with: $(I, J) \models_{MR} a^{Aggr}$ iff $J \models_2 a^{Aggr}$ and there exists an interpretation Z such that $Z \subseteq I$ and $Z \models_2 a^{Aggr}$.

Consider the aggregate atom $SUM(\{1 : p, -1 : q\}) \geq 0$ and two intervals, namely $(\emptyset, \{p, q, s\}) <_p (\emptyset, \{q\})$. We have that $(\emptyset, \{p, q, s\}) \models_{MR} SUM(\{1 : p, -1 : q\}) \geq 0$ while $(\{p\}, \{q\}) \not\models_{MR} SUM(\{1 : p, -1 : q\}) \geq 0$. Hence, $\models_{MR}$ is not lower-regular.

Proposition 9
(i) $\models_{MR}$ extends $\models_2$, that is, $(I, I) \models_{MR} \psi$ iff $I \models_2 \psi$. (ii) $\models_{MR}$ is lower-monotone, that is, if $I \subseteq I'$ and $(I, J) \models_{MR} \psi$, then $(I', J) \models_{MR} \psi$.

While non-regular, the ternary relation $\models_{MR}$ still extends the satisfaction relation $\models_2$ and induces a monotone operator $\lambda I : A_{\models_{MR}}(I, J)$. Thus, an answer set may still be defined as an interpretation J that is a least fixpoint of this operator. But, due to non-regularity, some unexpected answer sets are obtained.

Example 1
Take the program:

$$s \leftarrow SUM(\{1 : p, -1 : q\}) \geq 0.$$
$$q \leftarrow SUM(\{1 : s\}) > 0. \qquad p \leftarrow SUM(\{1 : q\}) > 0.$$

https://doi.org/10.1017/S1471068422000126 Published online by Cambridge University Press

The bodies of these rules are equivalent to $p \vee \neg q$, respectively s and q. Therefore, one expects its answer sets to be the same as those of the simplified program:

$$s \leftarrow p. \qquad s \leftarrow \neg q. \qquad q \leftarrow s. \qquad p \leftarrow q.$$

To check that $J = \{p, q, s\}$ is an answer set, we observe that $(\emptyset, J) \models_{MR} SUM(\{1 : p, -1 : q\}) \geq 0$, hence the first iteration derives the head s. Two more iterations reconstruct the fixpoint $\{p, q, s\}$ which therefore is an answer set according to these semantics.

On the other hand, the Gelfond-Lifschitz reduct of the simplified program with respect to $J = \{p, q, s\}$ is:

$$s \leftarrow p. \qquad q \leftarrow s. \qquad p \leftarrow q.$$

Since $\emptyset$ is a model of the reduct, $\{p, q, s\}$ is not an answer set of the simplified program. The culprit for "too many" answer sets of the aggregate program is the non-regularity of $\models_{MR}$ which leads to the *unsafe* derivation of aggregate atoms: in the first derivation, $(\emptyset, \{p, q, s\}) \models_{MR} SUM(\{1 : p, -1 : q\}) \geq 0$ which derived s, but this derivation was *unsafe* since this aggregate atom is not satisfied in the more precise $(\emptyset, \{q\})$. However, things change when looking only at convex aggregate atoms.

Definition 13 (*Convex aggregate atom*)
An aggregate atom a^{Aggr} is convex iff for all interpretations I, Z, J, such that $I \subseteq Z \subseteq J$, it holds that if $I \models_2 a^{Aggr}$ and $J \models_2 a^{Aggr}$, then $Z \models_2 a^{Aggr}$.

Proposition 10
For convex aggregate atoms, $\models_{MR}$ behaves lower-regular and equivalent with $\models_{ult}$.

