HEX Semantics via Approximation Fixpoint Theory[⋆]

Christian Antić, Thomas Eiter, and Michael Fink

Institute of Information Systems, Vienna University of Technology
Favoritenstraße 9-11, A-1040 Vienna, Austria
{antic,eiter,fink}@kr.tuwien.ac.at

Abstract. Approximation Fixpoint Theory (AFT) is an algebraic framework for studying fixpoints of possibly nonmonotone lattice operators, and thus extends the fixpoint theory of Tarski and Knaster. In this paper, we uniformly define 2-, and 3-valued (ultimate) answer-set semantics, and well-founded semantics of disjunction-free HEX programs by applying AFT. In the case of disjunctive HEX programs, AFT is not directly applicable. However, we provide a definition of 2-valued (ultimate) answer-set semantics based on non-deterministic approximations and show that answer sets are minimal, supported, and derivable in terms of bottom-up computations. Finally, we extensively compare our semantics to closely related semantics, including constructive dl-program semantics. Since HEX programs are a generic formalism, our results are applicable to a wide range of formalisms.

1 Introduction

HEX programs [10] enrich disjunctive logic programs under the answer-set semantics [12] (ASP programs) by external atoms for software interoperability. As the latter can represent arbitrary computable Boolean functions, HEX programs constitute a powerful extension of ordinary logic programs that has been exploited in a range of applications.[1] Furthermore, they are closely related to other extensions of ASP programs, such as dl-programs (considered below), modular logic programs, or multi-context systems with ASP components (see [8]). The semantics of HEX programs has been defined in terms of FLP-answer sets, which adhere to minimal models or, even more restricting, to models free of unfoundedness. However, FLP-answer sets of HEX programs may permit circular justifications (cf. [18]), and concepts such as well-founded semantics (which is based on unfounded sets) [21] may be cumbersome to define.

Approximation Fixpoint Theory (AFT) [5,7] is an abstract algebraic framework for studying fixpoints of (monotone or nonmonotone) lattice operators in terms of (monotone) *approximations*. In this sense, AFT extends the well-known Tarski-Knaster fixpoint theory to arbitrary lattice operators, with applications in logic programming and non-monotonic reasoning [5,6,7,4,16]; in particular, the major semantics of normal logic programs [5] and of default and autoepistemic logic [6] can be elegantly characterized within the framework of AFT; whole families of 2- and 3-valued semantics can be obtained, where the underlying fixpoint iteration incorporates a notion of foundedness.

[⋆] This work was supported by the Austrian Science Fund (FWF) grant P24090.

[1] http://www.kr.tuwien.ac.at/research/systems/dlvhex/applications.html.

This suggests to use AFT for giving semantics to HEX programs targeted for foundedness, by defining suitable operators in such a way that existing semantics for HEX programs can be reconstructed or refined, in the sense that a subset of the respective answer sets are selected (sound "approximation"). The benefit is twofold: by coinciding semantics, we get operable fixpoint constructions, and by refined semantics, we obtain a sound approximation that is constructive. Furthermore, related semantics emanate in a systematic fashion rather than adhoc, and their relationships are understood at an abstract level and need not be established on an individual basis. Finally, semantics of related formalisms like those mentioned above might be defined in a similar way.

Motivated by this, we consider 2- and 3-valued semantics of HEX programs (the latter has not been considered so far), and provide respective notions of answer sets using AFT. In this way, we reobtain and refine existing semantics of HEX programs and dl-programs. We also consider disjunctive HEX programs (for which AFT is not directly applicable) and define 2-valued answer semantics following a method in [15].

The main contributions of this paper can be summarized as follows:

(1) We define the full class of 3-valued semantics [12,17,21] of *normal* (i.e., *disjunction-free*) HEX programs [10] in a uniform way by applying the AFT framework [5,7] (cf. Section 3). In particular, this class contains 2-, and 3-valued answer-set semantics [12,17], and well-founded semantics [21]. Moreover, we define *ultimate* versions which are the most precise approximation semantics with respect to AFT [7].

(2) We exhaustively compare our semantics with the FLP semantics in [10]. They coincide on *monotone* normal HEX programs, but diverge for arbitrary normal HEX programs: due to constructiveness, each 2-valued AFT answer set is an FLP-answer set but not vice versa (cf. Theorem 2). Also, each 2-valued answer set is *well-supported* [18] which is key to characterize relevant models (cf. Section 5), and thus free of circular justifications. Moreover, our 2-valued and Shen's strongly well-supported answer-set semantics coincide [18] (cf. Theorem 8). However, our AFT approach is more general.

(3) Combining ideas from AFT, logic programs with aggregates [15], and disjunctive logic programming [13], we introduce 2-valued (ultimate) answer sets for disjunctive HEX programs along the lines of [15]. To this end, we translate some of the concepts of AFT to *non-deterministic* operators and use the notion of *computation* [14] to iterate them bottom up; we show that all (ultimate) answer sets are derivable by computations. Furthermore, (ultimate) answer sets are supported models and each 2-valued answer set is an FLP-answer set but not vice vera.

(4) We exploit the results for *description logic (dl-)programs* [9], which can be viewed as special HEX programs whose external atoms amount to so-called *dl-atoms* representing queries to a description logic ontology. Initially, a strong and weak answer set semantics of dl-programs was defined and later a well-founded semantics for *monotone* dl-programs [9]; our results generalize it *a fortiori* to arbitrary dl-programs. It turns out that for *monotone* dl-programs, the semantics coincide for the Fitting approximation Φ_P^{HEX}; however, for general dl-programs, the answer set semantics diverges.

The results of this paper provide further evidence that AFT is a valuable tool to define and study semantics of LP extensions, with well-understood and desired properties. For space reason, proofs are omitted; they are available in [1], which also provides a more extensive discussion and contains additional results.

2 Preliminaries

2.1 HEX Programs

In this section, we recall HEX programs [10], where we restrict ourselves without loss of generality to the ground (variable-free) case.

Syntax. Let Σ and $\Sigma^{\#}$ be ranked alphabets of *symbols* and *external symbols*, respectively. Elements from $\Sigma^{\#}$ are superscripted with $\#$ ($f^{\#}, g^{\#}, h^{\#}$ etc.). [2]

In contrast to ordinary logic programs, HEX programs may contain besides *(ordinary) atoms* of the form $(p, c_1, \ldots, c_n) \in \Sigma^{n+1}$, $n \geq 0$, written in familiar form $p(c_1, \ldots, c_n)$, also so called *external atoms*. Formally, an *((m,n)-ary) external atom* has the form $f^{\#}[\mathbf{i}](\mathbf{o})$ where $f^{\#} \in \Sigma^{\#}$, $\mathbf{i} = (i_1 \ldots i_m) \in \Sigma^m$ *(=input)*, $m \geq 0$, and $\mathbf{o} = (o_1 \ldots o_n) \in \Sigma^n$ *(=output)*, $n \geq 0$. We often omit the arguments $\mathbf{i}$ and $\mathbf{o}$ from notation and simply write $f^{\#}$. A HEX-*atom* is an atom or an external atom. A *rule* has the form

$$a_1 \vee \ldots \vee a_k \leftarrow b_1, \ldots, b_\ell, \sim b_{\ell+1}, \ldots, \sim b_m, \quad k \geq 0, m \geq \ell \geq 0, \tag{1}$$

where $a_1, \ldots, a_k$ are atoms and $b_1, \ldots, b_m$ are HEX-atoms. It will be convenient to define, for a rule r, $H(r) = a_1 \vee \ldots \vee a_k$ *(head)*, $B^+(r) = \{b_1, \ldots, b_\ell\}$, $B^\sim(r) = \{\sim b_{\ell+1}, \ldots, \sim b_m\}$, and $B(r) = B^+(r) \cup B^\sim(r)$ *(body)*. With a slight abuse of notation, we will treat $H(r)$ as the set $\{a_1, \ldots, a_k\}$, i.e., we write, for instance, $a \in H(r)$, $H(r) - \{a\}$ and so on. Finally, a HEX *program* P is a *finite* set of rules of form (1).

