FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Uniform Approximation-Theoretic Semantics for Logic Programs with External Atoms

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Computational Intelligence

eingereicht von

Christian Antić, B.Sc.
Matrikelnummer 0525482

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuer: O.Univ.Prof. Dipl.-Ing. Dr.techn. Thomas Eiter
Mitwirkung: Dipl.-Ing. Dr.techn. Michael Fink

Wien, 17. Oktober 2012

(Unterschrift Verfasser) (Unterschrift Betreuer)

Erklärung zur Verfassung der Arbeit

Christian Antić, B.Sc.
Werndlgasse 8/5
1210 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Wien, 17. Oktober 2012) (Unterschrift Verfasser)

Abstract

HEX programs are disjunctive logic programs under so called *FLP-answer-set semantics* enhanced with higher-order and external atoms. Since external atoms are generic in the sense that they can represent arbitrary computable Boolean functions, many formalisms (e.g., dl-programs, logic programs with aggregates, etc.) can be simulated by HEX programs.

For the class of classical logic programs, major semantics can be characterized in terms of fixpoints of lattice operators defining one step of logical derivation. However, for logic programs with negation, these operators may be nonmonotone, and in this case the fixpoint theory of Tarksi and Knaster is not applicable. Approximation Theory, on the other hand, is an algebraic framework for studying fixpoints of monotone and *nonmonotone* lattice operators, and thus extends the theory of Tarski and Knaster to the more general class of arbitrary lattice operators.

In the first part of the thesis we extend the classical van Emden-Kowalski operator, and the classical Fitting operator to the class of normal (i.e., disjunction-free) HEX programs, and uniformly define (i) (ultimate) Kripke-Kleene-, (ii) (ultimate) well-founded-, and (iii) (3-valued ultimate) answer-set semantics by applying Approximation Theory. As a result, we obtain well-supported 2-valued (ultimate) answer-set semantics, and well-founded semantics compatible with the standard FLP-answer-set semantics. For monotone normal HEX programs, 2-valued answer-set semantics coincide with the standard FLP-answer-set semantics.

In the case of disjunctive HEX programs, Approximation Theory is not directly applicable. However, by combining ideas from Approximation Theory, logic programs with aggregates, and disjunctive logic programming, we define 2-valued (ultimate) answer-set semantics of disjunctive HEX programs based on non-deterministic operators. We show that these semantics satisfy some desirable properties. Particularly, we show that (ultimate) answer sets are minimal, supported, and derivable in terms of bottom-up computations.

As HEX programs are generic formalisms, the obtained semantics can be applied for a range of formalisms.

Kurzfassung

HEX Programme sind klassische disjunktive Logikprogramme, erweitert um externe Atome und Atome höherer Stufe. Externe Atome erlauben es, beliebige Boolesche Funktionen darzustellen und ermöglichen so den Zugang zu externen Ressourcen (zB prozeduralen Programmen und Datenbanken) und die Repräsentierung unterschiedlicher Formalismen (zB DL Programme, Logikprogramme mit Aggregaten, etc.), ohne dabei die deklarative Natur der Semantik einzubüßen. Dies wurde erreicht, indem Gebrauch von der *FLP-Antwortmengensemantik* gemacht wurde.

Die wohl-fundierte- und die Antwortmengensemantik, obwohl, wie später gezeigt wurde, eng zusammenhängend, sind von Grund auf unterschiedlicher Natur. Wird eine neue Klasse von Logikprogrammen definiert, so werden, in der Regel, die entsprechenden Definitionen dieser klassischen Semantiken "zu Fuß" adaptiert. Die *Approximationstheorie* (engl.: *approximation theory*) schlägt stattdessen einen uniformen, algebraischen Weg vor: basierend auf der Tatsache, dass alle klassichen Semantiken (zB auch die Kripke-Kleene- und Supported Semantik) durch Fixpunkte entsprechender Ein-Schritt-Ableitungsoperatoren charakterisiert werden, können, wie in der Theorie gezeigt, wohl-fundierte- und Antwortmengensemantik als Fixpunkte eines einzigen Operators charakterisiert werden. Da sich dieser Operator in der Regel leicht auf neue Programmklassen erweitern lässt, bietet die Approximationstheorie ein praktikables Werkzeug zur Adaptierung klassischer Semantiken.

Im ersten Teil dieser Arbeit erweitern wir den klassischen van Emden-Kowalski Operator und den klassischen Fitting Operator auf die Klasse normaler (d.h. nicht-disjunktiver) HEX Programme, und definieren (i) eine (ultimative) Kripke-Kleene-, (ii) eine (ultimative) wohl-fundierte-, und (iii) eine (ultimate 3-wertige) Antwortmengensemantik mit Hilfe der Approximationstheorie. Dadurch erhalten wir eine (ulitmative) 2-wertige Antwortmengensemantik die keine selbstbegründeten Atome enthält und eine wohl-fundierte Semantik die mit der FLP-Antwortmengensemantik kompatibel sind. Für monotone, normale HEX Programme fällt die 2-wertige Antwortmengensemantik mit der FLP-Antwortmengensemantik zusammen.

Für die Klasse der disjunktiven HEX Programme ist die Approximationstheorie zwar nicht direkt anwendbar, doch durch die Kombination von Ideen aus der Approximationstheorie, der Logikprogrammierung mit Aggregaten, und der disjunktiven Logikprogrammierung, definieren wir eine 2-wertige (ultimative) Antwortmengensemantik für disjunktive HEX Programme basierend auf nicht-deterministischen Operatoren. Weiters zeigen wir, dass diese Semantik folgende wünschneswerte Eigenschaften besitzt: (ultimative) Antwortmengen sind minimal, supported, und ableitbar in Form von kunstruktiven Berechnungen.

Die in dieser Arbeit erhaltenen Semantiken lassen sich, da HEX Programme sehr allgemein sind, auf verschiedenste, durch HEX Programme repräsentierbare, Formalismen anwenden.

Danksagungen

Ich danke Professor Thomas Eiter für den Raum zur Selbstständigkeit, der kompetenten Betreuung und für die interessanten Lehrveranstaltungen, die mich zur Wahl dieses Diplomarbeitsthemas bewogen haben.

Ich danke Michael Fink für die vielen inspirierenden und intensiven Gespräche in angenehmer Atmosphäre.

Ich danke meiner Mutter und meinen Großeltern für Ihre finanzielle Unterstützung, ohne die mir ein Studium nicht möglich gewesen wäre.

Ich danke meiner Freundin Katja Maria Grafl für die vielen ermutigenden Worte, für Ihr unerschöpfliches Verständnis, und für ihre liebevolle Art, mit der sie die alltägliche Arbeit versüßt hat.

Contents

CHAPTER 1

Introduction

This chapter is introductory. First, in Section 1.1 we give a very brief and informal overview of the field of logic programming. Second, in Section 1.2 we motivate our approach of using Approximation Theory for defining semantics of HEX programs. Third, in Section 1.3 we outline the main contributions of this thesis. Finally, in Section 1.4 we give an overview of the structure of the rest of the thesis.

1.1 Background

Classical first-order logic[1] was initially introduced as a mathematical tool for precisely formalizing and analyzing mathematical theories. These theories are *monotone* in their nature. That is, given a set of assumptions (i.e., axioms), the set of derivable conclusions monotonically increases when adding new assumptions.

In contrast, commonsense reasoning (McCarthy, 1959) is an inherently *nonmonotone* form of (logical) reasoning for which classical first-order logic, in its original form, is not an adequate formalism. In the last three decades much effort has been investigated to capture essential parts of commonsense reasoning, with *default logic* (Reiter, 1980), *autoepistemic logic* (Moore, 1985), and *logic programs with negation* (Clark, 1978; Lloyd, 1987) being the most prominent formalisms today, the latter arguably being the most successful nonmonotonic reasoning and knowledge representation formalism with many practical applications.

Logic programs are rule-based systems with the rules and facts being written in a sublanguage of the language of classical first-order logic extended by a unary operator "$\sim$" (or "not") denoting *negation-as-failure* (or *default negation*) (Clark, 1978). While each monotone (i.e., negation-free) logic program has a unique least Herbrand model (with the least model semantics (Van Emden and Kowalski, 1976) being the accepted semantics for this class of programs), for general logic programs a large number of different purely declarative semantics exist. Many of it have been introduced some 20 years ago, among them the *Kripke-Kleene semantics* (Fitting,

[1] We assume the reader to be familiar with the basics of mathematical logic (see, e.g., Hinman (2005)).

1985), the *answer-set semantics* (Gelfond and Lifschitz, 1991), and the *well-founded semantics* (Van Gelder et al., 1991) (for a survey on negation in logic programming see, e.g., **?**).

The well-founded semantics, because of its nice computational properties,[2] plays an important role in Database Theory. However, with the emergence of efficient solvers such as, e.g., DLV (Leone et al., 2006), Smodels (Simons et al., 2002), Cmodels (Giunchiglia et al., 2006), and Clasp (Gebser et al., 2012), programming under answer-set semantics led to a predominant declarative problem solving paradigm, called *answer-set programming* (or *ASP*) (Marek and Truszczyński, 1999; Lifschitz, 2002). Answer-set programming has a wide range of applications (Brewka et al., 2011), and has been successfully applied to various AI-related subfields such as, e.g., planning, and diagnosis (for a survey see, e.g., (Eiter et al., 2009b)). Driven by this practical needs, a large number of extensions of classical answer-set programs have been proposed, e.g., *aggregates* (see, e.g., (Faber et al., 2004, 2011; Pelov and Truszczyński, 2004; Pelov, 2004)), *choice rules* (Niemelä et al., 1999), *dl-atoms* (Eiter et al., 2004a, 2008), and general *external atoms* (Eiter et al., 2005).

Description logic programs (Eiter et al., 2004a, 2008), designed as a tool for reasoning in the Semantic Web, extend classical normal programs by so called *dl-atoms*, i.e., atoms which build a bidirectional link between the program and a description logic knowledge base.

By allowing arbitrary external sources, Eiter et al. (2005) define so called *HEX programs*, extending classical normal programs under answer-set semantics (Gelfond and Lifschitz, 1991) by (i) disjunctive rules (Gelfond and Lifschitz, 1991; Lobo et al., 1992; Eiter et al., 1997), (ii) higher-order atoms, and (iii) external atoms for software interoperability (Eiter et al., 2009a). Disjunctions in the head of rules introduce non-determinism into the semantics of logic programs, and lead to the "guess & check" programming paradigm (Eiter et al., 2000; Lifschitz, 2002). Since external atoms can represent arbitrary computable Boolean functions, HEX programs constitute a powerful extension of classical disjunctive programs with many applications[3] (see, e.g., Hoehndorf et al. (2007), Heymans and Toma (2008), and Basol et al. (2010)). However, since external atoms may be nonmonotone, Eiter et al. (2005) define answer-set semantics of HEX programs in terms of the FLP-reduct (Faber et al., 2004, 2011) (called *FLP-answer-set semantics* henceforth) instead of the traditional Gelfond-Lifschitz reduct (Gelfond and Lifschitz, 1988) used for classical disjunctive programs (Gelfond and Lifschitz, 1991).

For excellent introductions to the field of answer-set programming we refer the reader to (Eiter et al., 2009b; Brewka et al., 2011; Baral, 2003).

1.2 Problem Statement and Method

Lifting the semantics of classical normal or disjunctive programs to an extended class of programs (e.g., HEX programs) means in general: decide which semantics are relevant, and then try to reasonably adapt the original definitions to the new setting. Beside the fact that adapting each semantics separately can be cumbersome, this strategy may lead, for each intended seman-

[2] Computing the unique 3-valued well-founded model is tractable, i.e., polynomial.

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

tics, to many possible extensions, and in general it is not clear which extension to accept as the "canonical" one.

Approximation Theory is an abstract algebraic framework for studying fixpoints of (monotone or nonmonotone) lattice operators in terms of *(monotone) approximations*. In this sense, Approximation Theory extends the well-known fixpoint theory of Tarski and Knaster (Tarski, 1955) to the more general class of arbitrary lattice operators. Based on results in (Fitting, 1985, 1991, 2002), Denecker et al. (2000a) show that, given a classical normal (i.e., disjunction-free) program Π, the following semantics can be characterized in terms of the monotone Fitting approximation Φ_Π (Fitting, 1985) of the van Emden-Kowalski lattice operator T_Π: (i) *Kripke-Kleene semanitcs* (Fitting, 1985), (ii) *well-founded semantics* (Van Gelder et al., 1991), and (iii) *(3-valued) answer-set semantics* (Gelfond and Lifschitz, 1991; Przymusinski, 1990). However, Fitting's operator Φ_Π is not the most precise[4] approximation of T_Π. Denecker et al. (2004) show that T_Π has a most precise approximation $\mathcal{T}_\Pi$, called the *ultimate* approximation of T_Π. This ultimate approximation yields the most precise Kripke-Kleene and well-founded semantics, and the largest number of answer sets. Generally, for each lattice operator O defined on a complete lattice L, there exists an ultimate (i.e., most precise) approximation $\mathcal{O}$ of O. Denecker et al. (2004) show that $\mathcal{O}$ can be *algebraically* characterized in terms of O. However, Denecker et al. (2004) also show that computing ultimate semantics is more complex from a computational point of view.

Motivated by this elegant algebraic characterization of the major semantics of classical normal programs (Denecker et al., 2000a) and of default and autoepistemic logic (Denecker et al., 2000b, 2003) within the framework of Approximation Theory, in the first part of this thesis we uniformly define (ultimate) Kripke-Kleene, 2-, and 3-valued (ultimate) answer-set semantics, and (ultimate) well-founded semantics of *normal* (i.e., *disjunction-free*) HEX programs by exploiting the machinery of Approximation Theory. More precisely, given a normal HEX program Π, we define the extended van Emden-Kowalski operator T_Π^{hex}, and the extended Fitting operator Φ_Π^{hex}, and show that Φ_Π^{hex} is an approximation of T_Π^{hex}. Thus, by applying Approximation Theory, we obtain Φ_Π^{hex}-Kripke-Kleene semantics, 2-, and 3-valued Φ_Π^{hex}-answer-set semantics, and Φ_Π^{hex}-well-founded semantics. Moreover, by substituting the ultimate approximation $\mathcal{T}_\Pi^{hex}$ for Φ_Π^{hex}, we obtain ultimate (i.e., more precise) versions of the listed semantics. Furthermore, we compare our fixpoint semantics with the "standard" FLP-answer-set semantics; particularly, we show that every 2-valued Φ_Π^{hex}-answer set is also an FLP-answer set, whereas the contrary does, in general, not hold. We argue that this divergence is due to the well-supportedness (Shen, 2011) (i.e., constructiveness) of our answer-set semantics.

Unfortunately, for *disjunctive* HEX programs, the extended van Emden-Kowalski operator is not a lattice operator and, hence, Approximation Theory is not directly applicable. However, by combining ideas from Approximation Theory, logic programs with aggregates (Pelov and Truszczyński, 2004; Pelov, 2004), and disjunctive logic programming (Minker and Rajasekar, 1990; Lobo et al., 1992; Fernández and Minker, 1995; Seipel et al., 1997), in the second part of this thesis we define 2-valued (ultimate) answer-set semantics of disjunctive HEX programs. In detail, given a disjunctive HEX program Π, we define (i) the non-deterministic van Emden-Kowalski operator N_Π^{hex}, (ii) the non-deterministic Fitting operator $\mathcal{F}_\Pi^{hex}$, and the more precise

[4] With respect to Approximation Theory

(iii) non-deterministic ultimate operator $\mathcal{N}_{\Pi}^{hex}$, and show that $\mathcal{F}_{\Pi}^{hex}$ and $\mathcal{N}_{\Pi}^{hex}$ are approximations of N_{Π}^{hex}. Then, specific minimal fixpoints of $\mathcal{F}_{\Pi}^{hex}$ and $\mathcal{N}_{\Pi}^{hex}$ define answer sets and ultimate answer sets of Π, respectively. However, this definition is non-constructive. Therefore, as proposed by (Pelov and Truszczyński, 2004), we define the notion of *computation* (Marek et al., 2004) as an adequate notion of iterating non-deterministic operators in a bottom-up manner; moreover, we show that every (ultimate) answer set is derivable by some computation. Furthermore, we show that (ultimate) answer sets are supported (Brass and Dix, 1997) and, as in the case of normal HEX programs, every 2-valued answer set is also an FLP-answer set, whereas the converse does, in general, not hold.