Faber et al. (2011). They provide semantics for a broader class of ASP programs including negated aggregate atoms. This approach constructs a reduct P^J for a program P with respect to an interpretation J by deleting all rules of which the body is not satisfied in J. In general, the immediate consequence operator T_{P^J} is not monotone, so answer sets cannot be defined using the *lfp* construction of this operator. Nevertheless, the *FPL*-answer sets are the answer sets of the following *TSR*:

Definition 14 ($\models_{FPL}$)
We define $\models_{FPL}$ as follows: $(I, J) \models_{FPL} \psi$ iff $I \models_2 \psi$ and $J \models_2 \psi$.

Proposition 11
$\models_{FPL}$ extends $\models_2$, that is, $(I, I) \models_{FPL} \psi$ iff $I \models_2 \psi$. For conjunctions of aggregate free literals, $\models_{FPL}$ coincides with $\models_{GL}$.

This very simple *TSR* is neither lower-regular nor lower-monotone, thus the constructive test is inapplicable.

As a consequence, discrepancies between some aggregate programs and their aggregate-free simplification are also present. Using again Example 1, $(\emptyset, \{p, q, s\}) \models_{FPL} SUM(\{1 : p, -1 : q\}) \geq 0$. But again, $(\emptyset, \{q\}) \not\models_{FPL} SUM(\{1 : p, -1 : q\}) \geq 0$. Similarly to $\models_{MR}$, the *FPL*-semantics leads to $\{p, q, s\}$ as an answer set. It is obvious that if $(I, J) \models_{FPL} \psi$, then $(I, J) \models_{MR} \psi$, since $I \subseteq I$. Accordingly $\models_{FPL}$ is less precise then $\models_{MR}$. Analogously, if $(I, J) \models_{ult} \psi$, then $(I, J) \models_{FPL} \psi$ since the interpretations I and J are both elements of $[I, J]$.

Table 1. *A summary of the different ternary satisfaction relations*

	LPST	*GZ*[9]	*MR*	*FPL*	*F*
Regular	arbitrary	arbitrary	convex	convex	anti-monotone
PDB	ult	triv	ult	ult	ult
lfp	arbitrary	arbitrary	arbitrary	convex	convex
$<_p$	$\{MR, FPL, F\}$	$\{bnd, LPST, MR, FPL, F\}$	$\emptyset$	$\{MR\}$	$\emptyset$

Proposition 12
For convex aggregate atoms, the *TSR* $\models_{FPL}$ behaves lower-regular and equivalent to $\models_{ult}$.

Ferraris (2011). This semantics is closely related to the *FPL*-semantics. Actually, they coincide for negation-free programs. Ferraris (2011) also cover more extensive instances of answer set programs, such as arbitrary propositional theories. It obtains the reduct P^J for a program P and interpretation J by replacing all maximal subformulas in P that are unsatisfied in J by $\bot$. This corresponds to the *TSR* $\models_F$:

Definition 15 ($\models_F$)
$\models_F$ extends $\models_{GL}$ with: Let $a^{Aggr} = Agg(\{a_1 : cond_1 \dots, a_n : cond_n\}) * w$, $(I, J) \models_F a^{Aggr}$ iff $J \models_2 a^{Aggr}$ and $I \models_2 Agg(\{a_i : cond_i \in \{a_1 : cond_1, \dots, a_n : cond_n\} | J \models_2 cond_i\}) * w$.

Since $\models_F$ and $\models_{FPL}$ coincide for negation-free programs, an analogous discussion of Example 1 leads to the conclusion that the satisfaction relation $\models_F$ is not lower-regular due to the lack of monotonicity.

Proposition 13
$\models_F$ extends $\models_2$, that is, $(I, I) \models_F \psi$ iff $I \models_2 \psi$.

Proposition 14
For convex aggregate atoms, $\models_F$ behaves lower-monotone, that is, if $I \subseteq I'$ and $(I, J) \models_F \psi$, then $(I', J) \models_F \psi$.

Proposition 15
For anti-monotone aggregate atoms, $\models_F$ behaves lower-regular and equivalent with $\models_{ult}$.