FLP Semantics. We denote the set of all atoms (resp., external atoms) occurring in P by At_P (resp., $At_P^{\#}$). Define the *Herbrand base* of P by $HB_P = At_P \cup At_P^{\#}$. A *(2-valued) interpretation* I of P is any subset of At_P; for any $p \in \Sigma$, we denote by $p^I = \{\mathbf{c} : p(\mathbf{c}) \in I\}$ its *extension* in I. The set of all interpretations of P is $\mathcal{I}_P = \mathfrak{P}(At_P)$. We associate with every $f^{\#} \in \Sigma^{\#}$ a computable interpretation function $f : \mathcal{I}_P \times \Sigma_P^{m+n} \to \{\mathbf{t}, \mathbf{f}\}$.[2]

We define the entailment relation as follows: (i) For an atom a, $I \models a$ if $a \in I$, (ii) for an external atom $f^{\#}[\mathbf{i}](\mathbf{o})$, $I \models f^{\#}[\mathbf{i}](\mathbf{o})$ if $f(I, \mathbf{i}, \mathbf{o}) = \mathbf{t}$, (iii) for a rule r, $I \models B(r)$ if $I \models b$ for every $b \in B^+(r)$ and $I \not\models b'$ for every $\sim b' \in B^\sim(r)$, (iv) $I \models r$ if whenever $I \models B(r)$ then $I \models a$ for some $a \in H(r)$, and (v) $I \models P$ if $I \models r$ for each $r \in P$, and in this case we say that I is a *model* of P.

Define the *FLP-reduct of* P relative to I [11] by $fP^I = \{r \in P : I \models B(r)\}$. We say that I is an *FLP-answer set* [10] of P if I is a minimal model of fP^I.

Example 1. Let $P = \{q(a);\ p(a) \leftarrow p \subseteq^{\#} q, q(a)\}$, where $p \subseteq^{\#} q$ is infix notation for $\subseteq^{\#}[p, q]$, and let $I = \{p(a), q(a)\}$. We interpret $\subseteq^{\#}$ as set inclusion and define $\subseteq (I, p, q) = \mathbf{t}$ if $p^I \subseteq q^I$ where $p^I = q^I = \{a\}$. Since $fP^I = P$ and I is a minimal model of P, I is an FLP-answer set of P.

2.2 Approximation Fixpoint Theory

In this section, we briefly summarize essential notions and results given in [5,7].

In the sequel, we let $(L, \leq)$ denote a complete lattice. We call every $(x_1, x_2) \in L^2$ fulfilling $x_1 \leq x_2$ *consistent* and denote by L^c the set of all such pairs.

[2] [10] uses $\#g$ and $f_{\#g}$ in place of $g^{\#}$ and g, respectively, and calls symbols constants.

Define the *precision ordering* on L^c by $(x_1, x_2) \leq_p (y_1, y_2)$ if $x_1 \leq y_1$ and $y_2 \leq x_2$, i.e. intuitively, (y_1, y_2) is a "tighter" interval inside (x_1, x_2). We identify every $x \in L$ with $(x, x) \in L^c$ and call such pairs *exact*; note that they are the maximal elements w.r.t. $\leq_p$. For $z \in L$ and $(x_1, x_2) \in L^c$, we thus have $(x_1, x_2) \leq_p z$ iff $x_1 \leq z \leq x_2$, and call (x_1, x_2) an *approximation* of z. As distinct exact pairs have no upper bound, $(L^c, \leq_p)$ is not a lattice but a chain-complete poset.

An *operator* on L is any function $O : L \to L$; it is *monotone*, if for every $x, y \in L$ such that $x \leq y$, $O(x) \leq O(y)$. An element $x \in L$ is a *pre-fixpoint of* O, if $O(x) \leq x$, and a *fixpoint of* O, if $O(x) = x$. If existent, we denote the *least fixpoint of* O by $\mathrm{lfp}(O)$.

We now define approximations of O which will play a central role throughout the rest of the paper. We say that an operator $\mathcal{A} : L^c \to L^c$ is an *approximation* of O, if (i) $\mathcal{A}(x, x) = O(x)$ for each $x \in L$, and (ii) $\mathcal{A}$ is monotone with respect to $\leq_p$. Intuitively, $\mathcal{A}$ is a monotone extension of O to L^c. Clearly, $\mathcal{A}$ and O have the same fixpoints in L. Moreover, $\mathcal{A}$ has a least fixpoint $k(\mathcal{A})$, called the *$\mathcal{A}$-Kripke-Kleene fixpoint*.

For every $x, y \in L$, define the *interval* between x and y by $[x, y] = \{z \in L : x \leq z \leq y\}$. Given an element $(x_1, x_2) \in L^c$, we define the *projection* of (x_1, x_2) to the i-th coordinate, $1 \leq i \leq 2$, by $(x_1, x_2)_i = x_i$. We call an element $(x_1, x_2) \in L^c$ *$\mathcal{A}$-reliable*, if $(x_1, x_2) \leq_p \mathcal{A}(x_1, x_2)$, and in this case the restriction of $\mathcal{A}(\,.\,, x_2)_1$ (resp., $\mathcal{A}(x_1, \,.\,)_2$) to $[\bot, x_2]$ (resp., $[x_1, \top]$) is a monotone operator on the complete lattice $([\bot, x_2], \leq)$ (resp., $([x_1, \top], \leq)$). Therefore, $\mathcal{A}(\,.\,, x_2)_1$ (resp., $\mathcal{A}(x_1, \,.\,)_2$) has a least fixpoint in $([\bot, x_2], \leq)$ (resp., $([x_1, \top], \leq)$).

Let (x_1, x_2) be $\mathcal{A}$-reliable. Define the *$\mathcal{A}$-stable revision operator* by $\mathcal{A}^{\downarrow\uparrow}(x_1, x_2) = (\mathcal{A}^{\downarrow}(x_2), \mathcal{A}^{\uparrow}(x_1))$, where $\mathcal{A}^{\downarrow}(x_2) = \mathrm{lfp}(\mathcal{A}(\,.\,, x_2)_1)$ and $\mathcal{A}^{\uparrow}(x_1) = \mathrm{lfp}(\mathcal{A}(x_1, \,.\,)_2)$. Roughly, $\mathcal{A}^{\downarrow}(x_2)$ underestimates every (minimal) fixpoint of O, whereas $\mathcal{A}^{\uparrow}(x_1)$ is an upper bound as tight as possible to the minimal fixpoints of O. The stable revision operator $\mathcal{A}^{\downarrow\uparrow}$ has fixpoints and a least fixpoint on the chain-complete poset $(L^{\mathcal{A}}_{pr}, \leq_p)$ of the so called *$\mathcal{A}$-prudent* pairs $L^{\mathcal{A}}_{pr} = \{(x_1, x_2) \in L^c \mid x_1 \leq \mathcal{A}^{\downarrow}(x_2)\}$. We thus define the *$\mathcal{A}$-well-founded fixpoint* by $w(\mathcal{A}) = \mathrm{lfp}(\mathcal{A}^{\downarrow\uparrow})$. Furthermore, we call every $\mathcal{A}$-reliable fixpoint (x_1, x_2) of $\mathcal{A}^{\downarrow\uparrow}$ an *$\mathcal{A}$-stable fixpoint*, and if in addition $\mathcal{A}$ is an approximation of O and $x_1 = x_2$ an *$\mathcal{A}$-stable fixpoint of O* (note that x_1 is then a fixpoint of O).