1.3 Summary of Contributions

The main *contributions* of this thesis can be summarized as follows:

1. We define the full class of 3-valued semantics (Gelfond and Lifschitz, 1991; Przymusinski, 1990; Van Gelder et al., 1991; Fitting, 1985) of *normal* (i.e., *disjunction-free*) HEX programs (Eiter et al., 2005) in a uniform and principled way by applying the algebraic framework of Approximation Theory (Denecker et al., 2000a, 2002, 2004). In particular, this class contains Kripke-Kleene semantics (Fitting, 1985), 2-, and 3-valued answer-set semantics (Gelfond and Lifschitz, 1991; Przymusinski, 1990), and well-founded semantics (Van Gelder et al., 1991) which all play an important role in classical logic programming. Moreover, we define *ultimate* versions of this semantics which are the most precise semantics with respect to Approximation Theory (Denecker et al., 2004).
2. We show that 2-valued answer sets of normal HEX programs are free of circular justifications. That is, we show that every 2-valued answer set is *well-supported* (Shen, 2011; Fages, 1994) (i.e., *constructive*), which is a key requirement for characterizing relevant models. Moreover we exhaustively compare our semantics with the standard one as defined in (Eiter et al., 2005), and conclude that although the semantics are equivalent for *monotone* normal HEX programs, they diverge for the class of general normal HEX programs. More precisely, we show that each 2-valued Φ_{Π}^{hex}-answer set is an FLP-answer set, whereas the converse does, in general, not hold. Moreover, we show that our 2-valued Φ_{Π}^{hex}-answer-set semantics coincides with Shen's strongly well-supported answer-set semantics (Shen, 2011). However, since we use Approximation Theory, our approach is more general and more elegant from a mathematical point of view.
3. We define 2-valued (ultimate) answer-set semantics (Gelfond and Lifschitz, 1991) of *disjunctive* HEX programs (Eiter et al., 2005) by translating some of the concepts of Approximation Theory to the class of *non-deterministic* operators (Pelov, 2004; Pelov and Truszczyński, 2004), and by combining them with ideas from classical disjunctive logic programming (Minker and Rajasekar, 1990; Lobo et al., 1992; Fernández and Minker, 1995; Seipel et al., 1997). Crucial is here the definition of the non-deterministic van Emden-Kowalski operator N_{Π}^{hex}. We define N_{Π}^{hex} in such a way that N_{Π}^{hex} and its approximations are "iterable" in terms of *computations* (Marek et al., 2004). This yields a *bottom-up* approach for *constructively* deriving (ultimate) answer sets.

For HEX programs, to the best of our knowledge, there exists no well-founded semantics up so far. Hence, at least for disjunction-free programs, we lift one of the key classical logic programming semantics to a very general class of programs with a large number of applications.

Finally, in contrast to (Pelov and Truszczyński, 2004), our approach is constructive, i.e., the defined non-deterministic operators can be incrementally applied in a bottom-up manner, thus generalizing the fixpoint characterizations of (ultimate) 2-valued answer-set semantics of normal HEX programs to the class of disjunctive HEX programs.

1.4 Structure of the Thesis

The rest of the thesis essentially consists of the following three main parts: (i) The next two chapters are introductory. In Chapter 2 we define syntax (Section 2.1) and FLP-answer-set semantics (Section 2.2) of HEX programs (Eiter et al., 2005). In Chapter 3 we present essential parts of Approximation Theory (Denecker et al., 2000a, 2002, 2004) by mainly following the lines of (Denecker et al., 2004). In particular, in Section 3.1 we introduce basic notions from Lattice Theory and define the notion of a (product) bilattice (Ginsberg, 1988). In Section 3.2 we define approximations of lattice operators defined on a complete lattice. Every approximation has a least fixpoint, called the *Kripke-Kleene fixpoint*. In Section 3.3 we define stable-revision operators and call their fixpoints *stable fixpoints*. The set of stable fixpoints has a least fixpoint, called the *well-founded fixpoint* which can be computed by transfinite induction. Finally, in Section 3.4 we define *ultimate* approximations which have the most precise Kripke-Kleene-, and well-founded fixpoints, and the most stable fixpoints. The combination of (classical) logic programming and Approximation Theory is illustrated throughout the whole chapter with examples.

(ii) In Chapter 4 we define fixpoint semantics of *normal* (i.e., *disjunction-free*) HEX programs by exploiting the machinery of Approximation Theory presented in Chapter 3. In particular, we define Kripke-Kleene semantics (Section 4.1), 3-valued answer-set semantics (Section 4.2), well-founded semantics (Section 4.3), and ultimate semantics (Section 4.4).

In Chapter 5 we show the following semantic properties: 2-valued answer-set semantics and FLP-answer-set semantics coincide in the case of *monotone* normal HEX programs (Section 5.1); every 2-valued answer set is an FLP-answer set, whereas the converse does, in general, not hold (Section 5.2); (ultimate) well-founded semantics approximate FLP-answer set semantics (Section 5.3); and 2-valued (ultimate) answer sets are well-supported (Shen, 2011; Fages, 1994) (Section 5.4).

(iii) In Chapter 6 we define 2-valued answer-set semantics of *disjunctive* HEX programs (Eiter et al., 2005). Particularly, in Section 6.1 we define the *non-deterministic* van Emden-Kowalski operator. In Section 6.2 we define bottom-up computations (Marek et al., 2004) of the non-deterministic van Emden-Kowalski operator. In Section 6.3 we translate the concept of an approximation (see Section 3.2) to the class of *non-deterministic* van Emden-Kowalski operators (Pelov, 2004; Pelov and Truszczyński, 2004), and define non-deterministic Fitting and ultimate approximations. In Section 6.4 we define 2-valued answer-set semantics, and in Section 6.5 we define ultimate answer-set semantics.

In Chapter 7 we show the following semantic properties: (ultimate) answer-set semantics of disjunctive HEX programs extend 2-valued (ultimate) answer-set semantics of normal HEX programs Section 7.1; (ultimate) answer sets are (algorithmically) derivable by computations (Marek et al., 2004) (Section 7.2); (ultimate) answer sets are supported (Brass and Dix, 1997) (Section 7.3); and every answer set is an FLP-answer set, whereas the converse does, in general, not hold (Section 7.4).

Finally, in Chapter 8 we list and discuss related work, and we finish with conclusions and a general discussion given in Chapter 9.

CHAPTER 2

HEX programs

HEX programs[1] (Eiter et al., 2005) extend classical disjunctive programs (Lobo et al., 1992; Minker and Rajasekar, 1990; Eiter et al., 1997) under answer-set semantics (Gelfond and Lifschitz, 1991) by (i) *higher-order atoms* for higher-order reasoning, and (ii) *external atoms* for software interoperability (Eiter et al., 2009a). Higher-order reasoning is an important requirement, e.g., for programming languages in the Semantic Web. Roughly, a higher-order atom can be an expression of form $p(q)$, where p and q are both predicates, i.e., Π states a second-order property of q. For instance, the rule[2]

$$q(a) \leftarrow p \subseteq q, p(a)$$

intuitively states that whenever a belongs to p and p is a subset of q, then a belongs to q. Since p and q are "properties", $p \subseteq q$ is a second-order relation of p and q.

External atoms permit the use of external sources (e.g., databases, procedural or object-oriented software programs, etc.) while preserving the declarative nature of logic programs. Eiter et al. (2005) achieves this loose coupling by defining answer-set semantics based on the FLP-reduct (Faber et al., 2004, 2011) instead of the traditional Gelfond-Lifschitz reduct (Gelfond and Lifschitz, 1988) which is extensively used in classical logic programming. Since external atoms can represent arbitrary computable Boolean functions, they constitute a powerful extension of classical disjunctive programs.

In this chapter we briefly summarize the notions and results given in (Eiter et al., 2005). In particular, in Section 2.1 we define the syntax of HEX programs. In Section 2.2 we define answer-set semantics (Gelfond and Lifschitz, 1991) based on the FLP-reduct (Faber et al., 2004, 2011). As for classical disjunctive programs (Gelfond and Lifschitz, 1991), every FLP-answer set is minimal and 2-valued, that is, every HEX-atom (i.e., atom or external atom) is either *true* or *false* with respect to some interpretation.

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

[2] Notice that unary predicates can be interpreted as sets.

2.1 Syntax

A *language* $\mathcal{L}^{hex} = (\Sigma, \Sigma^{\#})$ *of (ground*[3] *function-free) HEX programs* (Eiter et al., 2005) is specified by denumerable sets

$$\Sigma = \bigcup_{n \geq 0} \Sigma^{(n)} \quad \text{and} \quad \Sigma^{\#} = \bigcup_{m,k \geq 0} \Sigma^{\#,(m,k)}$$

of *symbols* and *external (predicate) symbols*, respectively. Every symbol $a \in \Sigma^{(n)}$ (resp., external symbol $f^{\#} \in \Sigma^{\#,(m,n)}$) has *arity* n (resp., (m,n)) and the sets Σ and $\Sigma^{\#}$ are disjoint, that is, $\Sigma \cap \Sigma^{\#} = \emptyset$. Elements from $\Sigma^{\#}$ are superscripted with $\#$ (e.g., $f^{\#}, g^{\#}, h^{\#}$ etc.).

In contrast to classical logic programs (Lloyd, 1987), HEX programs may contain so called *higher-order* and *external atoms*. Formally, a *(n-ary) higher-order atom* (or *atom*) is a word $a_0 \ldots a_n \in \Sigma^{n+1}$, $n \geq 0$, where $a_0 \in \Sigma^{(n)}$ and $a_1, \ldots, a_n \in \Sigma$ are symbols. We often write atoms in the more familiar form

$$a_0(a_1, \ldots, a_n).$$

By a *constant* we mean a 0-ary symbol $a_0 \in \Sigma^{(0)}$.

An *((m,n)-ary) external atom* has the form[4]

$$f^{\#}[\mathbf{i}](\mathbf{o}), \tag{2.1}$$

where $f^{\#} \in \Sigma^{\#,(m,n)}$, $\mathbf{i} = i_1 \ldots i_m \in \Sigma^m$ *(=input)*, $m \geq 0$, and $\mathbf{o} = o_1 \ldots o_n \in \Sigma^n$ *(=output)*, $n \geq 0$. For convenience, we assume that for every constant $a \in \Sigma^{(0)}$ there exists an $(1,0)$-ary external symbol $a^{\#} \in \Sigma^{\#,(1,0)}$ representing a (see Example 2.2.1).

A *HEX-atom* is an atom or an external atom, and a *HEX-literal* is a HEX-atom $\ell = a$ or negated HEX-atom $\ell = {\sim} a$, where $\sim$ denotes *negation as failure* (or *default negation*) (Clark, 1978).

A *rule* has the form

$$a_1 \vee \ldots \vee a_k \leftarrow b_1, \ldots, b_\ell, {\sim} b_{\ell+1}, \ldots, {\sim} b_m, \quad k \geq 1, m \geq \ell \geq 0, \tag{2.2}$$

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)*, $B^+(r) = \{b_1, \ldots, b_\ell\}$ *(positive body)*, $B^-(r) = \{b_{\ell+1}, \ldots, b_m\}$ *(negative body)*, $B^{\sim}(r) = \{{\sim} b_{\ell+1}, \ldots, {\sim} b_m\}$ *(NAF-body)*, and $B(r) = B^+(r) \cup B^{\sim}(r)$ *(body)*. To ease notations, we often treat $H(r)$ as a set and write, for a rule r of form (2.2), e.g., $|H(r)| = k$, $a \in H(r)$ if a occurs in $H(r)$, $H(r) - \{a_k\}$ for $a_1 \vee \ldots \vee a_{k-1}$ and so on. Moreover, we say that r is (i) *positive* if $B^{\sim}(r) = \emptyset$, (ii) *normal* (or *disjunction-free*) if $|H(r)| = 1$, (iii) *disjunctive* if $|H(r)| \geq 1$, (iv) *propositional* if every atom a in r has arity 0, and (v) a *fact* if $B(r) = \emptyset$.

Example 2.1.1. Consider, for $p, q \in \Sigma^{(n)}$, $n \geq 1$, the external atom $\subseteq^{\#}[\,](p,q) \in \Sigma^{\#,(0,2)}$, intuitively stating that the extension of p is a subset of the extension of q (see Example 2.2.2). We will write atoms of this form in the more familiar infix notation as $p \subseteq^{\#} q$, and will generally omit parentheses in such cases where no parameters are given.

[3] In the literature, logic programs normally may contain variables; however, since HEX programs do not contain function symbols, each HEX program can be represented by its finite (propositional) grounded version.

[4] Notice that we use here a slightly different syntax than originally used in (Eiter et al., 2005).

A *HEX program* Π is a *finite* set of rules of form (2.2), and we say that Π is (i) *positive* if every rule is positive, (ii) *normal* if every rule is normal, (iii) *disjunctive* if there exists a disjunctive rule in Π, (iv) *propositional* if every rule is propositional, and (v) *classical* if Π contains neither external atoms nor atoms of order ≥ 1, that is, every symbol $p \in \Sigma$ is either a predicate- or constant symbol in the usual sense.

In the rest of the thesis we assume that each HEX program Π implicitly determines its language $\mathcal{L}_\Pi = (\Sigma_\Pi, {\Sigma_\Pi}^\#)$ given by the (finitely many) symbols and external predicate symbols occurring in Π, respectively. Furthermore, we denote the set of all atoms (resp., external atoms) occurring in Π by Λ_Π (resp., $\Lambda_\Pi^\#$).

Notice that Eiter et al. (2005) additionally permit constraints (i.e., rules with empty head) to occur in Π. However, it is well-known that constraints can be represented by normal rules (Gelfond and Lifschitz, 1991), as the next example shows.

Example 2.1.2 (Constraint)**.** A *constraint* is a normal rule r of form

$$\bot \leftarrow b_1, \ldots, b_\ell, \sim b_{\ell+1}, \ldots, \sim b_m, \sim \bot, \quad m \geq 1, \tag{2.3}$$

where $\bot \in \Sigma$ is a special symbol interpreted as "contradiction". Intuitively, the rule states that $B(r)$ is contradictory, that is, whenever $b_1, \ldots, b_\ell, \sim b_{\ell+1}, \ldots, \sim b_m$ all hold in some interpretation I, then a contradiction $\bot$ is derivable. Under FLP-answer-set semantics (see Section 2.2), a constraint of form (2.3) eliminates each model I with $I \models B(r)$ from the answer sets.

Moreover, Gelfond and Lifschitz (1991) introduced extended logic programs by syntactically allowing the occurrence of *strong negation* (or *classical negation*). However, strong negation does not add an additional expressive power, i.e., logic programs containing strong negation can always be translated into equivalent programs without strong negation (Gelfond and Lifschitz, 1991), as the following example demonstrates.[5]

Example 2.1.3 (Strong negation)**.** Let Π be a HEX program and a be an atom occurring in Π. Adding a fresh symbol a' to Σ_Π and a constraint of form

$$\bot \leftarrow a, a', \sim \bot$$

yields a HEX program Π' in which, intuitively, a' behaves like $\neg a$ (the *strong negation* of a). Consequently, given a HEX program Π over $\mathcal{L}_\Pi^{hex} = (\Sigma_\Pi, {\Sigma_\Pi}^\#)$, the program

$$\Pi' = \Pi \cup \left\{ \bot \leftarrow a, a', \sim \bot : a \in \Lambda_\Pi, a' \in \Sigma' \right\}$$

with language $\mathcal{L}_{\Pi'}^{hex} = (\Sigma \cup \Sigma', \Sigma^\#)$, $\Sigma' = \{a' : a \in \Lambda_\Pi\}$, essentially contains two forms of negation, namely: (i) negation as failure "$\sim$" (Clark, 1978), and (ii) *strong* (or *classical*) *negation* "$\neg$" (Gelfond and Lifschitz, 1991).

[5] Readers not familiar with answer-set semantics should consult this example after reading the next section on FLP-answer-set semantics.

2.2 FLP-Answer-Set Semantics

In this section we define answer-set semantics (Gelfond and Lifschitz, 1991) of HEX programs (Eiter et al., 2005) based on the FLP-reduct (Faber et al., 2004, 2011), called *FLP-answer-set semantics* henceforth.

Let Π be a HEX program and $\mathcal{L}_P^{hex} = (\Sigma_\Pi, {\Sigma_\Pi}^\#)$ be the language of Π. The *Herbrand base* HB_Π of Π is the set of all HEX-atoms occurring in Π, that is, $HB_\Pi = \Lambda_\Pi \cup \Lambda_\Pi^\#$. A *(2-valued) interpretation* I of Π is any subset of Λ_Π. Notice that we do not allow external atoms to occur in I. Define the set $\mathcal{I}_\Pi$ of interpretations of Π by $\mathcal{I}_\Pi = \mathfrak{P}(\Lambda_\Pi)$. The greatest element in $\mathcal{I}_\Pi$ with respect to $\subseteq$ is Λ_Π.