Table 1 gives an overview. Row 1 indicates for which aggregate atoms the ternary satisfaction relation behaves lower-regular. Row 2 shows which semantics from Pelov *et al.* (2007) coincides with this semantics for the subclass of programs such that the semantics behaves lower-regular. Row 3 indicates for which aggregate atoms the semantics is monotone in the first component such that the answer sets can be constructed as the *lfp* of Φ_P. Row 4 gives the set S_a of semantics discussed in this paper that are strictly more precise than the considered instance a. Consequently every a-answer will always

[9] Note that the *GZ*-semantics is not defined for programs with conditions under default negation within aggregate-atoms and the table should be interpreted accordingly.

https://doi.org/10.1017/S1471068422000126 Published online by Cambridge University Press

be an answer set for every semantics in S_a. This does not generally hold the other way around. Interestingly, all non-regular semantics coincide with the ult-semantics of Pelov (2004) for the subclass of programs in which they behave lower-regular.[10] For programs outside this subclass, the non-regular semantics differ and may derive more answer sets than the ult-semantics.

6 Conclusions and future work

Approximation Fixpoint Theory describes various types of constructions from nonmonotonic operators and was designed to formalize the view of Logic Programming as constructive definitions, a view at least implicit in stratified logic programs, in the KK and WF semantics, but also in the logic FO(ID) introduced by Denecker and Ternovska (2008). We studied aggregate programs from this view point, showing how regular (3- or 4-valued) extensions of the strong Kleene truth assignment induce extensions of KK, WF and stable semantics. We showed that regular truth assignments correspond one-to-one to pairs of a lower-regular and an upper-regular ternary satisfaction relation $(I, J) \models_3 \psi$, where the lower-regular one suffices for defining stable models. To study the relation with ASP, we then made a generalized study of *TSR*s as a tool to define answer sets. We analysed different properties of *TSR*s, and many semantics of aggregate programs in the literature to determine the corresponding lower ternary satisfaction relation and the properties that influence them, such as convexity, (anti-)monotonicity, and the sign of conditions in aggregates. We obtained many results linking many ASP semantics in various degrees to the AFT-framework.

In the ASP community, other views of LP and ASP exist than that as a logic of constructive definitions. They are developed in various frameworks such as the framework of HT (for more details see the paper by Cabalar *et al.* 2017) or of Ferraris and Lifschitz (for more details see the paper by Ferraris *et al.* 2007). These frameworks often extend the original logic programming formalism in various directions, for example, with disjunction in the head, other negations. They may entirely redefine full FO and the meaning of its connectives. The base idea is that a program corresponds to a theory in some logic (e.g., HT or FO) from which answer set are derived using some equilibrium characterisation. Although these semantics are not constructive in the sense of AFT, there are surely many interesting mathematical relationships to AFT. For instance, it is striking that the logic of HT is also defined in terms of a ternary satisfaction relation. It is a goal for future work to investigate this.

Supplementary material

To view supplementary material for this article, please visit http://10.1017/S1471068422000126.

[10] Regularity is a property of a semantics. A program cannot be regular or non-regular. However, if a program belongs to a specific subclass of programs, a non-regular semantics may behave lower-regular anyway.

References

Belnap, N. D. 1977. A useful four-valued logic. In *Modern Uses of Multiple-Valued Logic*, J. M. Dunn and G. Epstein, Eds., 8–37.

Bogaerts, B. 2019. Weighted abstract dialectical frameworks through the lens of approximation fixpoint theory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2686–2693.

Cabalar, P., Pearce, D. and Valverde, A. 2017. Stable reasoning. *Journal of Applied Non-Classical Logics 27*, 3-4, 238–254. Taylor & Francis.

Charalambidis, A., Rondogiannis, P. and Symeonidou, I. 2018. Approximation fixpoint theory and the well-founded semantics of higher-order logic programs. *Theory and Practice of Logic Programming 18*, 3-4, 421–437.