Proposition 1 ([7]). *For every $x \in L$, x is an $\mathcal{A}$-stable fixpoint of O iff x is a fixpoint of O and $\mathcal{A}^{\downarrow}(x) = x$.*

In [7] the authors show that there exists a most precise approximation $\mathcal{O}$, called the *ultimate approximation* of O, which can be algebraically characterized in terms of O. Let for $C \in \{\bigwedge, \bigvee\}$ denote $C\, O([x_1, x_2]) = C\, \{O(x) \mid x_1 \leq x \leq x_2\}$.

Theorem 1 ([7]). *The ultimate approximation of O is given, for every $(x_1, x_2) \in L^c$, by $\mathcal{O}(x_1, x_2) = (\bigwedge O([x_1, x_2]), \bigvee O([x_1, x_2]))$.*

We summarize some basic facts about the ultimate approximation and its relationship to every other approximation of O.

Proposition 2 ([7]). *For every approximation $\mathcal{A}$ of O:*

1. *$k(\mathcal{A}) \leq_p w(\mathcal{O})$ and $w(\mathcal{A}) \leq_p w(\mathcal{O})$.*

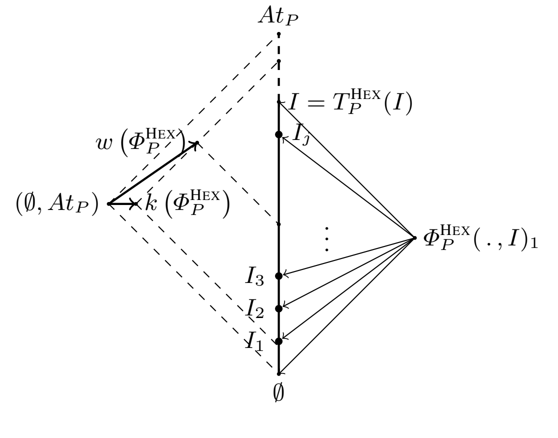
At_P
$I = T_P^{\text{HEX}}(I)$
I_j
$w\left(\Phi_P^{\text{HEX}}\right)$
$(\emptyset, At_P)$
$k\left(\Phi_P^{\text{HEX}}\right)$
$\Phi_P^{\text{HEX}}(\,.\,, I)_1$
I_3
I_2
I_1
$\emptyset$

Fig. 1. Illustration of the relations between the Φ_P^{HEX}-Kripke-Kleene-, the Φ_P^{HEX}-well-founded, and the 2-valued Φ_P^{HEX}-answer-set semantics. On the left side: (i) the *Kripe-Kleene fixpoint* $k\left(\Phi_P^{\text{HEX}}\right)$ is the least fixpoint of Φ_P^{HEX}; (ii) the *well-founded fixpoint (least 3-valued stable fixpoint)* $w\left(\Phi_P^{\text{HEX}}\right)$ is the least fixpoint of $\Phi_P^{\text{HEX},\downarrow\uparrow}$. On the right side: monotone iteration of the *2-valued* Φ_P^{HEX}*-stable model* I. If we replace Φ_P^{HEX} by $\mathcal{T}_P^{\text{HEX}}$, we obtain the more precise *ultimate* semantics.

2. *If $k(\mathcal{A})$ (resp., $w(\mathcal{A})$) is exact, then $k(\mathcal{A}) = k(\mathcal{O})$ (resp., $w(\mathcal{A}) = w(\mathcal{O})$) and it is the unique ultimate stable fixpoint of O.*
3. *Every $\mathcal{A}$-stable fixpoint of O is an ultimate stable fixpoint of O and for every ultimate fixpoint x of O, $w(\mathcal{O}) \leq_p x$.*

3 Fixpoint Semantics for Normal HEX Programs

In this section, we uniformly extend the 2- and 3-valued answer-set semantics [12,17] and the well-founded semantics [21] of ordinary logic program to the class of normal (i.e., disjunction-free) HEX programs by applying AFT (for a summary, see Figure 1).

In the sequel, let P be a normal HEX program. We can straightforwardly extend the well-known van Emden-Kowalski operator T_P to T_P^{HEX}. A *(consistent) 3-valued interpretation* is a pair (I_1, I_2) such that $I_1 \subseteq I_2$; by $\mathcal{I}_P^c$ we denote the set of all such pairs. The precision ordering $\subseteq_p$ on $\mathcal{I}_P^c$ is given by $(J_1, J_2) \subseteq_p (I_1, I_2)$ if $J_1 \subseteq I_1$ and $I_2 \subseteq J_2$ (cf. Section 2.2). The intuitive meaning is that every $a \in I_1$ (resp., $a \notin I_2$) is *true* (resp., *false*), whereas every $a \in I_2 - I_1$ is *undefined*.

Definition 1. *We identify each $(I_1, I_2) \in \mathcal{I}_P^c$ with the 3-valued evaluation $(I_1, I_2) : HB_P \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ defined by:*

1. *For every $a \in At_P$, $(I_1, I_2)(a) = \mathbf{t}$ if $a \in I_1$, $(I_1, I_2)(a) = \mathbf{f}$ if $a \notin I_2$, and $(I_1, I_2)(a) = \mathbf{u}$ if $a \in I_2 - I_1$.*
2. *For every $f^\# \in At_P^\#$, $(I_1, I_2)(f^\#) = \mathbf{t}$ (resp., $(I_1, I_2)(f^\#) = \mathbf{f}$) if $J \models f^\#$ (resp., $J \not\models f^\#$) for every $J \in [I_1, I_2]$, and $(I_1, I_2)(f^\#) = \mathbf{u}$ otherwise.*

We then directly obtain the following two approximations Φ_P^{HEX} and $\mathcal{T}_P^{\text{HEX}}$ of T_P^{HEX}: (i) The *extended Fitting approximation* Φ_P^{HEX} as an extension of the traditional Fitting operator Φ_P, i.e., $\Phi_P(I_1, I_2) = (I_1', I_2')$ where $I_1' = \{H(r) : r \in P : (I_1, I_2)(B(r)) = \mathbf{t}\}$ and

$I_2' = \{H(r) : r \in P : (I_1, I_2)(B(r)) \in \{\mathbf{t}, \mathbf{u}\}\}$ (given the usual extension of 3-valued interpretation to conjunctions of literals—denoted as sets $B(r)$ here); and (ii) the *ultimate approximation* $\mathcal{T}_P^{\text{HEX}}$, which is the most precise approximation of T_P^{HEX} and algebraically definable by (cf. Theorem 1) $\mathcal{T}_P^{\text{HEX}}(I_1, I_2) = \left(\bigcap T_P^{\text{HEX}}([I_1, I_2]), \bigcup T_P^{\text{HEX}}([I_1, I_2])\right)$.