Intuitively, every $I \in \mathcal{I}_\Pi$ corresponds to a 2-valued evaluation function $\langle\ .\ \rangle_I$ which assigns to every $a \in I$ (resp., $a \notin I$) the truth-value *true* (resp., *false*). More precisely, given an interpretation $I \in \mathcal{I}_\Pi$, define, for every atom $a \in \Lambda_\Pi$, the *(2-valued) evaluation function (with respect to I)* $\langle\ .\ \rangle_I$ by

$$\langle\, a \,\rangle_I = \begin{cases} \mathbf{t} & \text{if } a \in I, \\ \mathbf{f} & \text{if } a \notin I, \end{cases}$$

where $\mathbf{t}$ (resp., $\mathbf{f}$) denotes the truth-value *true* (resp., *false*). Conversely, given a 2-valued evaluation function $\langle\ .\ \rangle$, define

$$I = \{a \in \Lambda_\Pi : \langle\, a \,\rangle = \mathbf{t}\}.$$

Clearly, this correspondence is one-one; since $\mathcal{I}_\Pi$ is partially ordered by $\subseteq$, we define the ordering $\leq$ on $2 = \{\mathbf{f}, \mathbf{t}\}$ by $\mathbf{f} \leq \mathbf{t}$, and extend it to evaluation functions pointwise, that is,

$$\langle\ .\ \rangle_1 \leq \langle\ .\ \rangle_2 \Leftrightarrow \langle\, a \,\rangle_1 \leq \langle\, a \,\rangle_2 \quad \text{for each atom } a.$$

Moreover, we denote the set of all evaluation functions of a given HEX program Π by 2^{Λ_Π}. Then, $(2^{\Lambda_\Pi}, \leq)$ is isomorphic to $(\mathcal{I}_\Pi, \subseteq)$ and they are both complete lattices.

Notice that we have not assigned a truth-value to external atoms so far. Therefore, we now extend $\langle\ .\ \rangle_I$ to external atoms as follows. We associate with every $m+n$-ary external symbol $f^\# \in {\Sigma_\Pi}^\#$ a computable *(2-valued) interpretation function* $f : \mathcal{I}_\Pi \times \Sigma_\Pi^{m+n} \longrightarrow 2$, and define, for every $\mathbf{i} \in \Sigma_\Pi^m$, $\mathbf{o} \in \Sigma_\Pi^n$,

$$\left\langle\, f^\#\,[\,\mathbf{i}\,]\,(\mathbf{o}) \,\right\rangle_I = f\,(I, \mathbf{i}, \mathbf{o})\,.$$

Moreover, define, for every symbol $a_0 \in \Sigma_\Pi^{(n)}$ of arity n, the *extension* of a_0 with respect to I by

$$a_0^I = \{(a_1, \ldots, a_n) \in \Sigma_\Pi^n : a_0(a_1, \ldots, a_n) \in I\}\,.$$

Example 2.2.1. Let Π be a HEX program. In Section 2.1 we mentioned that for each constant $a \in \Sigma_\Pi^{(0)}$ there exists an external atom $a^\# \in \Sigma_\Pi^{\#,(1,0)}$ representing a. We make this representation more precise: we define the external atom $a^\#$ by $a : \mathcal{I}_\Pi \times \Sigma_\Pi \longrightarrow 2$, [6]

$$a(I, i) = \begin{cases} \mathbf{t} & \text{if } i = a \text{ and } a \in I, \\ \mathbf{f} & \text{otherwise,} \end{cases}$$

[6] Notice that the constant $a \in \Sigma_\Pi^{(0)}$ is different from the interpretation function a of $a^\#$.

and write $a^{\#}$ instead of $a^{\#}[a]$. For instance, the rule $b \leftarrow a$ can be translated into $b \leftarrow a^{\#}$ while preserving the original meaning. However, since external atoms cannot occur in the head of rules, the correspondence is not one-one.

Example 2.2.2. Let Π be a HEX program and consider the external atom $p \subseteq^{\#} q$ of Example 2.1.1. We interpret $\subseteq^{\#}$ as set inclusion and define

$$\subseteq(I, p, q) = \left\langle\, p \subseteq^{\#} q \,\right\rangle_I = \begin{cases} \mathbf{t} & \text{if } p^I \subseteq q^I, \\ \mathbf{f} & \text{otherwise.} \end{cases}$$

Notice that the extension of $\langle\ .\ \rangle_I$ to external atoms given above is a *total* function with domain HB_Π. On the other hand, every interpretation $I \in \mathcal{I}_\Pi$ only determines the truth-value of atoms in Λ_Π. Therefore, the correspondence between 2^{HB_Π} (i.e., the set of all *total* evaluation functions) and $\mathcal{I}_\Pi$ is no longer one-one.

We now define models of Π, that is, interpretations which are compatible with Π. To this end, we define the *logical entailment relation*, for each HEX-atom $a \in HB_\Pi$ and each rule $r \in \Pi$, by

1. $I \models a \Leftrightarrow \langle\, a \,\rangle_I = \mathbf{t}$,
2. $I \models B(r) \Leftrightarrow I \models b$ for every $b \in B^+(r)$ and $I \not\models b'$ for every $b' \in B^-(r)$,
3. $I \models r \Leftrightarrow$ whenever $I \models B(r)$ then $I \models a$ for some $a \in H(r)$,
4. $I \models \Pi \Leftrightarrow I \models r$ for each $r \in \Pi$.

If $I \models \Pi$ then we say that I is a *model* of Π, and we say that Π is *satisfiable* (resp., *unsatisfiable*) if Π has some (resp., no) model.

We now define FLP-answer-set semantics (Eiter et al., 2005) as follows. Let $I \in \mathcal{I}_\Pi$. Define the *FLP-reduct of* Π *relative to* I (Faber et al., 2004, 2011) by

$$f\Pi^I = \{r \in \Pi : I \models B(r)\}\,.$$

We say that I is a *FLP-answer set* of Π if I is a minimal model of $f\Pi^I$ (with respect to set inclusion).

Example 2.2.3. Consider the disjunctive HEX program Π consisting of the following rules:

$$p(a) \vee q(a) \leftarrow$$
$$\bot \leftarrow {\sim} p \subseteq^{\#} q, {\sim} \bot$$

where $I \models p \subseteq^{\#} q$ iff $p^I \subseteq q^I$ (see Example 2.2.2). Intuitively, the first rule states that $a \in p^I$ or $a \in q^I$, whereas the second rule states that p^I necessarily is a subset of q^I (see Example 2.1.2). We show that $I = \{q(a)\}$ is the only FLP-answer set of Π. Let r_1 and r_2 denote the first and second rule, respectively. Since $I \models p \subseteq^{\#} q$, we have $I \not\models B(r_2)$ which implies $f\Pi^I = \{r_1\}$; moreover, since $I \models q(a)$, I is a model of $f\Pi^I$. Hence, since $\emptyset$ is not a model of $f\Pi^I$, and $\emptyset$ is the only proper subset of I, I is a minimal model of $f\Pi^I$ and, thus, an FLP-answer set of Π.

In contrast, if $I = \{p(a)\}$, then $I \models B(r_2)$ but $I \not\models \bot$ which shows that I is not a model of Π. Finally, for $I' = \{p(a), \bot\}$ it is easy to see that I' is a model but not a minimal model of $f\Pi^{I'} = \Pi$, since $I = \{p(a)\}$ is a smaller one. That is, I' is also not an FLP-answer set.

The next proposition states an elementary property of FLP-answer sets, which we will extensively use in the rest of the thesis.

Proposition 2.2.4 (Eiter et al. (2005), Theorem 2)**.** *Let* Π *be a HEX program, and let* $I \in \mathcal{I}_\Pi$*. If* I *is an FLP-answer set of* Π*, then* I *is a minimal model of* Π*.*

CHAPTER 3

Approximation Theory and Classical Logic Programming

The well-known fixpoint theory (Tarski, 1955) of monotone operators on complete lattices plays a fundamental role in classical logic programming (Lloyd, 1987). More precisely, given a positive (i.e., negation-free) classical normal program Π, the van Emden-Kowalski operator (Van Emden and Kowalski, 1976) T_Π is monotone on the complete lattice of all interpretations of Π ordered by set inclusion. Then, the theorem of Tarski and Knaster (Tarski, 1955) implies that T_Π has a least fixpoint, called the *least model* of Π (Van Emden and Kowalski, 1976) (see Example 3.2.3).

Approximation Theory (Denecker et al., 2000a, 2002, 2004) is an algebraic framework for studying fixpoints of monotone and nonmonotone lattice operators in terms of *monotone* approximations. In this sense, it can be considered as an extension of the fixpoint theory of Tarski and Knaster (Tarski, 1955) to the more general class of arbitrary lattice operators. The underlying algebraic structure is that of a bilattice (Ginsberg, 1988), which has been extensively studied, e.g., in (Fitting, 1991, 2002).

This chapter briefly summarizes essential notions and results given in (Denecker et al., 2004). The rest of this chapter is organized as follows. In Section 3.1 we recall basic definitions from Lattice Theory and introduce the fundamental notion of a (product) bilattice (Ginsberg, 1988).

In Section 3.2 we define, for a given lattice operator O defined on a complete lattice L, *monotone* approximations of O. The results of Tarski and Knaster imply that each such approximation $\mathcal{A}$ of O has a least fixpoint $k(\mathcal{A})$, called the *$\mathcal{A}$-Kripke-Kleene fixpoint*. The $\mathcal{A}$-Kripke-Kleene fixpoint approximates every fixpoint x of O, i.e., if $O(x) = x$ then $k(\mathcal{A})_1 \leq x \leq k(\mathcal{A})_2$, where $k(\mathcal{A})_j$, $1 \leq j \leq 2$, is the projection of $k(\mathcal{A})$ to the j-th component. For instance, given a classical normal program Π, the Fitting operator Φ_Π (Fitting, 1985) is an approximation of T_Π (Denecker et al., 2000a). Since fixpoints of T_Π are supported models (Apt et al., 1988) of Π, the Φ_Π-Kripke-Kleene fixpoint $k(\Phi_\Pi)$ approximates every supported model of Π (see Example 3.2.5 and Example 3.2.8).

In Section 3.3 we define, for a given approximation $\mathcal{A}$ of O, the *stable-revision operator* $\mathcal{A}^{\downarrow\uparrow}$ of $\mathcal{A}$. This operator is monotone and gives rise to *$\mathcal{A}$-stable fixpoints*. Again, by the results of Tarski and Knaster, the set of all $\mathcal{A}$-stable fixpoints has a least element $w(\mathcal{A})$, called the *$\mathcal{A}$-well-founded fixpoint*. By definition, $w(\mathcal{A})$ approximates every $\mathcal{A}$-stable fixpoint. An important result (Denecker et al., 2000a; Fitting, 2002) in classical logic programming is that (i) the Φ_Π-stable fixpoints characterize the 3-valued answer sets (Przymusinski, 1990) of Π, (ii) the exact Φ_Π-stable fixpoints characterize the answer sets (Gelfond and Lifschitz, 1991) of Π (see Example 3.3.7), and (iii) $w(\Phi_\Pi)$ characterizes the well-founded model (Van Gelder et al., 1991) of Π (see Example 3.3.10).

Finally, in Section 3.4 we define the *ultimate* approximation of O, which is the most precise approximation of O with respect to Approximation Theory. Denecker et al. (2004) showed that the ultimate approximation of O can be algebraically characterized in terms of O (see Theorem 3.4.1) and that it has the most precise Kripke-Kleene fixpoint, the most precise well-founded fixpoint, and the most stable fixpoints.

3.1 Bilattices

In this section we first recall basic notions from Lattice Theory (see, e.g., Davey and Priestley (2002)) and then define bilattices (Ginsberg, 1988) which play a fundamental role in Approximation Theory and logic programming (Denecker et al., 2000a, 2002, 2004; Fitting, 1991, 1994, 2002).

By a *partially ordered set* (or *poset*) we mean a set L together with a binary relation $\leq$ such that

1. $x \leq x$ for every x (*reflexivity*),
2. if $x \leq y$ and $y \leq x$ then $x = y$ for every x, y *(asymmetry)*,
3. if $x \leq y$ and $y \leq z$ then $x \leq z$ for every x, y, z *(transitivity)*.

We call a poset a *chain* if $x \leq y$ or $y \leq x$ for every x and y; moreover, if $x \not\leq x$ for every x (*irreflexivity*), then L is a *strict* partial order.

For $Y \subseteq L$, we say that $x \in L$ is an *upper bound* (resp., *lower bound*) of Y if $y \leq x$ (resp., $y \geq x$) for every $y \in Y$. The *least upper bound* (resp., *least lower bound*) of Y is denoted by $\inf Y$ (resp., $\sup Y$) if it exists. Moreover, we call a poset L *chain-complete* if $\sup C$ exists for every chain $C \subseteq L$.

A *lattice* is a poset $(L, \leq)$ such that $\inf\{x, y\}$ and $\sup\{x, y\}$ exist for every x, y. We say that L is *complete* if $\sup X$ and $\inf X$ exist for every $X \subseteq L$. Complete lattices have a *greatest element* $\top$ and a *least element* $\bot$.

Define, for $a, b \in L$,

$$a \wedge b = \inf\{a, b\} \quad \text{and} \quad a \vee b = \sup\{a, b\}.$$

Then, the following equations hold:

1. $x \wedge y = y \wedge x$, $x \vee y = y \vee x$ for every x, y *(commutativity)*,

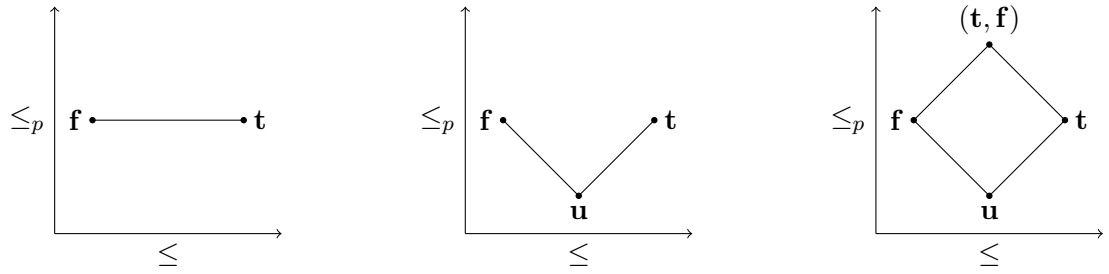

Figure 3.1: (Pelov, 2004, Figure 2.1) From left to right: (i) the complete lattice of Boolean truth-values 2, (ii) Kleene's 3-valued logic 3 ($= 2^c$), and (iii) the smallest non-trivial bilattice of Belnap's 4-valued logic 4 ($= 2^2$).

2. $x \wedge (y \wedge z) = (x \wedge y) \wedge z$, $x \vee (y \vee z) = (x \vee y) \vee z$ for every x, y, z *(associativity)*,
3. $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$, $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$ for every x, y, z *(absorption)*,
4. $x \wedge x = x$, $x \vee x = x$ for every x *(idempotence)*.

With the definition of $\wedge$ and $\vee$ given above, one can dually consider every lattice L either as a partially ordered set $(L, \leq)$ or as an algebraic structure $(L, \wedge, \vee)$. Furthermore, every lattice $(L, \leq)$ (resp., $(L, \wedge, \vee)$) induces the dual lattice of L given by $(L, \geq)$ (resp., $(L, \vee, \wedge)$). Notice that every lattice is inherently equal to its dual lattice, that is, they are isomorphic.

Example 3.1.1. A trivial complete lattice is the singleton $1 = \{a\}$ together with $a \leq a$. More interestingly is the complete lattice $2 = \{\mathbf{f}, \mathbf{t}\}$ of Boolean truth-values, where $\mathbf{f} \leq \mathbf{t}$ (see Section 2.2). Taking the product of 2 yields $2^2 = 4 = \{(\mathbf{t}, \mathbf{t}), (\mathbf{t}, \mathbf{f}), (\mathbf{f}, \mathbf{t}), (\mathbf{f}, \mathbf{f})\}$, where $(\mathbf{t}, \mathbf{t})$ (resp., $(\mathbf{f}, \mathbf{f})$) is identified with $\mathbf{t}$ (resp., $\mathbf{f}$). The set 4 is equipped with two natural orderings, both yielding the structure of a complete lattice (see Figure 3.1): (i) $\mathbf{f} \leq (\mathbf{f}, \mathbf{t}), (\mathbf{t}, \mathbf{f}) \leq \mathbf{t}$ (i.e., $(\mathbf{f}, \mathbf{t})$ and $(\mathbf{t}, \mathbf{f})$ are incomparable), and (ii) $(\mathbf{f}, \mathbf{t}) \leq_p \mathbf{f}, \mathbf{t} \leq_p (\mathbf{t}, \mathbf{f})$ (i.e., the truth-values $\mathbf{f}$ and $\mathbf{t}$ are incomparable).