Denecker, M., Bruynooghe, M. and Vennekens, J. 2012. Approximation fixpoint theory and the semantics of logic and answers set programs. In *Correct Reasoning - Essays on Logic-Based AI in Honour of Vladimir Lifschitz*. Springer, 178–194.

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

Denecker, M., Marek, V. W. and Truszczynski, M. 2003. Uniform semantic treatment of default and autoepistemic logics. *Artificial Intelligence 143*, 1, 79–122.

Denecker, M., Marek, V. W. and Truszczyński, M. 2004. Ultimate approximation and its application in nonmonotonic knowledge representation systems. *Information and Computation 192*, 1, 84–121. Elsevier.

Denecker, M., Pelov, N. and Bruynooghe, M. 2001. Ultimate well-founded and stable semantics for logic programs with aggregates. In *Logic Programming, 17th International Conference, ICLP 2001, Paphos, Cyprus, 26 November–1 December 2001, Proceedings*, P. Codognet, Ed., Lecture Notes in Computer Science, vol. 2237. Springer, 212–226.

Denecker, M. and Ternovska, E. 2008. A logic of nonmonotone inductive definitions. *ACM Transactions on Computational Logic (TOCL) 9*, 2, 1–52. ACM, New York, NY, USA.

Faber, W., Pfeifer, G. and Leone, N. 2011. Semantics and complexity of recursive aggregates in answer set programming. *Artificial Intelligence 175*, 1, 278–298. Elsevier.

Ferraris, P., Lee, J. and Lifschitz, V. 2007. A new perspective on stable models. *IJCAI 7*, 372–379.

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

Fitting, M. 1985. A Kripke-Kleene semantics for logic programs. *Journal of Logic Programming 2*, 4, 295–312.

Gebser, M., Harrison, A., Kaminski, R., Lifschitz, V. and Schaub, T. 2015. Abstract gringo. *Theory and Practice of Logic Programming 15*, 4–5, 449–463.

Gelfond, M. and Kahl, Y. 2014. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach*. Cambridge University Press.

Gelfond, M. and Lifschitz, V. 1988. The stable model semantics for logic programming. In *The Stable Model Semantics for Logic Programming*. ICLP/SLP, 1070–1080.

Gelfond, M. and Zhang, Y. 2019. Vicious circle principle, aggregates, and formation of sets in ASP based languages. *Artificial Intelligence 275*, 28–77. Elsevier.

Gödel, K. 1932. Zum intuitionistischen aussagenkalkül. *Anzeiger der Akademie der Wissenschaften in Wien 69*.

Heyninck, J. and Arieli, O. 2021. Approximation fixpoint Theory for non-deterministic operators and its application in disjunctive logic programming. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, vol. 18, 1, 334–344.

Kemp, D. B. and Stuckey, P. J. 1991. Semantics of logic programs with aggregates. In *Logic Programming, Proceedings of the 1991 International Symposium, San Diego, California, USA, 28 Oct.–1 Nov 1991*, V. A. Saraswat & K. Ueda, Eds. MIT Press, 387–401.

Kleene S. 1952. *Introduction to Metamathematics*. Van Nostrand, New York.

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. Elsevier.

Marek, V. W. and Remmel, J. B. 2004. Set constraints in logic programming. In *International Conference on Logic Programming and Nonmonotonic Reasoning*. Springer, 167–179.

Pelov, N. 2004. *Semantics of Logic Programs with Aggregates*. Ph.D. thesis, KU Leuven.

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. Cambridge University Press.

Son, T. C., Pontelli, E. and Tu, P. H. Answer sets for logic programs with arbitrary abstract constraint atoms. *Journal of Artificial Intelligence Research 29*, 353–389.

Strass, H. 2013. Approximating operators and semantics for abstract dialectical frameworks. *Artificial Intelligence 205*, 39–70.