The approximations Φ_P^{HEX} and $\mathcal{T}_P^{\text{HEX}}$ give rise to the Φ_P^{HEX}*-Kripke-Kleene semantics* and the *ultimate Kripke-Kleene semantics*, respectively (cf. Section 2.2).

3.1 Answer-Set Semantics

Recall that given an ordinary normal program P, its 3-valued answer sets are characterized by the fixpoints of the stable revision operator $\Phi_P^{\downarrow\uparrow}$ of the Fitting approximation Φ_P. Moreover, the 2-valued answer sets of P are the fixpoints of $\Phi_P^{\downarrow}$ (cf. [5]).[3]

Likewise, in this section, we extend these definitions to normal HEX programs. In the sequel, let $\mathcal{A}_P^{\text{HEX}} \in \left\{\Phi_P^{\text{HEX}}, \mathcal{T}_P^{\text{HEX}}\right\}$. By instantiating the definition in Section 2.2, we define the stable revision operator of $\mathcal{A}_P^{\text{HEX}}$, for every $\mathcal{A}_P^{\text{HEX}}$-reliable $(I_1, I_2) \in \mathcal{I}_P^c$, by $\mathcal{A}_P^{\text{HEX},\downarrow\uparrow}(I_1, I_2) = (\mathcal{A}_P^{\text{HEX},\downarrow}(I_2), \mathcal{A}_P^{\text{HEX},\uparrow}(I_1))$, where $\mathcal{A}_P^{\text{HEX},\downarrow}(I_2) = \text{lfp}(\mathcal{A}_P^{\text{HEX}}(\,.\,, I_2)_1)$ and $\mathcal{A}_P^{\text{HEX},\uparrow}(I_1) = \text{lfp}(\mathcal{A}_P^{\text{HEX}}(I_1,\,.\,)_2)$.

Definition 2 (Answer-Set Semantics). *Let $(I_1, I_2) \in \mathcal{I}_P^c$ be $\mathcal{A}_P^{\text{HEX}}$-reliable, respectively let $I \in \mathcal{I}_P$. We say that*

1. *(I_1, I_2) is a* 3-valued Φ_P^{HEX}-answer set *(resp.,* 3-valued ultimate answer set*) of P, if (I_1, I_2) is a fixpoint of $\Phi_P^{\text{HEX},\downarrow\uparrow}$ (resp., $\mathcal{T}_P^{\text{HEX},\downarrow\uparrow}$).*
2. *I is a* 2-valued Φ_P^{HEX}-answer set *(resp.,* 2-valued ultimate answer set*) of P, if I is a fixpoint of T_P^{HEX} and $\Phi_P^{\text{HEX},\downarrow}(I) = I$ (resp., $\mathcal{T}_P^{\text{HEX},\downarrow}(I) = I$).*

Example 2 (Example 1 cont'd). We claim that $I = \{p(a), q(a)\}$ is a 2-valued Φ_P^{HEX}-answer set (and, therefore, an ultimate answer set) of P. First, observe that $T_P^{\text{HEX}}(I) = I$. Second, since $\Phi_P^{\text{HEX}}(\emptyset, I)_1 = \{q(a)\}$, $\Phi_P^{\text{HEX}}(\{q(a)\}, I)_1 = I$, and $\Phi_P^{\text{HEX}}(I, I)_1 = I$, we have $\Phi_P^{\text{HEX},\downarrow}(I) = I$. Hence, I is a 2-valued Φ_P^{HEX}-answer set.

The next Theorem and Example 3 summarize some basic relationships between the standard FLP semantics and the 2-valued Φ_P^{HEX}-answer-set semantics.

Theorem 2. *Let P be a normal* HEX *program.*

1. *If P is monotone (i.e., contains only monotone external atoms[4]) and negation-free, then $I \in \mathcal{I}_P$ is an Φ_P^{HEX}-answer set iff I is an FLP-answer set of P.*
2. *If $I \in \mathcal{I}_P$ is an Φ_P^{HEX}-answer set, then I is an FLP-answer set of P.*

However, the next example shows that the converse of condition (2) fails in general.

Example 3. Let $P = \left\{a \leftarrow f^{\#}[a, b];\ b \leftarrow g^{\#}[a, b]\right\}$ where f and g are always *true* except for $f(\{a\}, a, b) = \mathbf{f}$ and $g(\{b\}, a, b) = \mathbf{f}$. It is easy to verify that $I = \{a, b\}$ is a minimal model of $fP^I = P$ and, hence, an FLP-answer set of P. In contrast, we have $\mathcal{T}_P^{\text{HEX},\downarrow}(I) = \emptyset$. Consequently, by Proposition 1, I is not an ultimate answer set of P and, hence, by Proposition 2, not an Φ_P^{HEX}-answer set.

[3] Note that $\Phi_P^{\downarrow}(I) = \text{lfp}(T_{P^I})$ where P^I is the Gelfond-Lifschitz reduct [12].

[4] We say that an external atom $f^{\#}[\mathbf{i}](\mathbf{o})$ is *monotone*, if for every $J, J' \in \mathcal{I}_P$ such that $J \subseteq J'$, $f(J, \mathbf{i}, \mathbf{o}) \le f(J', \mathbf{i}, \mathbf{o})$, where $\mathbf{f} < \mathbf{t}$.

Intuitively, the divergence of the 2-valued answer-set semantics based on AFT and the FLP semantics is due to the "non-constructiveness" of the FLP semantics. The intuition behind "constructiveness" is formalized by Fages' *well-supportedness* (adapted to HEX by Shen [18]). Indeed, Theorems 2 and 8 characterize our (2-valued) $\Phi_P^{\textsc{Hex}}$-answer-set semantics as the strict well-supported subclass of the FLP semantics (cf. the discussion in Section 5). While incomparability of ultimate and FLP semantics already follows from the ordinary case [7].

3.2 Well-Founded Semantics

Well-founded semantics play an important role in logic programming and database theory. However, for (normal) HEX programs, to the best of our knowledge, there exist no well-founded semantics up so far. In this section, we define well-founded semantics of normal HEX programs as a special case of 3-valued $\Phi_P^{\textsc{Hex}}$-answer-set semantics, by instantiating the constructions of AFT given in Section 2.2.

Recall from Section 2.2 that every stable revision operator has fixpoints and a least fixpoint. This leads to the following definition.

Definition 3 (Well-Founded Semantics). *Define the* $\mathcal{A}_P^{\textsc{Hex}}$-well-founded model *by* $w\left(\mathcal{A}_P^{\textsc{Hex}}\right) = \mathrm{lfp}(\mathcal{A}_P^{\textsc{Hex},\downarrow\uparrow})$. *We call the* $\mathcal{T}_P^{\textsc{Hex}}$*-well-founded model* $w\left(\mathcal{T}_P^{\textsc{Hex}}\right)$ *the* ultimate well-founded model *of* P.

Example 4. Reconsider the normal HEX program P of Example 3 where we have seen that $I = \{a, b\}$ is an FLP-answer set but not an ultimate answer set of P. Since $\mathcal{T}_P^{\textsc{Hex},\downarrow\uparrow}(\emptyset, \{a, b\}) = (\emptyset, \{a, b\})$, $w\left(\mathcal{T}_P^{\textsc{Hex}}\right) = (\emptyset, \{a, b\}) = w\left(\Phi_P^{\textsc{Hex}}\right)$, i.e., a and b are both *undefined* in the (ultimate) well-founded model.