Example 3.1.2. Let Π be a HEX program. The set of all interpretations $\mathcal{I}_\Pi$ ordered by $\subseteq$ is a complete lattice with least element $\emptyset$ and greatest element Λ_Π. We have seen in Section 2.2 that $(\mathcal{I}_\Pi, \subseteq)$ is isomorphic to the set of all evaluation functions 2^{Λ_Π} ordered by $\leq$. Hence, $(2^{\Lambda_\Pi}, \leq)$ is a complete lattice with least element $\langle\ .\ \rangle_I \equiv \mathbf{f}$ and greatest element $\langle\ .\ \rangle_I \equiv \mathbf{t}$.

We noted above that every lattice induces its dual lattice by "inverting" the ordering. Moreover, Example 3.1.1 shows that a complete lattice $(L, \leq)$ may additionally induce the product L^2 of L with two orderings $\leq$ and $\leq_p$ both naturally yielding the structure of a complete lattice. This motivates the following definition.

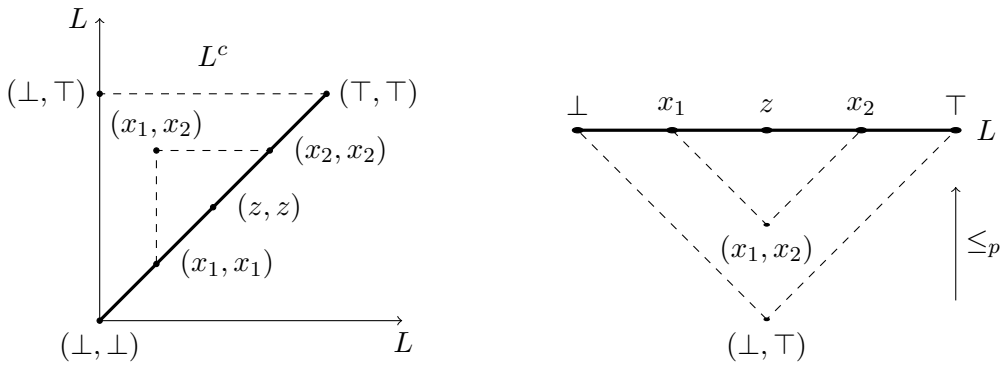

Figure 3.2: The chain-complete poset $L^c \subseteq L^2$ of a complete lattice L ordered by $\leq_p$, where (x_1, x_2) is an approximation of z, i.e., $(x_1, x_2) \leq_p z$. The right figure is obtained from the left by identifying the diagonal set of L^2 with L, and omitting the irrelevant part of L^2.

In the sequel L will denote a complete lattice together with a binary relation $\leq$. By the *product bilattice* (Ginsberg, 1988) of L we mean the set L^2 together with $\leq_p$ and $\leq$ defined, for each $(x_1, x_2), (y_1, y_2) \in L^2$, by[1]

1. $(x_1, x_2) \leq_p (y_1, y_2) \Leftrightarrow x_1 \leq y_1$ and $y_2 \leq x_2$ *(precision ordering)*,
2. $(x_1, x_2) \leq (y_1, y_2) \Leftrightarrow x_1 \leq y_1$ and $x_2 \leq y_2$.

Example 3.1.3. The bilattice $(4, \leq, \leq_p)$, $4 = 2^2$, of Example 3.1.1 (see Figure 3.1) is the smallest non-trivial bilattice (Ginsberg, 1988).

3.2 Approximations, Operators, Kripke-Kleene Fixpoints

Let L be a complete lattice ordered by $\leq$. We call elements $(x_1, x_2) \in L^2$ fulfilling $x_1 \leq x_2$ *consistent* and denote the set of all consistent elements of L^2 by L^c. Since the diagonal set of L^c, that is, the set $\{(x, x) \in L^c : x \in L\}$, is isomorphic to L (given the ordering $(x, x) \leq (y, y)$ if $x \leq y$), we identify every $x \in L$ with $(x, x) \in L^c$ and call such elements *exact*. Notice that exact elements are exactly the maximal elements with respect to $\leq_p$.

For $z \in L$ and $(x_1, x_2) \in L^c$, we have

$$(x_1, x_2) \leq_p z \Leftrightarrow x_1 \leq z \leq x_2,$$

and therefore imagine z to lie "between" x_1 and x_2, and say that (x_1, x_2) is an *approximation* of z (see Figure 3.2). Since two different maximal elements have no upper bound, the structure $(L^c, \leq_p)$ is not a lattice; however, it forms a chain-complete poset.

[1] In the literature, the orderings are usually denoted $\leq_t$ ("truth-ordering") and $\leq_k$ ("knowledge-ordering"), respectively (Ginsberg, 1988; Fitting, 1991, 2002).

Example 3.2.1. Consider the bilattice $4 = 2^2$ of Example 3.1.1 (see Figure 3.1). The pair $(\mathbf{f}, \mathbf{t})$ is consistent (i.e., $\mathbf{f} \leq \mathbf{t}$) and therefore approximates $\mathbf{t}$ and $\mathbf{f}$, whereas $(\mathbf{t}, \mathbf{f})$ is not consistent. The set of consistent pairs of 4 is given by $3 = 2^c = \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$, where $\mathbf{u} = (\mathbf{f}, \mathbf{t})$ corresponds to the truth-value *undefined*. The restriction of $\leq$ and $\leq_p$ of 4 to 3 yields the substructure $(3, \leq, \leq_p)$ of 4, where $(3, \leq)$ is a complete lattice (*Kleene's strong 3-valued logic* (Kleene, 1952)) and $(3, \leq_p)$ is a chain-complete poset.

Example 3.2.2. Let Π be a HEX program and let $\mathcal{I}_\Pi$ be the set of all interpretations of Π. In Example 3.1.2 we have seen that $\mathcal{I}_\Pi$ is a complete lattice. Hence, the construction of the product bilattice given above yields the bilattice $\mathcal{I}_\Pi^2$ ordered by

1. $(J_1, J_2) \subseteq_p (I_1, I_2) \Leftrightarrow J_1 \subseteq I_1$ and $I_2 \subseteq J_2$,
2. $(J_1, J_2) \subseteq (I_1, I_2) \Leftrightarrow J_1 \subseteq I_1$ and $J_2 \subseteq I_2$.

By restricting to consistent pairs of interpretations, that is, pairs of form (I_1, I_2) such that $I_1 \subseteq I_2$, we obtain the set $\mathcal{I}_\Pi^c$ of *3-valued interpretations*. In (I_1, I_2) we consider every $a \in I_1$ (resp., $a \notin I_2$) to be *true* (resp., *false*), and every $a \in I_2 - I_1$ to be *undefined* (see Section 4.1).

Furthermore, we define, for each x, y, the *interval* $[x, y]$ by

$$[x, y] = \{z \in L : x \leq z \leq y\}.$$

Clearly, $[x, y] \neq \emptyset$ iff $x \leq y$. Moreover, if $(L, \leq)$ is a complete lattice, then $\big([x, y], \leq_{[x,y]}\big)$ is a complete lattice, where $\leq_{[x,y]}$ is the restriction of $\leq$ to $[x, y]^2$ and $x \leq y$.

An *operator* on L is any function $O : L \longrightarrow L$. We say that O is *monotone* if for every $x_1, x_2 \in L$ such that $x \leq y$, $O(x) \leq O(y)$. Furthermore, 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 $\mathrm{lfp}(O)$.

Example 3.2.3 (van Emden-Kowalski operator)**.** Let Π be a classical normal program, and let $\mathcal{L}_\Pi = \Sigma_\Pi$ be the language of Π. The complete lattice of all interpretations of Π is $\mathcal{I}_\Pi = \mathfrak{P}(\Lambda_\Pi)$ ordered by set inclusion. Define the *van Emden-Kowalski operator* (Van Emden and Kowalski, 1976) of Π, for each $I \in \mathcal{I}_\Pi$, by

$$T_\Pi(I) = \{H(r) : r \in \Pi : I \models B(r)\}.$$

Then, since Π contains no disjunctive rules, T_Π is a lattice operator defined on $\mathcal{I}_\Pi$. If Π is positive (i.e., Π contains no negation as failure (Clark, 1978)), T_Π is monotone, and, by the theorem of Tarski and Knaster (Tarski, 1955), has a least fixpoint $\mathrm{lfp}(T_\Pi)$ which defines the *least model semantics* of Π (Van Emden and Kowalski, 1976).

If Π contains negation as failure, T_Π may be nonmonotone. Consider, e.g., the classical program $\Pi = \{p \leftarrow\sim q\}$. Then $T_\Pi(\emptyset) = \{p\}$, but $T_\Pi(\{q\}) = \emptyset$. However, pre-fixpoints of T_Π characterize models of Π, that is, every interpretation $I \in \mathcal{I}_\Pi$ is a model of Π iff $T_\Pi(I) \subseteq I$ (Apt et al., 1988). Moreover, we say that I is a *supported model of* Π (Apt et al., 1988) if I is a model of Π and every $a \in I$ has a justification in Π with respect to I, i.e., there exists a rule $r \in \Pi$ such that $H(r) = a$ and $I \models B(r)$. A well-known result in classical logic programming is that fixpoints of T_Π characterize supported models of Π, that is, I is supported iff $T_\Pi(I) = I$ (Apt et al., 1988).

Example 3.2.4. Let Π be the classical normal program consisting of the following propositional rules:

$$a \leftarrow\sim b,$$
$$b \leftarrow c,$$
$$c \leftarrow b.$$

The set of all interpretations of Π is given by $[\emptyset, \Lambda_\Pi] = \mathcal{I}_\Pi$. An easy computation shows that $T_\Pi(\{a\}) = \{a\}$ and $T_\Pi(\{b, c\}) = \{b, c\}$ are the only fixpoints of T_Π. Hence, $\{a\}$ and $\{b, c\}$ are the supported models of Π.

We now define approximations of O. 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 say that an operator $\mathcal{A} : L^c \longrightarrow L^c$ is an *approximation* of O if

1. $\mathcal{A}$ maps exact elements to exact elements,
2. $\mathcal{A}(x) = O(x)$ for each exact element x,[2] and
3. $\mathcal{A}$ 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.

Example 3.2.5 (Fitting operator)**.** Let Π be a classical normal program. Fitting (1985) defines the *Fitting operator* Φ_Π of Π on the chain-complete poset $(\mathcal{I}^c_\Pi, \subseteq_p)$ (see Example 3.2.2) by $\Phi_\Pi(I_1, I_2) = (I_1', I_2')$ such that

$$I_1' = \left\{H(r) : r \in \Pi : B^+(r) \subseteq I_1 \text{ and } B^-(r) \cap I_2 = \emptyset\right\},$$
$$I_2' = \left\{H(r) : r \in \Pi : B^+(r) \subseteq I_2 \text{ and } B^-(r) \cap I_1 = \emptyset\right\}.$$

It is easy to verify that Φ_Π is an approximation of T_Π (Denecker et al., 2000a).

Example 3.2.6. For the classical program Π of Example 3.2.4, we obtain $\Phi_\Pi(\{a\}, \{a\})_1 = \{a\} = \Phi_\Pi(\{a\}, \{a\})_2$, and $\Phi_\Pi(\{b, c\}, \{b, c\})_1 = \{b, c\} = \Phi_\Pi(\{b, c\}, \{b, c\})_2$.

We now define the Kripke-Kleene fixpoint of an approximation $\mathcal{A}$. Since $(L^c, \leq_p)$ is a chain-complete poset, and $\mathcal{A}$ is monotone on L^c, $\mathcal{A}$ has a least fixpoint $k(\mathcal{A})$,

$$k(\mathcal{A}) = \text{lfp}(\mathcal{A}),$$

which we call the *$\mathcal{A}$-Kripke-Kleene fixpoint*.

We say that $(a_1, a_2) \in L^c$ is *$\mathcal{A}$-reliable* if $(a_1, a_2) \leq_p \mathcal{A}(a_1, a_2)$, that is, if $\mathcal{A}(a_1, a_2)$ is "more precise" than (a_1, a_2). In this case we can imagine $\mathcal{A}$ to be a *revision* operator. Since $(\bot, \top)$ is $\mathcal{A}$-reliable and $\mathcal{A}$ is monotone, we can iterate $\mathcal{A}$ over $(\bot, \top)$ and obtain a transfinite sequence $(\bot, \top) \leq_p \mathcal{A}(\bot, \top) \leq_p \ldots \leq_p \mathcal{A}^\alpha(\bot, \top) \leq_p \ldots$, where α is an ordinal. The limit of this sequence coincides with the $\mathcal{A}$-Kripke-Kleene fixpoint $k(\mathcal{A})$.

Since $k(\mathcal{A})$ is the least fixpoint of $\mathcal{A}$ (with respect to $\leq_p$) and every fixpoint of O is a maximal fixpoint of $\mathcal{A}$ with respect to $\leq_p$, we have the following result.

[2] More precisely, $\mathcal{A}(x, x)_1 = O(x) = \mathcal{A}(x, x)_2$.

Proposition 3.2.7. *Let O be an operator on L, and let $\mathcal{A}$ be an approximation of O. Then, for every fixpoint x of O, $k(\mathcal{A}) \leq_p x$.*

Example 3.2.8 (Fitting semantics)**.** The Fitting operator Φ_Π (see Example 3.2.5) is an approximation of T_Π and has therefore a least fixpoint with respect to $\subseteq_p$, denoted $k(\Phi_\Pi)$, which we call the Φ_Π-*Kripke-Kleene model* (or *Fitting model*) of Π (Fitting, 1985). By Proposition 3.2.7, $k(\Phi_\Pi)$ approximates every fixpoint of T_Π, i.e., every supported model of Π (see Example 3.2.3).

Example 3.2.9. Let Π be given as in Example 3.2.4. Since the supported models $\{a\}$ and $\{b, c\}$ of Π are disjoint, we have $k(\Phi_\Pi) = (\emptyset, \{a, b, c\})$. Hence,

$$k(\Phi_\Pi)_1 = \emptyset \subseteq \{a\}, \{b, c\} \subseteq \{a, b, c\} = k(\Phi_\Pi)_2,$$

which shows that $k(\Phi_\Pi)$ indeed approximates every fixpoint of T_Π.

3.3 Stable revision operator and stable fixpoints

Let (a_1, a_2) be $\mathcal{A}$-reliable. The restriction of $\mathcal{A}(\,.\,, a_2)_1$ (resp., $\mathcal{A}(a_1, \,.\,)_2$) to $[\bot, a_2]$ (resp., $[a_1, \top]$) is a monotone operator on the complete lattice $([\bot, a_2], \leq)$ (resp., $([a_1, \top], \leq)$). Therefore, $\mathcal{A}(\,.\,, a_2)_1$ (resp., $\mathcal{A}(a_1, \,.\,)_2$) has a least fixpoint in $([\bot, a_2], \leq)$ (resp., $([a_1, \top], \leq)$).

Define the $\mathcal{A}$-*stable revision operator* by

$$\mathcal{A}^{\downarrow\uparrow}(a_1, a_2) = \left(\mathcal{A}^{\downarrow}(a_2), \mathcal{A}^{\uparrow}(a_1)\right),$$

where

$$\mathcal{A}^{\downarrow}(a_2) = \mathrm{lfp}\,(\mathcal{A}(\,.\,, a_2)_1) \quad \text{and} \quad \mathcal{A}^{\uparrow}(a_1) = \mathrm{lfp}\,(\mathcal{A}(a_1, \,.\,)_2)\,.$$

Roughly, $\mathcal{A}^{\downarrow}(a_2)$ underestimates every (minimal) fixpoint of O, whereas $\mathcal{A}^{\uparrow}(a_1)$ is an upper bound as tight as possible to the minimal fixpoints of O.

Proposition 3.3.1. *Let O be an operator on L, let $\mathcal{A}$ an approximation of O, and let (a_1, a_2) be $\mathcal{A}$-reliable. Then, for every (pre-)fixpoint x of O, if $x \leq a_2$ then $\mathcal{A}^{\downarrow}(a_2) \leq x$.*

In general, $\mathcal{A}^{\downarrow\uparrow}(a_1, a_2)$ has not to be a refinement of (a_1, a_2). We say that an $\mathcal{A}$-reliable pair (a_1, a_2) is $\mathcal{A}$-*prudent* if $a_1 \leq \mathcal{A}^{\downarrow}(a_2)$. The next proposition states that $\mathcal{A}^{\downarrow\uparrow}(a_1, a_2)$ is a refinement of $\mathcal{A}$-prudent pairs (a_1, a_2).