We can compute $w\left(\mathcal{A}_P^{\textsc{Hex}}\right)$ by iterating $\mathcal{A}_P^{\textsc{Hex},\downarrow\uparrow}$, starting at $(\emptyset, At_P)$, until a fixpoint is reached. The $\mathcal{A}_P^{\textsc{Hex}}$-well-founded model is the least 3-valued $\mathcal{A}_P^{\textsc{Hex}}$-answer set and approximates every other 3-valued $\mathcal{A}_P^{\textsc{Hex}}$-answer set, i.e., $w\left(\mathcal{A}_P^{\textsc{Hex}}\right) \subseteq_p (I_1, I_2)$ for every $\mathcal{A}_P^{\textsc{Hex}}$-answer set $(I_1, I_2) \in \mathcal{I}_P^c$. In particular, $w\left(\mathcal{A}_P^{\textsc{Hex}}\right)$ approximates every 2-valued $\mathcal{A}_P^{\textsc{Hex}}$-answer set; this relation also holds with respect to the FLP semantics.

Theorem 3. *For each FLP-answer set* $I \in \mathcal{I}_P$ *of* P, $w\left(\mathcal{A}_P^{\textsc{Hex}}\right) \subseteq_p I$.

Example 5 (Example 4 cont'd). Observe that $w\left(\mathcal{T}_P^{\textsc{Hex}}\right) = (\emptyset, \{a, b\}) \subseteq_p I$, and that $I = \{a, b\}$ is an FLP-answer set of the normal HEX program P of Example 3.

4 Fixpoint Semantics for Disjunctive HEX Programs

In this section, we extend the 2-valued answer-set semantics [12] to the class of disjunctive HEX programs. For such programs, $T_P^{\textsc{Hex}}$ is no longer a lattice operator; thus AFT—which studies fixpoints of lattice operators—is not applicable. However, by combining ideas from disjunctive logic programming and AFT, Pelov and Truszczyński [15] extended parts of the AFT to the case of *non-deterministic* operators.

In the sequel, let P be a disjunctive HEX program. First, we define the non-deterministic immediate consequence operator $N_P^{\textsc{Hex}}$. To this end, we recall the *Smyth ordering* [19]

$\sqsubseteq$ on $\mathfrak{P}(\mathcal{I}_P)$, in which $\mathcal{J} \sqsubseteq \mathcal{K}$ if for every $K \in \mathcal{K}$ some $J \in \mathcal{J}$ exists such that $J \subseteq K$. Note that $\sqsubseteq$ is reflexive and transitive, but not anti-symmetric; thus, $\mathfrak{P}(\mathcal{I}_P)$ endowed with $\sqsubseteq$ is not a poset. However, if we consider only the anti-chains in $\mathfrak{P}(\mathcal{I}_P)$ (i.e., only those $\mathcal{J}$ where each $J \in \mathcal{J}$ is minimal w.r.t. $\subseteq$), denoted $\mathfrak{P}_{min}(\mathcal{I}_P)$, then $(\mathfrak{P}_{min}(\mathcal{I}_P), \sqsubseteq)$ is a poset with least element $\{\emptyset\}$ (cf. [13]). Denote for any set D of disjunctions over At_P (i.e., subset of the disjunctive Herbrand base DHB_P of P [13]) by $MM(D)$ the set of the minimal models of D.

Definition 4. *By the* non-deterministic van Emden-Kowalski operator *of P we refer to the operator $N_P^{\text{HEX}} : \mathcal{I}_P \to \mathfrak{P}_{min}(\mathcal{I}_P)$ where $N_P^{\text{HEX}}(I) = MM\left(I \cup T_P^{\text{HEX}}(I)\right)$.*

Intuitively, $N_P^{\text{HEX}}(I) = \mathcal{J}$ consists of all interpretations $J \in \mathcal{J}$ representing minimal possible outcomes of P after one step of rule applications; moreover, when applying N_P^{HEX} to I we assume each $a \in I$ to be *true*.

We call $I \in \mathcal{I}_P$ a *fixpoint* of N_P^{HEX}, if $I \in N_P^{\text{HEX}}(I)$, and *pre-fixpoint* of N_P^{HEX}, if $N_P^{\text{HEX}}(I) \sqsubseteq \{I\}$. We denote the set of all minimal fixpoints of N_P^{HEX} by $\mathrm{mfp}(N_P^{\text{HEX}})$.

Proposition 3. *For every $I \in \mathcal{I}_P$, I is a minimal model of P iff $I \in \mathrm{mfp}(N_P^{\text{HEX}})$.*

4.1 Non-Deterministic Approximations and Computations

We can consider N_P^{HEX} as an "extension" of T_P^{HEX} to the class of disjunctive HEX programs. However, an important property of T_P^{HEX} which N_P^{HEX} does not enjoy is *iterability*. In this section, we define non-deterministic approximations [15] of N_P^{HEX} and show how to "iterate" them in terms of *computations* [14].

Definition 5. *Define, for each $(I_1, I_2) \in \mathcal{I}_P^c$, (i) the* non-deterministic Fitting approximation *of N_P^{HEX} by $\mathcal{F}_P^{\text{HEX}}(I_1, I_2) = MM\left(I_1 \cup \Phi_P^{\text{HEX}}(I_1, I_2)_1\right)$, and (ii) the* non-deterministic ultimate approximation *by $\mathcal{N}_P^{\text{HEX}}(I_1, I_2) = MM\left(I_1 \cup \mathcal{T}_P^{\text{HEX}}(I_1, I_2)_1\right)$.*

In the sequel, let $\mathcal{A}_P^{\text{HEX}} \in \{\mathcal{F}_P^{\text{HEX}}, \mathcal{N}_P^{\text{HEX}}\}$. The next result shows that $\mathcal{F}_P^{\text{HEX}}$ (resp., $\mathcal{N}_P^{\text{HEX}}$) similarly relates to N_P^{HEX} as Φ_P^{HEX} (resp., $\mathcal{T}_P^{\text{HEX}}$) relates to T_P^{HEX}.

Proposition 4. *Let P be a* HEX *program. Then,*

1. *$\mathcal{A}_P^{\text{HEX}}(I, I) = N_P^{\text{HEX}}(I)$, for every $I \in \mathcal{I}_P$;*
2. *$(J_1, J_2) \subseteq_p (I_1, I_2)$ implies $\mathcal{A}_P^{\text{HEX}}(J_1, J_2) \sqsubseteq \mathcal{A}_P^{\text{HEX}}(I_1, I_2)$, for every pair (J_1, J_2) and (I_1, I_2) from $\mathcal{I}_P^c$.*

Like [15], we use computations [14] to formalize the iterated approximation of N_P^{HEX}.