Proposition 3.3.2. *Let L be a complete lattice, let $\mathcal{A}$ be an approximating operator, and let $(a_1, a_2) \in L^c$ be $\mathcal{A}$-prudent. Then, $\mathcal{A}^{\downarrow\uparrow}(a_1, a_2)$ is $\mathcal{A}$-prudent and $(a_1, a_2) \leq_p \mathcal{A}^{\downarrow\uparrow}(a_1, a_2)$.*

Furthermore, for each $\mathcal{A}$-prudent pair (a_1, a_2), $\mathcal{A}^{\downarrow\uparrow}$ is more precise than $\mathcal{A}$.

Proposition 3.3.3. *Let L be a complete lattice, let $\mathcal{A}$ be an approximating operator, and let $(a_1, a_2) \in L^c$. If (a_1, a_2) is $\mathcal{A}$-prudent then $\mathcal{A}(a_1, a_2) \leq_p \mathcal{A}^{\downarrow\uparrow}(a_1, a_2)$.*

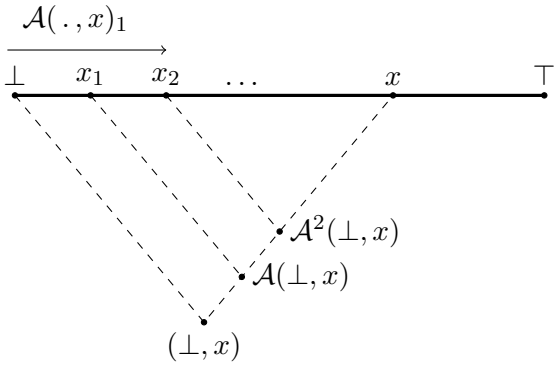

Figure 3.3: Computation of $\mathcal{A}^{\downarrow}(x) = \mathrm{lfp}(\mathcal{A}(\,.\,,x)_1)$.

Example 3.3.4. Let Π be the classical normal program of Example 3.2.4. Since $\Phi_\Pi(\{c\},\{c,b\}) = (\{c,b\},\{c,b\})$ and $(\{c\},\{c,b\}) \subseteq_p (\{c,b\},\{c,b\})$, we conclude that $(\{c\},\{c,b\})$ is Φ_Π-reliable. However, since $\Phi_\Pi^{\downarrow}(\{c,b\}) = \emptyset$, $\{c,b\}$ is not Φ_Π-prudent. In contrast, $\Phi_\Pi(\emptyset,\{a\}) = (\{a\},\{a\})$ and $\Phi_\Pi^{\downarrow}(\{a\}) = \{a\}$ shows that $(\emptyset,\{a\})$ is Φ_Π-prudent.

The stable revision operator $\mathcal{A}^{\downarrow\uparrow}$ has fixpoints and a least fixpoint on the chain-complete poset of $\mathcal{A}$-prudent pairs ordered by $\leq_p$. We say that an $\mathcal{A}$-reliable approximation (a_1,a_2) is an *$\mathcal{A}$-stable fixpoint* if (a_1,a_2) is a fixpoint of $\mathcal{A}^{\downarrow\uparrow}$. Furthermore, if $\mathcal{A}$ is an approximation of O, we call every exact $\mathcal{A}$-stable fixpoint (x,x) (notice that x is then a fixpoint of O) an *$\mathcal{A}$-stable fixpoint of O*.

Proposition 3.3.5. *Let L be a complete lattice, let $\mathcal{A}$ be an approximating operator, and let $(x_1,x_2) \in L^c$. If (x_1,x_2) is an $\mathcal{A}$-stable fixpoint, then (x_1,x_2) is a fixpoint of $\mathcal{A}$.*

The next proposition states that exact $\mathcal{A}$-stable fixpoints of O can be characterized by fixpoints of $\mathcal{A}^{\downarrow}$.

Proposition 3.3.6. *Let $\mathcal{A}$ be an approximating operator of O, and let $x \in L$. Then the following conditions are equivalent:*

1. *x is an $\mathcal{A}$-stable fixpoint of O,*
2. *x is a fixpoint of O and $\mathcal{A}^{\downarrow}(x) = x$.*

Example 3.3.7 (Answer-set semantics)**.** Let Π be a classical normal program, let $\mathcal{L}_\Pi = \Sigma_\Pi$ be the language of Π, and let $I \in \mathcal{I}_\Pi$ be an interpretation. Define the *Gelfond-Lifschitz reduct* (Gelfond and Lifschitz, 1988) of Π with respect to I by

$$\Pi^I = \{H(r) \leftarrow B^+(r) : r \in \Pi : I \cap B^-(r) = \emptyset\}.$$

Intuitively, we compute the reduct Π^I of Π by (i) deleting every rule in Π where the negative body is not compatible with I, and (ii) deleting the negative body of the remaining rules. This transformation yields a definite program Π^I which has a least model, given by $\mathrm{lfp}(T_{\Pi^I})$.

Furthermore, define the *Gelfond-Lifschitz operator* (Gelfond and Lifschitz, 1988) of Π, for each $I \in \mathcal{I}_\Pi$, by

$$\Gamma_\Pi(I) = \mathrm{lfp}(T_{\Pi^I}).$$

We say that I is an *answer set*[3] (Gelfond and Lifschitz, 1988) of Π if $\Gamma_\Pi(I) = I$. Notice that every classical normal program Π may have no answer sets, exactly one answer set, or more than one answer set. However, every answer set is a minimal model of Π and a fixpoint of T_Π.

The restriction $\Phi_\Pi(\,.\,, I)_1$ with respect to I "simulates" the operator T_{Π^I}, i.e., $\Phi_\Pi(\,.\,, I)_1 = T_{\Pi^I}$. Hence, we can reformulate the definition of Γ_Π in terms of Φ_Π by

$$\Gamma_\Pi(I) = \mathrm{lfp}\,(\Phi_\Pi(\,.\,, I)_1) = \Phi^{\downarrow}_\Pi(I).$$

That is, I is an answer set of Π iff I is an exact Φ_Π-stable fixpoint of T_Π (Denecker et al., 2000a). It is worth mentioning that for classical normal programs, FLP-answer-set semantics (Faber et al., 2004, 2011), and traditional answer-set semantics (Gelfond and Lifschitz, 1991) coincide.[4]

Example 3.3.8. Reconsider the classical normal program Π of Example 3.2.4 consisting of the following rules:

$$a \leftarrow\sim b$$
$$b \leftarrow c$$
$$c \leftarrow b$$

Recall that $\{a\}$ and $\{b, c\}$ are (minimal) fixpoints of T_Π. For $I = \{a\}$ we have $\Pi^I = \{a \leftarrow, b \leftarrow c, c \leftarrow b\}$. We compute the least fixpoint of T_{Π^I} by $T_{\Pi^I}(\emptyset) = \{a\}$, and $T_{\Pi^I}(\{a\}) = \{a\}$. That is, $\Gamma_\Pi(\{a\}) = \{a\}$ which shows that $\{a\}$ is an answer set of Π. In Example 3.3.7 we have seen that Γ_Π coincides with $\Phi^{\downarrow}_\Pi$. Indeed, since $\Phi_\Pi(\emptyset, \{a\})_1 = \{a\}$, and $\Phi_\Pi(\{a\}, \{a\})_1 = \{a\}$, we have $\Phi^{\downarrow}_\Pi(\{a\}) = \{a\}$ which shows that $\{a\}$ is a Φ_Π-stable fixpoint.

In contrast, for $I' = \{b, c\}$, and $\Pi^{I'} = \{b \leftarrow c, c \leftarrow b\}$, we have $T_{\Pi^{I'}}(\emptyset) = \emptyset$ and, hence, $\Gamma_\Pi(\{b, c\}) = \emptyset$ which shows that $\{b, c\}$ is not an answer set of Π. Likewise, $\Phi_\Pi(\emptyset, \{b, c\})_1 = \emptyset$ yields $\Phi^{\downarrow}_\Pi(\{b, c\}) = \emptyset$ which shows that $\{b, c\}$ is not a Φ_Π-stable fixpoint.

Since $\mathcal{A}^{\downarrow\uparrow}$ has a least fixpoint (with respect to $\leq_p$), we can define the *$\mathcal{A}$-well-founded fixpoint* by

$$w(\mathcal{A}) = \mathrm{lfp}(\mathcal{A}^{\downarrow\uparrow}).$$

We can compute the $\mathcal{A}$-well-founded fixpoint by iterating $\mathcal{A}^{\downarrow\uparrow}$, starting at $(\bot, \top)$, until a fixpoint is reached. More precisely, $w(\mathcal{A})$ is the limit of the sequence $((a^\alpha, b^\alpha))_{\alpha \geq 0}$ of elements of L^c, where

[3] In the literature the term "answer set" (Gelfond and Lifschitz, 1991) is reserved for semantics of *extended* logic programs (i.e., programs with strong negation, see Example 2.1.3). However, to be consistent with following terminology, we use the term "answer set" instead of "stable model" here.

[4] In fact, there exist at least thirteen characterizations of answer-set semantics Lifschitz (2010).

1. $\left(a^0, b^0\right) = (\bot, \top)$
2. $a^{\alpha+1} = \mathcal{A}^{\downarrow}\left(b^{\alpha}\right)$ and $b^{\alpha+1} = \mathcal{A}^{\uparrow}\left(a^{\alpha}\right)$
3. $\left(a^{\alpha}, b^{\alpha}\right) = \bigvee \left\{\left(a^{\beta}, b^{\beta}\right) : \beta < \alpha\right\}$ for every limit ordinal α.

Clearly, $w(\mathcal{A}) \leq_p (a_1, a_2)$ for every $\mathcal{A}$-stable fixpoint (a_1, a_2). Moreover, the well-founded fixpoint is more precise than the Kripke-Kleene fixpoint. Indeed, we have the following result.

Proposition 3.3.9. *Let O be an operator. Then, for every approximation $\mathcal{A}$ of O, $k(\mathcal{A}) \leq_p w(\mathcal{A})$.*

Example 3.3.10 (Well-founded semantics)**.** Let Π be a classical normal program, and let $\mathcal{L}_\Pi = \Sigma_\Pi$ be the language of Π. Let $(I_1, I_2) \in \mathcal{I}^c_\Pi$. We say that $U \subseteq \Sigma_\Pi$ is an *unfounded set* (Van Gelder et al., 1991) of Π (relative to (I_1, I_2)), if for every $a \in U$, and every $r \in \Pi$ such that $H(r) = a$,

1. $b \notin I_2$ or $b \in U$ for some $b \in B^+(r)$, or
2. $b \in I_1$ for some $b \in B^-(r)$.

The union of unfounded sets is again an unfounded set. Hence, for every $(I_1, I_2) \in \mathcal{I}^c_\Pi$ there exists a greatest unfounded set relative to (I_1, I_2), denoted $\mathcal{GUS}_\Pi(I_1, I_2)$. Intuitively, if (I_1, I_2) is compatible with Π, then $\mathcal{GUS}_\Pi(I_1, I_2)$ contains all atoms which can be safely switched to *false* such that the resulting 3-valued interpretation is still compatible with Π (Eiter et al., 2009b).

Moreover, define for every (I_1, I_2) the *partial van Emden-Kowalski operator* of Π by

$$\mathcal{Q}_\Pi(I_1, I_2) = \left\{H(r) : r \in \Pi : B^+(r) \subseteq I_1 \text{ and } B^-(r) \cap I_2 = \emptyset\right\}. \tag{3.1}$$

Similar to T_Π, $\mathcal{Q}_\Pi$ derives all immediate consequences of (I_1, I_2), assuming that every $a \in I_1$ (resp., $a \notin I_2$) is *true* (resp., *false*). Then, define the *(classical) well-founded operator* (Van Gelder et al., 1991) of Π, for each $(I_1, I_2) \in \mathcal{I}^c_\Pi$, by $\mathcal{W}_\Pi(I_1, I_2) = (I_1', I_2')$, where

$$\begin{aligned} I_1' &= \mathcal{Q}_\Pi(I_1, I_2), \\ I_2' &= HB_\Pi - \mathcal{GUS}_\Pi(I_1, I_2). \end{aligned}$$

The operator $\mathcal{W}_\Pi$ is monotone with respect to $\subseteq_p$. Hence, by the theorem of Tarski and Knaster (Tarski, 1955), $\mathcal{W}_\Pi$ has a least fixpoint given by $\mathrm{lfp}(\mathcal{W}_\Pi)$. Define the *(3-valued) well-founded model* of Π (Van Gelder et al., 1991) by

$$w(\Pi) = \mathrm{lfp}\left(\mathcal{W}_\Pi\right).$$

We say that $a \in HB_\Pi$ is *well-founded* (resp., *unfounded*) if $a \in w(\Pi)_1$ (resp., $a \notin w(\Pi)_2$), and we say that a is *undefined* if $a \in w(\Pi)_2 - w(\Pi)_1$. Every classical normal program Π has exactly one (3-valued) well-founded model $w(\Pi)$, and if $w(\Pi)$ is exact, then $w(\Pi)$ coincides with the single answer set of Π (Van Gelder et al., 1991). Moreover, $w(\Phi^{hex}_\Pi)$ characterizes the well-founded model of Π, i.e., $w(\Phi^{hex}_\Pi) = w(\Pi)$ (Denecker et al., 2000a).

Example 3.3.11. We compute the well-founded model of the classical normal program Π defined in Example 3.2.4. In Example 3.3.7 we have seen that $I = \{a\}$ is the only answer set, and the only Φ_Π-stable fixpoint of T_Π. Hence, we expect the well-founded model $w(\Pi)$ (resp., $w(\Phi_\Pi)$) to coincide with I (see Example 3.3.10).

Clearly, $\mathcal{Q}_\Pi(\emptyset, \{a\}) = \emptyset$ and it is easy to verify that $\{b, c\}$ is the greatest unfounded set with respect to $(\emptyset, \{a\})$, that is, $\mathcal{GUS}_\Pi(\emptyset) = \{b, c\}$. Hence, the first iteration yields $\mathcal{W}_\Pi(\emptyset, \{a\}) = (\emptyset, \{a\})$. In the second iteration we have $\mathcal{Q}_\Pi(\emptyset, \{a\}) = \{a\}$ and $\mathcal{GUS}_\Pi(\emptyset, \{a\}) = \{b, c\}$ which is equivalent to $\mathcal{W}_\Pi(\emptyset, \{a\}) = (\{a\}, \{a\})$. Finally, the third iteration yields the fixpoint $\mathcal{W}_\Pi(\{a\}, \{a\}) = (\{a\}, \{a\})$. Hence, $w(\Pi) = (\{a\}, \{a\})$ is the (2-valued) well-founded model of Π.

Likewise, the following computation, iterating the stable revision operator $\Phi_\Pi^{\downarrow\uparrow}$ by

$$\begin{aligned}
\Phi_\Pi^{\downarrow\uparrow}(\emptyset, \{a\}) &= \left(\Phi_\Pi^{\downarrow}(\{a, b, c\}), \Phi_\Pi^{\uparrow}(\emptyset)\right) = (\emptyset, \{a\}) \\
\Phi_\Pi^{\downarrow\uparrow}(\emptyset, \{a\}) &= \left(\Phi_\Pi^{\downarrow}(\{a\}), \Phi_\Pi^{\uparrow}(\emptyset)\right) = (\{a\}, \{a\}) \\
\Phi_\Pi^{\downarrow\uparrow}(\{a\}, \{a\}) &= \left(\Phi_\Pi^{\downarrow}(\{a\}), \Phi_\Pi^{\uparrow}(\{a\})\right) = (\{a\}, \{a\})
\end{aligned}$$

shows $w(\Phi_\Pi) = (\{a\}, \{a\}) = w(\Pi)$. Since $(\{a\}, \{a\})$ is exact (i.e., 2-valued), we identify it with the answer set and Φ_Π-stable fixpoint $\{a\}$ (see Example 3.3.7).

3.4 Ultimate approximations and ultimate fixpoints

In this section we define, for a given operator O, the most precise approximation of O (Denecker et al., 2004, Chapter 5).

Let L be a complete lattice and let O be an operator on L. Furthermore, let $\mathcal{A}$ and B be approximations of O. We say that $\mathcal{A}$ is *less precise* than B, in symbols $\mathcal{A} \leq_p \mathcal{B}$, if $\mathcal{A}(x_1, x_2) \leq_p \mathcal{B}(x_1, x_2)$ for each consistent pair (x_1, x_2). The set of all approximations ordered by $\leq_p$ is a complete lattice with least element

$$(x_1, x_2) \mapsto \begin{cases} (O(x), O(x)) & \text{if } x = x_1 = x_2, \\ (\bot, \top) & \text{otherwise,} \end{cases}$$

and greatest element $\mathcal{O}$, the *ultimate approximation* of O. The next theorem gives an algebraic characterization of $\mathcal{O}$.

Theorem 3.4.1. *Let L be a complete lattice, and let O be an operator on L. The ultimate approximation $\mathcal{O}$ of O is given, for every $(x_1, x_2) \in L^c$, by*

$$\mathcal{O}(x_1, x_2) = \left(\bigwedge_{x_1 \leq z \leq x_2} O(z), \bigvee_{x_1 \leq z \leq x_2} O(z) \right). \tag{3.2}$$

Since $\mathcal{O}$ is an approximation of O, we can define (i) the *ultimate Kripke-Kleene fixpoint* of O by $k(\mathcal{O})$ (see Section 3.2), (ii) the *ultimate well-founded fixpoint* of O by $w(\mathcal{O})$, and (iii) the *ultimate stable fixpoints* of O by the $\mathcal{O}$-stable fixpoints (see Section 3.3).

Example 3.4.2. Let Π be a classical normal program. The Fitting operator Φ_Π (see Example 3.2.5) is not the most precise approximation of T_Π. Following the definition given in Theorem 3.4.1, we obtain the ultimate approximation $\mathcal{T}_\Pi$ of T_Π, defined on $\mathcal{I}_\Pi^c$ by

$$\mathcal{T}_\Pi(I_1, I_2) = \left(\bigcap_{I_1 \subseteq J \subseteq I_2} T_\Pi(J), \bigcup_{I_1 \subseteq J \subseteq I_2} T_\Pi(J) \right).$$

This gives rise to the (i) ultimate Kripke-Kleene semantics, (ii) ultimate 3-valued answer-set semantics, and (iii) ultimate well-founded semantics of Π.

Example 3.4.3. Reconsider the classical normal program Π of Example 3.2.4 consisting of the following rules:

$$a \leftarrow\sim c$$
$$b \leftarrow c$$
$$c \leftarrow b.$$

We have seen in Example 3.2.4 that $I = \{a\}$ is the only answer set of Π. We now show that I is an ultimate answer set of T_Π by computing $\mathcal{T}_\Pi^{\downarrow}(I)$ as follows:

$$\mathcal{T}_\Pi(\emptyset, I)_1 = T_\Pi(\emptyset) \cap T_\Pi(I) = \{a\}$$
$$\mathcal{T}_\Pi(I, I)_1 = T_\Pi(I) = \{a\}.$$

That is, we have $\mathcal{T}_\Pi^{\downarrow}(I) = I$ which shows that I is indeed an ultimate answer set of Π.

The following proposition summarizes some fundamental relations concerning the ultimate approximation of O.

Proposition 3.4.4. *Let L be a complete lattice, and let O be an operator on L. For every approximation $\mathcal{A}$ of O:*

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

Example 3.4.5 (Denecker et al. (2004))**.** Let Π be the classical normal program consisting of the following rules:

$$a \leftarrow a$$
$$a \leftarrow\sim a.$$

Then, $\Phi_\Pi^\downarrow(\{a\}) = \emptyset$ and $\{a\}$ is therefore not a Φ_Π-stable fixpoint. In contrast, we have

$$\begin{aligned}\mathcal{T}_\Pi(\emptyset, \{a\})_1 &= T_\Pi(\emptyset) \cap T_\Pi(\{a\}) = \{a\}\\ \mathcal{T}_\Pi(\{a\}, \{a\})_1 &= T_\Pi(\{a\}) = \{a\}.\end{aligned}$$

That is, $\mathcal{T}_\Pi^\downarrow(\{a\}) = \{a\}$ which shows that $\{a\}$ is an ultimate answer set of Π.

The next result states that if O is monotone, then for computing $\mathcal{O}(x_1, x_2)$, we have to compute O only on the boundaries of $[x_1, x_2]$.

Proposition 3.4.6. *Let O be a monotone operator on the complete lattice L. Then, for every $(x_1, x_2) \in L^c$, $\mathcal{O}(x_1, x_2) = (O(x_1), O(x_2))$.*

Example 3.4.7. Let Π be the negation-free classical normal program consisting of the following propositional rules:

$$\begin{aligned}a &\leftarrow\\ b &\leftarrow a.\end{aligned}$$

Since Π contains no negation as failure, T_Π is monotone (see Example 3.2.3). Hence, by Proposition 3.4.6, the following computation shows that $I = \{a, b\}$ is an ultimate stable fixpoint of T_Π:

$$\begin{aligned}\mathcal{T}_\Pi(\emptyset, I)_1 &= T_\Pi(\emptyset) = \{a\}\\ \mathcal{T}_\Pi(\{a\}, I)_1 &= T_\Pi(\{a\}) = I\\ \mathcal{T}_\Pi(I, I)_1 &= T_\Pi(I) = I.\end{aligned}$$

That is, $\mathcal{T}_\Pi^\downarrow(I) = I$ which shows that $\{a, b\}$ is an ultimate answer set Π.

CHAPTER 4

Fixpoint Semantics of Normal HEX Programs

In this chapter we uniformly extend major semantics of classical normal programs (see Example 3.2.8, Example 3.3.7, and Example 3.3.10) to the class of normal (i.e., disjunction-free) HEX programs.

In particular, in Section 4.1 we define, for every given normal HEX program Π, the extended van Emden-Kowalski operator T_Π^{hex} of Π and show (i) that pre-fixpoints of T_Π^{hex} characterize models of Π, and (ii) that every FLP-answer set is a (minimal) fixpoint of T_Π^{hex}. Then, we define 3-valued interpretations of Π and extend evaluation functions to the 3-valued case. This construction directly yields the extended Fitting operator Φ_Π^{hex} as an extension of Φ_Π. We show that Φ_Π^{hex} is an approximation of T_Π^{hex}, and therefore conclude that Approximation Theory is applicable. This gives rise to the (i) Kripke-Kleene or Fitting semantics (Fitting, 1985) defined in Section 4.1, (ii) 3-valued answer-set semantics (Przymusinski, 1990) defined in Section 4.2, and (iii) well-founded semantics (Van Gelder et al., 1991) defined in Section 4.3 of Π (Figure 4.4 illustrates the relationships between these different semantics).

Moreover, by instantiating Theorem 3.4.1, we define in Section 4.4 the ultimate approximation $\mathcal{T}_\Pi^{hex}$ of T_Π^{hex} and obtain ultimate Kripke-Kleene semantics, ultimate 3-valued answer-set semantics, and ultimate well-founded semantics. These ultimate semantics are the most precise one with respect to Approximation Theory.

4.1 Kripke-Kleene semantics

In this section we define Kripke-Kleene semantics of normal (i.e., disjunction-free) HEX programs by applying the part of Approximation Theory prestend in Section 3.2.

Definition 4.1.1 (Extended van Emden-Kowalski operator)**.** Let Π be a normal HEX program. We define the *extended van Emden-Kowalski operator* of Π, for every interpretation $I \in \mathcal{I}_\Pi$, by

$$T_\Pi^{hex}(I) = \{H(r) : r \in \Pi : I \models B(r)\}.$$

As in the classical case, the result of applying T_Π to an interpretation I is an interpretation $J \in \mathcal{I}_\Pi$ consisting of each atom $a \in \Lambda_\Pi$ for which there exists a rule $r \in \Pi$ with $H(r) = a$ and $I \models B(r)$. Notice that since Π contains no disjunctive rules, T_Π is in fact a lattice operator defined on the complete lattice $\mathcal{I}_\Pi$ ordered by set inclusion.

The next result states that if Π is a classical normal program, then T_Π^{hex} and T_Π (Van Emden and Kowalski, 1976) (see Example 3.2.3) coincide.

Proposition 4.1.2. *Let Π be a classical normal program. Then, $T_\Pi^{hex} = T_\Pi$.*

As for classical normal programs (Apt et al., 1988) (see Example 3.2.3), the following proposition states that we can characterize the models of Π by the pre-fixpoints of T_Π^{hex}.

Proposition 4.1.3. *Let Π be a normal HEX program, and let $I \in \mathcal{I}_\Pi$. Then, I is a model of Π iff $T_\Pi^{hex}(I) \subseteq I$.*

The next result shows that we can relate minimal fixpoints of T_Π^{hex} and FLP-answer sets.

Corollary 4.1.4. *Let Π be a normal HEX program, and let $I \in \mathcal{I}_\Pi$. If I is an FLP-answer set of Π, then I is a minimal fixpoint of T_Π^{hex}.*

Proof. Follows directly from Proposition 2.2.4 and Proposition 4.1.3. □

Example 4.1.5. Let Π be the normal HEX program consisting of the following rules:

$$a \leftarrow b^{\#}$$
$$b \leftarrow a^{\#}$$

where $a^{\#}$ and $b^{\#}$ are defined as in Example 2.2.1. Then, $T_\Pi^{hex}(\{a, b\}) = \{a, b\}$ shows that $\{a, b\}$ is a fixpoint of T_Π^{hex} and, therefore, a model of Π. However, since $T_\Pi^{hex}(\emptyset) = \emptyset$, it is not a minimal fixpoint of Π.

We now define 3-valued interpretations and 3-valued evaluation functions (Denecker et al., 2004). Let $I \in \mathcal{I}_\Pi$. Recall from Section 2.2 that for each atom $a \in \Lambda_\Pi$, the 2-valued evaluation function $\langle\ .\ \rangle_I$ with respect to I is defined by

$$\langle\, a \,\rangle_I = \begin{cases} \mathbf{t} & \text{if } a \in I, \\ \mathbf{f} & \text{if } a \notin I, \end{cases} \tag{4.1}$$

where $2 = \{\mathbf{f}, \mathbf{t}\}$ and $\mathbf{f} \leq \mathbf{t}$. A *(consistent) 3-valued interpretation* of Π is a pair $(I_1, I_2) \in \mathcal{I}_\Pi^2$ of interpretations such that $I_1 \subseteq I_2$. We denote the set of all such elements by $\mathcal{I}_\Pi^c$ (see Figure 3.2) and define the following two orderings (see Example 3.2.2):

1. $(J_1, J_2) \subseteq_p (I_1, I_2) \Leftrightarrow J_1 \subseteq I_1$ and $I_2 \subseteq J_2$,
2. $(J_1, J_2) \subseteq (I_1, I_2) \Leftrightarrow J_1 \subseteq I_1$ and $J_2 \subseteq I_2$.

Since we identify each exact element (I, I) with the 2-valued interpretation I (see Section 3.2), we have

$$(I_1, I_2) \subseteq_p I \Leftrightarrow I_1 \subseteq I \subseteq I_2,$$

and in this case imagine (I_1, I_2) to be an *approximation* of I in the following sense: I_1 contains every atom $a \in I$ which is definitely *true* with respect to Π, and $\Lambda_\Pi - I_2$ contains every atom $a \notin I$ which is definitely *false*, whereas one remains agnostic for every $a \in I_2 - I_1$ (i.e., a has the truth-value *undefined*). Hence, every 3-valued interpretation $(I_1, I_2) \in \mathcal{I}_\Pi^c$ corresponds to a *3-valued evaluation function (with respect to* (I_1, I_2)*)* $\langle\ .\ \rangle_{(I_1,I_2)}$ defined, for each atom $a \in \Lambda_\Pi$, by

$$\langle\, a \,\rangle_{(I_1,I_2)} = \begin{cases} \mathbf{t} & \text{if } a \in I_1, \\ \mathbf{f} & \text{if } a \notin I_2, \\ \mathbf{u} & \text{otherwise,} \end{cases} \tag{4.2}$$

where $\mathbf{u}$ denotes the truth-value *undefined*. Notice that for every exact interpretation (I, I),

$$\langle\, a \,\rangle_{(I,I)} = \langle\, a \,\rangle_I\,. \tag{4.3}$$

Let $\mathsf{3} = \mathsf{2} \cup \{\mathbf{u}\}$ and extend $\leq$ to $\mathsf{3}$ by $\mathbf{f} \leq \mathbf{u} \leq \mathbf{t}$. Conversely to the construction above, given a 3-valued evaluation function $\langle\ .\ \rangle$, define

$$I_1 = \{a \in \Lambda_\Pi : \langle\, a \,\rangle = \mathbf{t}\} \quad \text{and} \quad I_2 = \{a \in \Lambda_\Pi : \langle\, a \,\rangle \geq \mathbf{u}\}.$$

Then, $I_1 \subseteq I_2$ and (I_1, I_2) is a 3-valued interpretation. As in the 2-valued case (see Section 2.2), the correspondence is one-one, that is, each consistent pair (I_1, I_2) can be identified with the 3-valued evaluation function $\langle\ .\ \rangle_{(I_1,I_2)}$ and vice versa. We denote the set of all 3-valued evaluation functions by $\mathsf{3}^{\Lambda_\Pi}$, and order $\mathsf{3}^{\Lambda_\Pi}$ by $\leq$ pointwise. Then, $(\mathsf{3}^{\Lambda_\Pi}, \leq)$ is isomorphic to $(\mathcal{I}_\Pi^c, \subseteq)$.

Notice that on $\mathcal{I}_\Pi^c$ we additionally have the precision ordering $\subseteq_p$. Therefore, define $\leq_p$ on $\mathsf{3}$ by $\mathbf{u} \leq_p \mathbf{f}$, $\mathbf{u} \leq_p \mathbf{t}$ (see Example 3.1.1 and Figure 3.1) and extend the ordering to $\mathsf{3}^{\Lambda_\Pi}$ pointwise, that is,

$$\langle\ .\ \rangle_1 \leq_p \langle\ .\ \rangle_2 \Leftrightarrow \langle\, a \,\rangle_1 \leq_p \langle\, a \,\rangle_2 \quad \text{for each atom } a.$$

Then, $(\mathsf{3}^{\Lambda_\Pi}, \leq, \leq_p)$ is isomorphic to $(\mathcal{I}_\Pi^c, \subseteq, \subseteq_p)$.

Let $(I_1, I_2) \in \mathcal{I}_\Pi^c$. We now extend $\langle\ .\ \rangle_{(I_1,I_2)}$ to external atoms, i.e., we extend 3-valued evaluation functions from Λ_Π to HB_Π.

Definition 4.1.6. Let Π be a (normal or disjunctive) HEX program. Define, for each external atom $f^\#[\mathbf{i}](\mathbf{o}) \in \Lambda_\Pi^\#$,

$$\left\langle f^\#[\mathbf{i}](\mathbf{o}) \right\rangle_{(I_1,I_2)} = \begin{cases} \mathbf{t} & \text{if } f(J, \mathbf{i}, \mathbf{o}) = \mathbf{t} \text{ for each } J \in [I_1, I_2], \\ \mathbf{f} & \text{if } f(J, \mathbf{i}, \mathbf{o}) = \mathbf{f} \text{ for each } J \in [I_1, I_2], \\ \mathbf{u} & \text{otherwise.} \end{cases} \tag{4.4}$$

Since $[I_1, I_2]$ is not empty, (4.4) is well-defined. Roughly, for $f^{\#}[\mathbf{i}](\mathbf{o})$ to be *true* (resp., *false*) with respect to (I_1, I_2), it has to be *true* (resp., *false*) for each interpretation J approximated by (I_1, I_2). Since an external atom represents an *arbitrary* Boolean function (see Section 2.2), external atoms cannot be treated in the same way as ordinary atoms. More precisely, every atom $a \in \Lambda_\Pi$ obviously fulfills the following monotonicity property: if $a \in I$ for some interpretation $I \in \mathcal{I}_\Pi$, then $a \in I'$ for every $I \subseteq I'$. In contrast, external atoms may be nonmonotone and therefore it is not sufficient to evaluate $\left\langle f^{\#}[\mathbf{i}](\mathbf{o}) \right\rangle_{(I_1,I_2)}$ only on the boundaries I_1 and I_2. However, for each exact interpretation $(I, I) \in \mathcal{I}_\Pi^c$, we have (see (4.3))

$$\left\langle f^{\#}[\mathbf{i}](\mathbf{o}) \right\rangle_{(I,I)} = \left\langle f^{\#}[\mathbf{i}](\mathbf{o}) \right\rangle_{I} = f(I, \mathbf{i}, \mathbf{o}). \tag{4.5}$$

Example 4.1.7. Consider the external atom $a^{\#}$ of Example 2.2.1. Since $a^{\#}$ is monotone (see Section 5.1), we have

$$\left\langle a^{\#} \right\rangle_{(I_1,I_2)} = \begin{cases} \mathbf{t} & \text{if } a \in I_1, \\ \mathbf{f} & \text{if } a \notin I_2, \\ \mathbf{u} & \text{otherwise.} \end{cases}$$

For instance, we have $\left\langle a^{\#} \right\rangle_{(\{a\},\{a,b\})} = \mathbf{t}$, $\left\langle a^{\#} \right\rangle_{(\emptyset,\{a\})} = \mathbf{u}$, and $\left\langle a^{\#} \right\rangle_{(\emptyset,\{b\})} = \mathbf{f}$.