Definition 6. *Let $I \in \mathcal{I}_P$. An $\mathcal{A}_P^{\text{HEX}}$-$I$-*computation *(in the sense of [14]) is a sequence $J^{I,\uparrow} = (J_i)_{i \geq 0}$, $J_i \in \mathcal{I}_P$, such that $J_0 = \emptyset$ and, for every $n \geq 0$,*
1. $J_n \subseteq J_{n+1} \subseteq I$, and 2. $J_{n+1} \in \mathcal{A}_P^{\text{HEX}}(J_n, I)$.
We call $J^{I,\infty} = \bigcup_{i \geq 0} J_i$ the result *of the computation $J^{I,\uparrow}$. Furthermore, $J \subseteq I$ is* $\mathcal{A}_P^{\text{HEX}}$-$I$-derivable*, if some computation $J^{I,\uparrow}$ with result $J^{I,\infty} = J$ exists.*

Example 6. Let $P = \{a \vee b;\ c \leftarrow a, \sim b\}$. We show that $I = \{a, c\}$ is $\mathcal{F}_P^{\text{HEX}}$-$I$-derivable. Let $J_0 = \emptyset$. We compute $\mathcal{F}_P^{\text{HEX}}(J_0, I) = MM(\{a \vee b\}) = \{\{a\}, \{b\}\}$, and let $J_1 = \{a\}$. For the next iteration, we compute $\mathcal{F}_P^{\text{HEX}}(J_1, I) = MM(\{a, a \vee b, c\}) = \{I\}$, and consider $J_2 = I$. Finally, since we have $\mathcal{F}_P^{\text{HEX}}(J_2, I) = \{I\}$, we set $J_n = I$ for every $n \geq 2$, and obtain a $\mathcal{F}_P^{\text{HEX}}$-$I$-computation $J^{I,\uparrow}$ with result $J^{I,\infty} = I$, which shows that I is $\mathcal{F}_P^{\text{HEX}}$-$I$-derivable.

4.2 Answer-Set Semantics

We now carry the concepts of Section 3.1 over to disjunctive HEX programs and the corresponding non-deterministic operators as follows: (i) instead of considering T_P^{HEX}, we consider the non-deterministic van Emden-Kowalski operator N_P^{HEX} as an appropriate one-step consequence operator; (ii) instead of iterating $\mathcal{A}_P^{\mathrm{HEX}}(\,.\,,I)_1$ to the least fixpoint $\mathcal{A}_P^{\mathrm{HEX},\downarrow}(I)$, we "iterate" $\mathcal{A}_P^{\mathrm{HEX}}(\,.\,,I)$ in terms of $\mathcal{A}_P^{\mathrm{HEX}}$-$I$-computations to the *minimal* fixpoints $\mathcal{A}_P^{\mathrm{HEX},\downarrow}(I)$; (iii) since disjunctive rules are non-deterministic, we consider minimal instead of least models, and minimal instead of least fixpoints. With these intuitions in place, we now define (2-valued) answer-set semantics (cf. [15]).

Definition 7 (Answer-Set Semantics). *We say that $I \in \mathcal{I}_P$ is an* $\mathcal{A}_P^{\mathrm{HEX}}$-answer set*, if $I \in \mathcal{A}_P^{\mathrm{HEX},\downarrow}(I) = \mathrm{mfp}\left(\mathcal{A}_P^{\mathrm{HEX}}(\,.\,,I)\right)$; it is an* ultimate answer set *of P, if I is an $\mathcal{N}_P^{\mathrm{HEX}}$-answer set of P.*

Example 7. Let $P = \{p(a) \vee q(a);\ \leftarrow \sim p \subseteq^{\#} q\}$, and let $I = \{q(a)\}$. First, we compute $\mathcal{F}_P^{\mathrm{HEX}}(\emptyset, I) = \{\{p(a)\}, I\}$ which shows that $\emptyset$ is not a fixpoint of $\mathcal{F}_P^{\mathrm{HEX}}(\,.\,,I)$; second, we compute $\mathcal{F}_P^{\mathrm{HEX}}(I,I) = \{I\}$, that is, I is a minimal fixpoint of $\mathcal{F}_P^{\mathrm{HEX}}(\,.\,,I)$ and thus an $\mathcal{F}_P^{\mathrm{HEX}}$-answer set. Since $I' = \{p(a)\}$ violates the constraint and is thus not an $\mathcal{F}_P^{\mathrm{HEX}}$-answer set, I is the only $\mathcal{F}_P^{\mathrm{HEX}}$-answer set.

The next result summarizes some basic relationships between the non-deterministic ultimate and Fitting approximation, e.g., that, as in the normal case, the ultimate approximation $\mathcal{N}_P^{\mathrm{HEX}}$ is "more precise" than the Fitting approximation $\mathcal{F}_P^{\mathrm{HEX}}$.

Proposition 5. *Let P be a* HEX *program. Then,*

1. *$\mathcal{F}_P^{\mathrm{HEX}}(I_1, I_2) \sqsubseteq \mathcal{N}_P^{\mathrm{HEX}}(I_1, I_2)$, for every $(I_1, I_2) \in \mathcal{I}_P^{\mathrm{c}}$.*
2. *If $I \in \mathcal{I}_P$ is an $\mathcal{F}_P^{\mathrm{HEX}}$-answer set, then I is an ultimate answer set of P.*

A basic requirement for semantics of logic programs is supportedness; in the disjunctive case, we say that an interpretation I is *supported* , if for every atom $a \in I$ there exists some rule $r \in P$ such that $I \models B(r)$, $a \in H(r)$, and $I \not\models H(r) - \{a\}$.

Theorem 4. *Let $I \in \mathcal{I}_P$. If I is an $\mathcal{A}_P^{\mathrm{HEX}}$-answer set, then I is supported.*

Next, we relate our fixpoint-based answer set semantics to the "standard" FLP semantics. Theorem 2 established that all 2-valued Φ_P^{HEX}-answer sets of a normal HEX program are FLP-answer sets. An analogous result holds for disjunctive HEX programs.

Theorem 5. *Let $I \in \mathcal{I}_P$. If I is an $\mathcal{F}_P^{\mathrm{HEX}}$-answer set, then I is an FLP-answer set of P.*

However, the converse of Theorem 5 does not hold in general (cf. Example 3).

Note that $\mathcal{A}_P^{\mathrm{HEX}}$-answer sets as in Definition 7 are non-constructive. However, we can construct every $\mathcal{A}_P^{\mathrm{HEX}}$-answer set bottom-up and identify it with an additional test.

Theorem 6. *Let $I \in \mathcal{I}_P$. Then, I is an $\mathcal{A}_P^{\mathrm{HEX}}$-answer set iff I is $\mathcal{A}_P^{\mathrm{HEX}}$-$I$-derivable and no $J \subset I$ exists such that $J \in \mathcal{A}_P^{\mathrm{HEX}}(J, I)$.*

Example 8. In Example 6, we have seen that the $\mathcal{F}_P^{\mathrm{HEX}}$-answer set $I = \{a, c\}$ of $P = \{a \vee b;\ c \leftarrow a, \sim b\}$ is $\mathcal{F}_P^{\mathrm{HEX}}$-$I$-derivable, and in Example 7 that the $\mathcal{F}_P^{\mathrm{HEX}}$-answer set $I = \{q(a)\}$ of $P = \{p(a) \vee q(a);\ \leftarrow \sim p \subseteq^{\#} q\}$ is $\mathcal{F}_P^{\mathrm{HEX}}$-$I$-derivable. On the other hand, $I = \{p(a), q(a)\}$ is $\mathcal{F}_P^{\mathrm{HEX}}$-$I$-derivable w.r.t. $P = \{p(a) \vee q(a);\ p(a) \leftarrow p \subseteq^{\#} q\}$, while $J = \{p(a)\} \in \mathcal{F}_P^{\mathrm{HEX}}(J, I)$; thus I is not an $\mathcal{F}_P^{\mathrm{HEX}}$-answer set of P.