Notice that, as in the 2-valued case (see Section 2.2), the correspondence between 3^{HB_Π} (i.e., the set of all *total* 3-valued evaluation functions) and $\mathcal{I}_\Pi^c$ is no longer one-one.

In Corollary 4.1.4 we have seen that FLP-answer sets of Π are minimal fixpoints of T_Π^{hex}. However, since T_Π^{hex} is, in general, nonmonotone, we cannot derive FLP-answer sets by a bottom-up iteration of T_Π^{hex}. We now define an operator which is monotone with respect to $\subseteq_p$ and coincides with T_Π^{hex} on $\mathcal{I}_\Pi$ (i.e., is an approximation of T_Π^{hex}, see Section 3.2). Approximation Theory then gives us Φ_Π^{hex}-Kripke-Kleene semantics.

Definition 4.1.8 (Extended Fitting operator)**.** Let Π be a normal HEX program. Define the *extended Fitting operator* of Π, for each $(I_1, I_2) \in \mathcal{I}_\Pi^c$, by

$$\Phi_\Pi^{hex}(I_1, I_2) = \left(I_1', I_2'\right),$$

such that

$$I_1' = \left\{ H(r) : r \in \Pi : \langle B(r) \rangle_{(I_1,I_2)} = \mathbf{t} \right\},$$
$$I_2' = \left\{ H(r) : r \in \Pi : \langle B(r) \rangle_{(I_1,I_2)} \geq \mathbf{u} \right\}.$$

We now discuss some intuitions of the definition of Φ_Π^{hex}. Given a 3-valued interpretation $(I_1, I_2) \in \mathcal{I}_\Pi^c$, $\Phi_\Pi^{hex}(I_1, I_2)_1$ contains every atom $a \in \Lambda_\Pi$ which is derivable from Π under the assumption that every atom in I_1 is *true* and every atom not in I_2 is *false*. That is, given a rule $r \in \Pi$ with $H(r) = a$, we have $a \in \Phi_\Pi^{hex}(I_1, I_2)_1$ if (i) every atom $b \in B^+(r)$ is contained in I_1 (i.e., b is assumed to be *true*), (ii) for every external atom $f^{\#}[\mathbf{i}](\mathbf{o}) \in B^+(r)$ and every $J \in [I_1, I_2]$, $f(J, \mathbf{i}, \mathbf{o}) = \mathbf{t}$ (i.e., $f^{\#}[\mathbf{i}](\mathbf{o})$ evaluates to *true* in every possible interpretation

greater than I_1 consisting only of atoms which are not assumed to be *false*), (iii) every atom $b \in B^-(r)$ is not contained in I_2 (i.e., b is assumed to be *false*), and (iv) for every external atom $g^{\#}[\mathbf{i}](\mathbf{o}) \in B^-(r)$ and every $J \in [I_1, I_2]$, $g(J, \mathbf{i}, \mathbf{o}) = \mathbf{f}$ (i.e., $g^{\#}[\mathbf{i}](\mathbf{o})$ evaluates to *false* in every possible interpretation greater than I_1 consisting only of atoms which are not assumed to be *false*).

On the other hand, $\Phi^{hex}_{\Pi}(I_1, I_2)_2$ contains every atom $a \in \Lambda_{\Pi}$ such that there exists a rule $r \in \Pi$ with $H(r) = a$ such that (i) every atom $b \in B^+(r)$ is contained in I_2 (i.e., b is either *true* or *undefined*), (ii) for every external atom $f^{\#}[\mathbf{i}](\mathbf{o}) \in B^+(r)$ there exists some $J \in [I_1, I_2]$ such that $f(J, \mathbf{i}, \mathbf{o}) = \mathbf{t}$, (iii) every atom $b \in B^-(r)$ is not contained in I_1 (i.e., b is either *false* or *undefined*), and (iv) for every external atom $g^{\#}[\mathbf{i}](\mathbf{o}) \in B^-(r)$ there exists some $J \in [I_1, I_2]$ such that $f(J, \mathbf{i}, \mathbf{o}) = \mathbf{f}$.

The next result states that Φ^{hex}_{Π} is an extension of the traditional Fitting operator Φ_{Π} as defined in (Fitting, 1985) (see Example 3.2.5).

Proposition 4.1.9. *Let Π be a classical normal program. Then, $\Phi^{hex}_{\Pi} = \Phi_{\Pi}$.*

Recall from Section 3.2 that Φ^{hex}_{Π} is an *approximation* of T^{hex}_{Π} if

1. $\Phi^{hex}_{\Pi}(I, I)_1 = T^{hex}_{\Pi}(I) = \Phi^{hex}_{\Pi}(I, I)_2$ for each $I \in \mathcal{I}_{\Pi}$, and
2. Φ^{hex}_{Π} is monotone with respect to $\subseteq_p$.

In Example 3.2.5 we mentioned that for every *classical* normal program Π, Φ_{Π} is an approximation of T_{Π} (Denecker et al., 2000a). The next result shows that this also holds in the case of normal HEX programs. Before proving this assertion, we show the following lemma.

Lemma 4.1.10. *Let Π be a HEX program, and let $(J_1, J_2), (I_1, I_2) \in \mathcal{I}^c_{\Pi}$. For every set of HEX-literals L, if $(J_1, J_2) \subseteq_p (I_1, I_2)$, then*

1. *if $\langle L \rangle_{(J_1,J_2)} = \mathbf{t}$, then $\langle L \rangle_{(I_1,I_2)} = \mathbf{t}$,*
2. *if $\langle L \rangle_{(I_1,I_2)} \geq \mathbf{u}$, then $\langle L \rangle_{(J_1,J_2)} \geq \mathbf{u}$.*

Proof. The first implication is straightforward. For the second implication notice that for an external atom $f^{\#}[\mathbf{i}](\mathbf{o}) \in \Lambda^{\#}_{\Pi}$ we have $\langle f^{\#}[\mathbf{i}](\mathbf{o}) \rangle_{(I_1,I_2)} = \mathbf{u}$ if there exists some $I', I'' \in [I_1, I_2]$ such that $f(I', \mathbf{i}, \mathbf{o}) = \mathbf{t}$ and $f(I'', \mathbf{i}, \mathbf{o}) = \mathbf{f}$. Then, since $J_1 \subseteq I_1$ and $I_2 \subseteq J_2$, we have $\langle f^{\#}[\mathbf{i}](\mathbf{o}) \rangle_{(J_1,J_2)} = \mathbf{u}$. Moreover, if we have $\langle f^{\#}[\mathbf{i}](\mathbf{o}) \rangle_{(I_1,I_2)} = \mathbf{t}$, then there exists at least one interpretation $I' \in [I_1, I_2]$ such that $f(I', \mathbf{i}, \mathbf{o}) = \mathbf{t}$. Since $I' \in [J_1, J_2]$, we thus have $\langle f^{\#}[\mathbf{i}](\mathbf{o}) \rangle_{(J_1,J_2)} \geq \mathbf{u}$. The rest of the proof is straightforward. □

Proposition 4.1.11. *Let Π be a normal HEX program. Then, Φ^{hex}_{Π} is an approximation of T^{hex}_{Π}.*

Proof. (1) Follows directly from (4.3), and (4.5). (2) Is an immediate consequence of Lemma 4.1.10. □

Proposition 4.1.11 shows that we can apply the Approximation Theory (Denecker et al., 2000a, 2002, 2004) presented in Chapter 3 to T^{hex}_{Π} and its approximation Φ^{hex}_{Π} (see Section 3.2). Since Φ^{hex}_{Π} is monotone (with respect to $\subseteq_p$), it has a least fixpoint on the chain-complete poset $(\mathcal{I}^c_{\Pi}, \subseteq_p)$. This leads to the following definition.

Definition 4.1.12 (Kripke-Kleene semantics)**.** Let Π be a normal HEX program. Define the Φ_Π^{hex}*-Kripke-Kleene model* of Π by

$$k(\Phi_\Pi^{hex}) = \mathrm{lfp}(\Phi_\Pi^{hex}).$$

The fixpoint $k\left(\Phi_\Pi^{hex}\right)$ is the limit of the transfinite sequence

$$(\emptyset, \Lambda_\Pi) \subseteq_p \Phi_\Pi^{hex}(\emptyset, \Lambda_\Pi) \subseteq_p \ldots \subseteq_p \Phi_\Pi^{hex,\alpha}(\emptyset, \Lambda_\Pi) \subseteq_p \ldots.$$

For a classical normal program Π, the Φ_Π-Kripke-Kleene model $k(\Phi_\Pi)$ approximates every supported model of Π (see Example 3.2.8). We have not explicitly defined supported model semantics for normal HEX programs. However, Proposition 4.1.3 shows that fixpoints of T_Π^{hex} are in fact models of Π and it is easy to see that these models are "supported" (in the classical sense). The following result is an immediate consequence of Proposition 3.2.7 and shows that the Φ_Π^{hex}-Kripke-Kleene model $k\left(\Phi_\Pi^{hex}\right)$ approximates each fixpoint of T_Π^{hex}.

Proposition 4.1.13. *Let Π be a normal HEX program, and let $I \in \mathcal{I}_\Pi$. If I is a fixpoint of T_Π^{hex}, then $k(\Phi_\Pi^{hex}) \subseteq_p I$.*

Corollary 4.1.14. *Let Π be a HEX program, and let $I \in \mathcal{I}_\Pi$. If I is an FLP-answer set of Π, then $k\left(\Phi_\Pi^{hex}\right) \subseteq_p I$.*

Proof. From Proposition 2.2.4 and Proposition 4.1.3 it follows that every FLP-answer set I of Π is a fixpoint of T_Π^{hex}. Hence, by Proposition 4.1.13, $k\left(\Phi_\Pi^{hex}\right) \subseteq_p I$. □

We now illustrate the Φ_Π^{hex}-Kripke-Kleene semantics by giving some examples.

Example 4.1.15. Let Π be the normal HEX program consisting of the following rules:

$$q(a) \leftarrow$$
$$p(a) \leftarrow p \subseteq^{\#} q, q(a)$$

where $I \models p \subseteq^{\#} q$ iff $p^I \subseteq q^I$ (see Example 2.2.2), and $\Lambda_\Pi = \{p(a), q(a)\}$. We compute the Φ_Π^{hex}-Kripke-Kleene model of Π as follows. In the first iteration, we have $q(a) \in \Phi_\Pi^{hex}(\emptyset, \Lambda_\Pi)_1$, and

$$p(a) \notin \Phi_\Pi^{hex}(\emptyset, \Lambda_\Pi)_1 \quad \text{and} \quad p(a) \in \Phi_\Pi^{hex}(\emptyset, \Lambda_\Pi)_2.$$

That is, the first iteration yields $\Phi_\Pi^{hex}(\emptyset, \Lambda_\Pi) = (\{q(a)\}, \Lambda_\Pi)$. For the second iteration, observe $\left\langle p \subseteq^{\#} q \right\rangle_{(\{q(a)\}, \Lambda_\Pi)} = \mathbf{t}$ which entails $p(a) \in \Phi_\Pi^{hex}(\{q(a)\}, \Lambda_\Pi)_1$. Consequently, we have $\Phi_\Pi^{hex}(\{q(a)\}, \Lambda_\Pi) = (\Lambda_\Pi, \Lambda_\Pi)$ and, hence, $k\left(\Phi_\Pi^{hex}\right) = (\Lambda_\Pi, \Lambda_\Pi)$. This result coincides with the intended meaning of Π.

Example 4.1.16. We modify the program of Example 4.1.15 to the program Π' consisting of the following rules:

$$q(a) \leftarrow$$
$$p(b) \leftarrow p \subseteq^{\#} q, q(a).$$

The second rule intuitively states that whenever p^I is a subset of q^I, p^I contains an element (namely b) not contained in q^I. Hence, Π' is inconsistent, i.e., it has no FLP-answer set. However, under Φ^{hex}_{Π}-Kripke-Kleene semantics we can still entail that $q(a)$ is *true*. That is, in the first iteration we again obtain $\Phi^{hex}_{\Pi'}(\emptyset, \Lambda_\Pi) = (\{q(a)\}, \{p(b), q(a)\})$. However, in the second iteration we now have $\left\langle\, p \subseteq^{\#} q \,\right\rangle_{(\{q(a)\},\{p(b),q(a)\})} = \mathbf{u}$ and, thus, we cannot entail $p(b)$. Consequently, $k(\Phi^{hex}_{\Pi'}) = (\{q(a)\}, \{p(b), q(a)\})$ which means that $q(a)$ is *true* and $p(b)$ is *undefined* in $k(\Phi^{hex}_{\Pi'})$.

Example 4.1.17. Reconsider the normal HEX program Π of Example 4.1.5 consisting of the following rules:

$$a \leftarrow b^{\#}$$
$$b \leftarrow a^{\#}.$$

Intuitively, a and b are indirectly self-supported, i.e., a is supported by b and vice versa. Since $I_1 = \emptyset$ and $I_2 = \{a, b\}$ are the fixpoints of T^{hex}_{Π}, and since $k\left(\Phi^{hex}_{\Pi}\right)$ approximates every fixpoint of T^{hex}_{Π}, we expect $k\left(\Phi^{hex}_{\Pi}\right) = (\emptyset, \{a, b\})$. Indeed, since $\left\langle\, a^{\#} \,\right\rangle_{(\emptyset,\{a,b\})} = \left\langle\, b^{\#} \,\right\rangle_{\emptyset,(\{a,b\})} = \mathbf{u}$, we have $\Phi^{hex}_{\Pi}(\emptyset, \{a, b\}) = (\emptyset, \{a, b\})$ and, hence, $k\left(\Phi^{hex}_{\Pi}\right) = (\emptyset, \{a, b\})$. That is, in the 3-valued Φ^{hex}_{Π}-Kripke-Kleene model of Π both atoms are *undefined*.

4.2 Three-Valued Answer-Set Semantics

Let us briefly recall the combination of Approximation Theory and classical logic programming for characterizing 3-valued answer-set semantics of classical normal programs (see Section 3.3). Given a classical normal program Π, the 3-valued answer sets of Π are characterized by the fixpoints of the stable revision operator $\Phi^{\downarrow\uparrow}_{\Pi}$ of the Fitting approximation Φ_Π. Moreover, 2-valued answer sets of Π are fixpoints of $\Phi^{\downarrow}_{\Pi}$, i.e., $I \in \mathcal{I}_\Pi$ is an answer set of Π iff I is a fixpoint of T_Π and $\Phi^{\downarrow}_{\Pi}(I) = I$ (see Example 3.3.7).

Likewise, in this section we extend these definitions to the class of normal HEX programs. More precisely, we define, for a given normal HEX program Π, the stable revision operator $\Phi^{hex,\downarrow\uparrow}_{\Pi}$ of Φ^{hex}_{Π} in a straightforward way (see Section 3.3), and obtain 3-, and 2-valued Φ^{hex}_{Π}-answer-set semantics.

In the sequel let Π be a normal HEX program. We say that $(I_1, I_2) \in \mathcal{I}^c_\Pi$ is Φ^{hex}_{Π}*-reliable* if $(I_1, I_2) \subseteq_p \Phi^{hex}_{\Pi}(I_1, I_2)$. That is, (I_1, I_2) is Φ^{hex}_{Π}-reliable if $\Phi^{hex}_{\Pi}(I_1, I_2)$ is a refinement of (I_1, I_2), i.e., $\Phi^{hex}_{\Pi}(I_1, I_2)$ contains more *true* and less *undefined* atoms than (I_1, I_2). Given a Φ^{hex}_{Π}-reliable 3-valued interpretation $(I_1, I_2) \in \mathcal{I}^c_\Pi$, the restriction of $\Phi^{hex}_{\Pi}(\,.\,, I_2)_1$ (resp., $\Phi^{hex}_{\Pi}(I_1, \,.\,)_2$) to $[\emptyset, I_2]$ (resp., $[I_1, \Lambda_\Pi]$) is a monotone operator on the complete lattice $([\emptyset, I_2], \subseteq)$ (resp., $([I_1, \Lambda_\Pi], \subseteq)$). Hence, by the Tarski and Knaster Theorem, $\Phi^{hex}_{\Pi}(\,.\,, I_2)_1$ (resp., $\Phi^{hex}_{\Pi}(I_1, \,.\,)_2$) has a least fixpoint in $([\emptyset, I_2], \subseteq)$ (resp., $([I_1, \Lambda_\Pi], \subseteq)$) (see Section 3.3).