5 Related Work

Approximation Fixpoint Theory. AFT [5,7] builds on Fitting's seminal work on bilattices and fixpoint semantics of logic programs. In [5], the framework was introduced upon (symmetric) approximations of $\mathcal{A}$ operating on the bilattice L^2; in logic programming, L^2 corresponds to the set $\mathcal{I}_P^2$ of all *4-valued* interpretations (I_1, I_2) of P. However, as pointed out in [7], under the usual interpretations of logic programs only the *consistent* (i.e., *3-valued*) fragment $\mathcal{I}_P^c$ of $\mathcal{I}_P^2$ has an intuitive meaning. Therefore, [7] advanced AFT for consistent approximations, i.e., the 3-valued case also adopted here.

As demonstrated also by our work, a strength of AFT is its flexibility regarding *language extensions*. Recall from Section 3 that to extend the semantics from ordinary normal programs to normal HEX programs, we just had to extend the 3-valued interpretation to the new language construct (i.e., external atoms). A principled way to cope with language extensions under 4-valued interpretations was recently mentioned in [4]; it hinges on 4-valued immediate consequence operators satisfying certain properties ($\leq_p$-monotonicity and symmetry). It is possible to generalize Definition 1 to this setting.

Pelov and Truszczyński's Computations [15]. We used *non-deterministic* operators to define 2-valued (ultimate) answer sets of *disjunctive* HEX programs, motivated by [15]. However, our approach is not entirely identical to [15]; we elaborate here on differences.

As aggregates can be simulated by external atoms (cf. Section 3.1 in [10]), we translate the definitions in [15] to the language of HEX programs and define, for a disjunctive HEX program P, $N_P^{Sel}(I) = Sel\left(T_P^{\text{HEX}}(I)\right)$ where $Sel : \mathfrak{P}(DHB_P) \to \mathfrak{P}(\mathcal{I}_P)$ is a *selection function*. As we only used the selection function MM in this paper, we focus on N_P^{MM} in the sequel.

Pelov and Truszczyński [15] proposed the notion of *computation* [14] as an appropriate formalization of the process of "iterating" N_P^{MM}. In Section 4, we successfully applied it to non-deterministic (ultimate) approximations, and proved in Theorem 6 that (ultimate) answer sets are *derivable*. The following example shows that the definition of N_P^{MM} as such is not compatible with the notion of computation.

Example 9 ([15], Example 3). Let $P = \{a \vee b \vee c; a \leftarrow b; b \leftarrow c; c \leftarrow a\}$. Observe that $I = \{a, b, c\}$ is the only model of P. By applying N_P^{MM} to $J_0 = \emptyset$, we obtain $N_P^{MM}(J_0) = \{\{a\}, \{b\}, \{c\}\}$. However, since $N_P^{MM}(\{a\}) = \{\{c\}\}$, $N_P^{MM}(\{b\}) = \{\{a\}\}$, and $N_P^{MM}(\{c\}) = \{\{b\}\}$, there is no computation $J^\uparrow$ with result $J^\infty = I$. On the other hand, it is easy to see that I is $\mathcal{A}_P^{\text{HEX}}$-$I$-derivable.

Description Logic Programs [9]. Description logic programs[5] (*dl-programs*) [9] are precursors of HEX programs [10] that allow *dl-atoms* (i.e., bi-directional links between a logic program and a description logic ontology) in rule bodies. As shown in [10], we can simulate every dl-program $\mathcal{KB}$ by a normal HEX program $P = P_{\mathcal{KB}}$, which allows us to compare the semantics defined in Section 3 with the strong and weak answer-set semantics and the well-founded semantics defined in [9].

Let $\mathcal{KB}$ be a dl-program and P be the respective normal HEX program; let $At_P^{\#,m}$ be a (fixed) set of all external atoms $a \in At_P^{\#}$ known to be monotone, and let $At_P^{\#,?} = At_P^{\#} - At_P^{\#,m}$. Then the *strong Gelfond-Lifschitz reduct* [9] of P relative to $I \in \mathcal{I}_P$ is

[5] `http://sourceforge.net/projects/dlvhex/files/dlvhex-dlplugin/`

$$sP^I = \Big\{ H(r) \leftarrow B^+(r) - At_P^{\#,?} : r \in P : I \models B^+(r) \cap At_P^{\#,?}, I \models B^{\sim}(r) \Big\}.$$

Note that sP^I is a negation-free monotone normal HEX program, which implies that $T_{sP^I}^{\text{HEX}}$ has a least fixpoint; we call I a *strong answer set* [9] of P, if $I = \text{lfp}(T_{sP^I}^{\text{HEX}})$.

The next example shows that neither the strong nor the weak answer-set semantics coincides with the (ultimate) answer sets of P.

Example 10. Let $P = \{p(a) \leftarrow \sim (not\ p(a))^{\#}\}$ where $I \models (not\ p(a))^{\#}$ if $I \not\models p(a)$.[6] We show that $I = \{p(a)\}$ is a strong answer set of P. As $I \models \sim (not\ p(a))^{\#}$, sP^I consists of the fact $p(a)$. Hence, I is the least fixpoint of $T_{sP^I}^{\text{HEX}}$ and thus a strong answer set of P. On the other hand $\mathcal{T}_P^{\text{HEX},\downarrow}(I) = \emptyset$, which shows that I is not an ultimate answer set of P and hence not an Φ_P^{HEX}-answer set (cf. Proposition 2). As every strong answer set of P is also a weak answer set [9], the same holds for the weak answer set semantics.

However, the next result shows that for *monotone* dl-programs, the semantics in this paper coincide with the semantics given in [9]; note that well-founded semantics was defined in [9] under restriction to monotone dl-programs using unfounded sets.

Theorem 7. *Let $\mathcal{KB}$ be a monotone dl-program and let $P = P_{\mathcal{KB}}$.*

1. *For each $I \in \mathcal{I}_P$, I is a strong answer set of P iff I is a 2-valued Φ_P^{HEX}-answer set.*
2. *For each $(I_1, I_2) \in \mathcal{I}_P^c$, (I_1, I_2) is the well-founded model of P as defined in [9] iff (I_1, I_2) is the Φ_P^{HEX}-well-founded model.*

Shen's Strongly Well-Supported Semantics [18]. Shen [18] defined (weakly and strongly) well-supported semantics for dl-programs. As the latter can be simulated by normal HEX programs, we rephrase Shen's definition in the HEX-setting.

Given a normal HEX program P and $(I_1, I_2) \in \mathcal{I}_P^c$, Shen's notion "$I_1$ up to I_2 satisfies literal ℓ" is equivalent to our 3-valued evaluation function, in symbols $\langle\, \ell \,\rangle_{(I_1,I_2)} = \mathbf{t}$. We thus can characterize Shen's fixpoint operator S_P^{HEX} [18, Definition 5] as follows.

Proposition 6. *For each $(I_1, I_2) \in \mathcal{I}_P^c$, $S_P^{\text{HEX}}(I_1, I_2) = \Phi_P^{\text{HEX}}(I_1, I_2)_1$.*

Finally, call I an *strongly well-supported answer set* [18] of P, if $I = \text{lfp}\left(S_P^{\text{HEX}}(\,.\,, I)\right)$. The following result is an immediate consequence of Proposition 6.

Theorem 8. *Let $I \in \mathcal{I}_P$. Then, I is a strongly well-supported answer set of P iff I is a 2-valued Φ_P^{HEX}-answer set.*

The equivalences above show that Shen's (strongly) well-supported answer-set semantics is naturally captured within the more general framework of AFT. However the use of AFT allowed us to obtain in addition the whole class of 3-valued (ultimate) answer-set semantics (which contain the well-founded semantics), in a more general (and perhaps more elegant) approach than the one in [18].

[6] For readers familiar with dl-programs, note that P amounts to the dl-program $\mathcal{KB} = (\emptyset, \{p(a) \leftarrow \sim DL[S \cap\!\!\!\!\cap\, p; \neg S](a)\})$ where $DL[S \cap\!\!\!\!\cap\, p; \neg S](a)$ is a dl-atom, S is a concept, and $\cap\!\!\!\!\cap$ is the constraint update operator (cf. [9]).

6 Discussion and Conclusion

The goal of this paper was to extend the well-founded-, and the (3-valued) answer-set semantics to the class of HEX programs [10] by applying AFT [5,7], and to compare them with the "standard" FLP semantics. This was in particular relevant, because HEX programs constitute a powerful extension of ordinary disjunctive programs, and are able to represent various other formalisms (e.g., dl-programs; see [10]).

As a result of our investigation, we obtained constructive and uniform semantics for a general class of logic programs with nice properties. More precisely, for normal HEX programs, our 2-valued answer-sets based on AFT turned out to be well-supported, which is regarded as a positive feature. Moreover, Shen's strongly well-supported answer set semantics (formulated for dl-programs) coincides with the 2-valued Φ_P^{HEX}-answer set semantics. Furthermore, to the best of our knowledge, the well-founded semantics for normal HEX programs has not been defined before; it coincides on positive programs representing dl-programs with the well-founded semantics in [9], and thus generalizes it *a fortiori* to arbitrary dl-programs. Finally, our 2-valued (ultimate) answer-set semantics of disjunctive HEX programs turned out to be bottom-up computable.

Regarding complexity, assume that checking $I \models f^{\#}$ is feasible in polynomial time. Then, generalizing ordinary normal logic programs to normal HEX programs does not increase the worst-case complexity of ultimate semantics [7]. Different to the ordinary case, however, computing well-founded and answer set semantics is not easier than ultimate approximation. More specifically, as deciding 3-valued entailment as in Definition 1 is coNP-hard, we obtain that deciding (in the ground case)
– consequence under the (ultimate) well-founded model is Δ_2^P-complete;
– brave consequence is Σ_2^P-complete for 3-valued (ultimate) answer sets; and
– existence of a 2-valued (ultimate) answer-set is Σ_2^P-complete.

Note that for disjunctive HEX programs, despite nondeterministic computations, deciding brave consequence for ultimate answer sets remains Σ_2^P-complete.

Open issues. Some open issues remain. First, in the case of *infinite* HEX programs, the operators defined in this paper all require an *infinite guess*, which makes the notion of *computation* (see Section 4) infeasible. A possible way to tackle this problem consists of three steps: (i) define a HEX program P to be *ω-evaluable*, if there exists an ω-Turing machine [3] M that accepts the answer sets of P; (ii) simulate M by a positive disjunctive HEX program P_M; and (iii) iterate the monotone operator $N_{P_M}^{\text{HEX}}$ in terms of computations as in Definition 6. For example, normal positive HEX programs with finitary external atoms are ω-evaluable, and more generally HEX programs in which atoms depend only on finitely many other atoms [2]; it remains to find further relevant classes of *ω-evaluable infinitary* HEX *programs*.

Second, in the definition of Φ_P^{HEX} (and, consequently, $\mathcal{F}_P^{\text{HEX}}$) the definition of the 3-valued entailment relation plays a crucial role. In a naive realization, evaluating $(I_1, I_2)(f^{\#})$ is exponential, which possibly can be avoided given further knowledge on f (e.g., monotonicity) or relevant interpretations $J \in [I_1, I_2]$ in the context of the program P. Developing respective pruning conditions is interesting and important from a practical perspective. Alternatively, one can imagine to define 3-valued entailment on a substructure of $[I_1, I_2]$, obeying suitable conditions.

Finally, Truszczyński [20] has extended AFT to algebraically capture the notions of strong and uniform equivalence; it is interesting to apply these results to the class of HEX programs by using the results obtained in this paper.

References

1. Antić, C.: Uniform approximation-theoretic semantics for logic programs with external atoms. Master's thesis, TU Vienna, `http://www.ub.tuwien.ac.at/dipl/2012/AC07814506.pdf` (2012)
2. Baselice, S., Bonatti, P. A., Criscuolo, G.: On finitely recursive programs. TPLP 9(2), 213-238 (2009)
3. Cohen, R.S., Gold, A.Y.: ω-computations on Turing machines. TCS 6, 1–23 (1978)
4. Denecker, M., Bruynooghe, M., Vennekens, J.: Approximation fixpoint theory and the semantics of logic and answer set programs. In: Correct Reasoning, *LNCS 7265*, pp. 178–194. Springer (2012)
5. Denecker, M., Marek, V., Truszczyński, M.: Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning. In: Minker, J. (ed.) Logic-Based Artificial Intelligence, pp. 127–144. Kluwer (2000)
6. Denecker, M., Marek, V., Truszczyński, M.: Uniform semantic treatment of default and autoepistemic logics. Artificial Intelligence pp. 79–122 (2003)
7. Denecker, M., Marek, V., Truszczyński, M.: Ultimate approximation and its application in nonmonotonic knowledge representation systems. Inf.Comp. 192(1), 84–121 (2004)
8. Eiter, T., Brewka, G., Dao-Tran, M., Fink, M., Ianni, G., Krennwallner, T.: Combining nonmonotonic knowledge bases with external sources. In: *Proc. FroCos 2009*, pp. 18–42. LNCS 5749, Springer (2009)
9. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R.: Well-founded semantics for description logic programs in the semantic web. ACM TOCL 12(2), 11:1–11:41 (2011)
10. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In: Proc. IJCAI 2005. pp. 90–96 (2005)
11. Faber, W., Leone, N., Pfeifer, G.: Recursive aggregates in disjunctive logic programs: semantics and complexity. In: Proc. JELIA 2004, *LNCS 3229*, pp. 200–212. Springer (2004)
12. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing 9(3-4), 365–385 (1991)
13. Lobo, J., Minker, J., Rajasekar, A.: Foundations of Disjunctive Logic Programming. MIT Press (1992)
14. Marek, V., Niemelä, I., Truszczyński, M.: Logic programs with monotone cardinality atoms. In: Proc. LPNMR 2004, *LNCS 2923*, pp. 154–166. Springer (2004)
15. Pelov, N., Truszczyński, M.: Semantics of disjunctive programs with monotone aggreggates - an operator-based approach. In: Proc. NMR 2004, pp. 327–334 (2004)
16. Pelov, N., Denecker, M., Bruynooghe, M.: Well-founded and stable semantics of logic programs with aggregates. Theory and Practice of Logic Programming 7(3), 301–353 (2007)
17. Przymusinski, T.: Well-founded semantics coincides with the three-valued stable semantics. Fundamenta Informaticae 13(4), 445–463 (1990)
18. Shen, Y.D.: Well-supported semantics for description logic programs. In: Proc. IJCAI 2011, pp. 1081–1086. AAAI Press (2011)
19. Smyth, M.B.: Power domains. Journal of Computer and System Sciences 16, 23–36 (1978)
20. Truszczyński, M.: Strong and uniform equivalence of nonmonotonic theories – an algebraic approach. Annals of Mathematics and Artificial Intelligence 48(3-4), 245–265 (May 2006)
21. Van Gelder, A., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. Journal of the ACM 38(3), 619–649 (1991)