Definition 4.2.1 (Stable revision operator)**.** Let Π be a normal HEX program. Define the *stable revision operator* of Φ^{hex}_{Π}, for each $(I_1, I_2) \in \mathcal{I}^c_\Pi$, by

$$\Phi^{hex,\downarrow\uparrow}_{\Pi}(I_1, I_2) = \left(\Phi^{hex,\downarrow}_{\Pi}(I_2), \Phi^{hex,\uparrow}_{\Pi}(I_1)\right),$$

where

$$\Phi_{\Pi}^{hex,\downarrow}(I_2) = \mathrm{lfp}\left(\Phi_{\Pi}^{hex}(\,.\,, I_2)_1\right) \quad \text{and} \quad \Phi_{\Pi}^{hex,\uparrow}(I_1) = \mathrm{lfp}\left(\Phi_{\Pi}^{hex}(I_1, \,.\,)_2\right).$$

We now discuss the intuitions behind these definitions (see Denecker et al. (2004)). The operator $\Phi_{\Pi}^{hex,\downarrow}$ is designed in such a way that $\Phi_{\Pi}^{hex,\downarrow}(I_2)$ underestimates every fixpoint $I \subseteq I_2$ of T_{Π}^{hex} as tight as possible. This lower bound is the limit of the sequence (notice that $\Phi_{\Pi}^{hex}(\,.\,, I_2)_1$ is monotone, see Figure 3.3)

$$\emptyset \subseteq_p \Phi_{\Pi}^{hex}(\emptyset, I_2)_1 \subseteq \ldots \subseteq \Phi_{\Pi}^{hex,\alpha}(\emptyset, I_2)_1 \subseteq \ldots$$

where α is an ordinal. This is summarized in the following proposition (see Proposition 3.3.1).

Proposition 4.2.2. *Let Π be a normal HEX program, and let $(I_1, I_2) \in \mathcal{I}_{\Pi}^c$ be Φ_{Π}^{hex}-reliable. Then, for every (pre-)fixpoint $I \in \mathcal{I}_{\Pi}$ of T_{Π}^{hex}, if $I \subseteq I_2$ then $\Phi_{\Pi}^{hex,\downarrow}(I_2) \subseteq I$.*

Since we are interested in minimal fixpoints of T_{Π}^{hex}, the new upper bound $\Phi_{\Pi}^{hex,\uparrow}(I_1)$ is an interpretation $I_2' \subseteq I_2$ that is closer to the minimal fixpoints of T_{Π}^{hex} than I_2. However, recall from Section 3.3 that $\Phi_{\Pi}^{hex,\downarrow\uparrow}(I_1, I_2)$ has not to be a refinement of (I_1, I_2). We say that (I_1, I_2) is Φ_{Π}^{hex}*-prudent* if $I_1 \subseteq \Phi_{\Pi}^{hex,\downarrow}(I_2)$. Intuitively, a 3-valued interpretation $(I_1, I_2) \in \mathcal{I}_{\Pi}^c$ is Φ_{Π}^{hex}-prudent, if all atoms in I_1 are logical consequences of Π if all atoms not in I_2 are *false*.

Example 4.2.3. Reconsider the normal HEX program Π of Example 4.1.17 consisting of the following rules:

$$a \leftarrow\sim b^{\#}$$
$$b \leftarrow\sim a^{\#}.$$

Since $\Phi_{\Pi}^{hex}(\emptyset, \{a, b\})_1 = \emptyset$, we have $\Phi_{\Pi}^{hex,\downarrow}(\{a, b\}) = \emptyset$. Hence, e.g., $(\{a\}, \{a, b\})$ and $(\{b\}, \{a, b\})$ are not Φ_{Π}^{hex}-prudent, whereas $(\emptyset, \{a, b\})$ is (trivially) Φ_{Π}^{hex}-prudent.

By translating Proposition 3.3.2 and Proposition 3.3.3 we immediately conclude the following two results.

Proposition 4.2.4. *Let Π be a normal HEX program, and let $(I_1, I_2) \in \mathcal{I}_{\Pi}^c$ be Φ_{Π}^{hex}-prudent. Then, $\Phi_{\Pi}^{hex,\downarrow\uparrow}(I_1, I_2)$ is Φ_{Π}^{hex}-prudent and $(I_1, I_2) \subseteq_p \Phi_{\Pi}^{hex,\downarrow\uparrow}(I_1, I_2)$.*

Proposition 4.2.5. *Let Π be a normal HEX program, and let $(I_1, I_2) \in \mathcal{I}_{\Pi}^c$. If (I_1, I_2) is Φ_{Π}^{hex}-prudent, then $\Phi_{\Pi}^{hex}(I_1, I_2) \subseteq_p \Phi_{\Pi}^{hex,\downarrow\uparrow}(I_1, I_2)$.*

The stable revision operator $\Phi_{\Pi}^{hex,\downarrow\uparrow}$ has fixpoints and a least fixpoint on the chain-complete poset $(\mathcal{I}_{\Pi}^c, \subseteq_p)$. This leads to the following definition.

Definition 4.2.6 (Answer-set semantics)**.** Let Π be a normal HEX program, and let $(I_1, I_2) \in \mathcal{I}_{\Pi}^c$. We say that $(I_1, I_2) \in \mathcal{I}_{\Pi}^c$ is a *(3-valued)* Φ_{Π}^{hex}*-answer set* of Π if (I_1, I_2) is a fixpoint of $\Phi_{\Pi}^{hex,\downarrow\uparrow}$, i.e., $\Phi_{\Pi}^{hex,\downarrow\uparrow}(I_1, I_2) = (I_1, I_2)$.

As a consequence of Proposition 3.3.6, the next result states that we can characterize *2-valued* (or, equivalently, *exact*) Φ_Π^{hex}-answer sets by fixpoints of $\Phi_\Pi^{hex,\downarrow}$ (see Figure 3.3).

Proposition 4.2.7. *Let Π be a normal HEX program, and let $I \in \mathcal{I}_\Pi$. Then, I is a 2-valued Φ_Π^{hex}-answer set iff I is a fixpoint of T_Π^{hex} and $\Phi_\Pi^{hex,\downarrow}(I) = I$.*

Example 4.2.8. Reconsider the normal HEX program Π of Example 4.2.3 consisting of the following rules:

$$a \leftarrow\sim b^{\#}$$
$$b \leftarrow\sim a^{\#}.$$

It is easy to verify that $k\left(\Phi_\Pi^{hex}\right) = (\emptyset, \{a, b\})$. Hence, since $\Phi_\Pi^{hex}(\emptyset, \{a, b\})_1 = \emptyset$, we have $\Phi_\Pi^{hex,\downarrow}(\{a, b\}) = \emptyset$. Moreover, we have $\Phi_\Pi^{hex}(\emptyset, \emptyset)_2 = \{a, b\}$ and, consequently, $\Phi_\Pi^{hex,\downarrow\uparrow}(\emptyset, \{a, b\}) = (\emptyset, \{a, b\})$. That is, $(\emptyset, \{a, b\})$ is a 3-valued Φ_Π^{hex}-answer set.

We now compute $\Phi_\Pi^{hex,\downarrow}(\{a\})$. In $(\emptyset, \{a\}) \in \mathcal{I}_\Pi^c$ the atom b is known to be *false*, whereas a is *undefined*. Under the assumption that b is *false*, we can conclude with the first rule that a is *true*. That is, we have $\Phi_\Pi^{hex}(\emptyset, \{a\})_1 = \{a\}$. For the second iteration, we compute $\Phi_\Pi^{hex}(\{a\}, \{a\})_1 = \{a\}$ and conclude $\Phi_\Pi^{hex,\downarrow}(\{a\}) = \{a\}$. That is, $\{a\}$ is a 2-valued Φ_Π^{hex}-answer set. Also, an analogous computation shows that $\{b\}$ is also a 2-valued Φ_Π^{hex}-answer set.

Example 4.2.9. Let Π be the normal HEX program consisting of the following rules:

$$p(a) \leftarrow\sim \exists^{\#}[\, q\,]$$
$$q(b) \leftarrow \exists^{\#}[\, q\,]$$

where $\Sigma_\Pi^{\#} = \{\exists^{\#}\}$ consists of an $(1, 0)$-ary external symbol $\exists^{\#}$, $\Sigma_\Pi^{(0)} = \{a, b\}$ are the constants, $\Sigma_\Pi^{(1)} = \{p, q\}$ are the 1-ary (predicate) symbols, and $\Lambda_\Pi = \{p(a), q(b)\}$ are the atoms occurring in Π, respectively. The intended meaning of $\exists^{\#}[\, q\,]$ is that it evaluates to *true* in an interpretation I iff there exists an atom $q(a)$ or $q(b)$ in I. More formally, we define the interpretation function $\exists : \mathcal{I}_\Pi \times \Sigma_\Pi \longrightarrow 2$ by

$$\exists(I, q') = \begin{cases} \mathbf{t} & \text{if there exists some constant } c \in \Sigma_\Pi^{(0)} \text{ such that } q'(c) \in I, \\ \mathbf{f} & \text{otherwise.} \end{cases}$$

For convenience, in the sequel we write $(\exists x\ q(x))^{\#}$ instaed of $\exists^{\#}[\, q\,]$ and thus rewrite Π as follows:

$$p(a) \leftarrow\sim (\exists x\ q(x))^{\#}$$
$$q(b) \leftarrow (\exists x\ q(x))^{\#}.$$

First, since $\{p(a)\} \not\models (\exists x\ q(x))^{\#}$, we have $T_\Pi^{hex}(\{p(a)\}) = \{p(a)\}$. Second, since $\{q(b)\} \models (\exists x\ q(x))^{\#}$, we have $T_\Pi^{hex}(\{q(b)\}) = \{q(b)\}$. That is, $\{p(a)\}$ and $\{q(b)\}$ are both fixpoints of T_Π^{hex}. Consequently, since $k\left(\Phi_\Pi^{hex}\right)$ approximates every fixpoint of T_Π^{hex} (see Proposition

4.1.13), we have $k\left(\Phi_\Pi^{hex}\right) = (\emptyset, \{p(a), q(b)\})$. Hence, $p(a)$ and $q(b)$ are both *undefined* under Φ_Π^{hex}-Kripke-Kleene semantics.

We show that under Φ_Π^{hex}-answer-set semantics we obtain more precise results. In particular, we show that $k\left(\Phi_\Pi^{hex}\right) = (\emptyset, \{p(a), q(b)\})$ is not a fixpoint of the stable revision operator $\Phi_\Pi^{hex,\downarrow\uparrow}$. We compute $\Phi_\Pi^{hex,\uparrow}(\emptyset)$ as follows:

$$\Phi_\Pi^{hex}(\emptyset, \emptyset)_2 = \{p(a)\}$$
$$\Phi_\Pi^{hex}(\emptyset, \{p(a)\})_2 = \{p(a)\}.$$

That is, we have $\Phi_\Pi^{hex,\uparrow}(\emptyset) = \{p(a)\} \neq \{p(a), q(b)\}$ which proves $\Phi_\Pi^{hex,\downarrow\uparrow}(\emptyset, \{p(a), q(b)\}) \neq (\emptyset, \{p(a), q(b)\})$. Moreover, since $\emptyset \not\models (\exists x\ q(x))^{\#}$, and $\{p(a)\} \not\models (\exists x\ q(x))^{\#}$, we have

$$\Phi_\Pi^{hex}(\emptyset, \{p(a)\})_1 = \Phi_\Pi^{hex}(\{p(a)\}, \{p(a)\})_1 = \{p(a)\},$$

which proves that $\{p(a)\}$ is an Φ_Π^{hex}-answer set. Clearly, $\{p(a)\}$ is a more precise than the Φ_Π^{hex}-Kripke-Kleene model $(\emptyset, \{p(a), q(b)\})$.

Notice that $q(b)$ is self-supported in Π and therefore $\{q(b)\}$ is not an Φ_Π^{hex}-answer set. Indeed, we have $\Phi_\Pi^{hex,\downarrow}(\{q(b)\}) = \emptyset$.

4.3 Well-Founded Semantics

In Example 3.3.10 we briefly presented well-founded semantics of classical normal programs in a traditional fashion using unfounded sets. We noticed that, given a classical normal program Π, we can characterize the well-founded model $w(\Pi)$ by the Φ_Π-well-founded fixpoint $w(\Phi_\Pi)$, i.e., $w(\Pi) = w(\Phi_\Pi)$.

Well-founded semantics play an important role in classical 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 Φ_Π^{hex}-answer-set semantics by instantiating the constructions of Approximation Theory given in Section 3.3.

Let Π be a normal HEX program. Recall from Section 4.2 that the stable revision operator $\Phi_\Pi^{hex,\downarrow\uparrow}$ of Φ_Π^{hex} has fixpoints and a least fixpoint in the chain-complete poset $(\mathcal{I}_\Pi^c, \subseteq_p)$ (Denecker et al., 2004). This leads to the following definition.

Definition 4.3.1 (Well-founded semantics)**.** Let Π be a normal HEX program. Define the Φ_Π^{hex}-*well-founded model* by

$$w\left(\Phi_\Pi^{hex}\right) = \mathrm{lfp}\left(\Phi_\Pi^{hex,\downarrow\uparrow}\right).$$

Since Φ_Π^{hex} is an extension of Φ_Π (see Proposition 4.1.9), we immediately conclude the following result.

Proposition 4.3.2. *Let Π be a classical normal program. Then, $w(\Pi) = w(\Phi_\Pi) = w(\Phi_\Pi^{hex})$.*

We can compute $w\left(\Phi_\Pi^{hex}\right)$ by iterating $\Phi_\Pi^{hex,\downarrow\uparrow}$, starting at $(\emptyset, \Lambda_\Pi)$, until a fixpoint is reached. More precisely, $w\left(\Phi_\Pi^{hex}\right)$ is the limit of the sequence $((I_1^\alpha, I_2^\alpha))_{\alpha \geq 0}$ of elements of $\mathcal{I}_\Pi^c$, where[1]

[1]Notice that $\vee$ denotes the supremum in the chain-complete poset $(\mathcal{I}_\Pi^c, \subseteq_p)$.

1. $\left(I_1^0, I_2^0\right) = (\emptyset, \Lambda_\Pi)$
2. $I_1^{\alpha+1} = \Phi_\Pi^{hex,\downarrow}\left(I_2^\alpha\right)$ and $I_2^{\alpha+1} = \Phi_\Pi^{hex,\uparrow}\left(I_1^\alpha\right)$
3. $(I_1^\alpha, I_2^\alpha) = \bigvee \left\{ \left(I_1^\beta, I_2^\beta\right) : \beta < \alpha \right\}$ for every limit ordinal α.

That is, the Φ_Π^{hex}-well-founded model is the least 3-valued Φ_Π^{hex}-answer set and approximates every 3-valued Φ_Π^{hex}-answer set, i.e., $w\left(\Phi_\Pi^{hex}\right) \subseteq_p (I_1, I_2)$ for every Φ_Π^{hex}-answer set $(I_1, I_2) \in \mathcal{I}_\Pi^c$. In particular, $w\left(\Phi_\Pi^{hex}\right)$ approximates every 2-valued Φ_Π^{hex}-answer set.

As a consequence of Proposition 3.3.9, the next result states that the Φ_Π^{hex}-well founded model is, in general, more precise than the Φ_Π^{hex}-Kripke-Kleene model.

Proposition 4.3.3. *Let Π be a normal HEX program. Then, $k\left(\Phi_\Pi^{hex}\right) \subseteq_p w\left(\Phi_\Pi^{hex}\right)$.*

Example 4.3.4. Reconsider the inconsistent normal HEX program Π' of Example 4.1.16 consisting of the following rules:

$$q(a) \leftarrow$$
$$p(b) \leftarrow p \subseteq^{\#} q, q(a).$$

We have seen in Example 4.1.16 that under $\Phi_{\Pi'}^{hex}$-Kripke-Kleene semantics, $q(a)$ is *true* and $p(b)$ is *undefined*, i.e., $k(\Phi_{\Pi'}^{hex}) = (\{q(a)\}, \{p(b), q(a)\})$. We show that the $\Phi_{\Pi'}^{hex}$-well-founded model $w(\Phi_{\Pi'}^{hex})$ coincides with the $\Phi_{\Pi'}^{hex}$-Kripke-Kleene model.

First, we compute $\Phi_{\Pi'}^{hex,\downarrow}(\{p(b), q(a)\})$. Since

$$\Phi_{\Pi'}^{hex}(\emptyset, \{p(b), q(a)\})_1 = \Phi_{\Pi'}^{hex}(\{q(a)\}, \{p(b), q(a)\})_1 = \{q(a)\},$$

we have

$$\Phi_{\Pi'}^{hex,\downarrow}(\{p(b), q(a)\}) = \{q(a)\}.$$

Second, the computation

$$\begin{aligned}\Phi_{\Pi'}^{hex}(\emptyset, \emptyset)_2 &= \{q(a)\}\\ \Phi_{\Pi'}^{hex}(\emptyset, \{q(a)\})_2 &= \{p(b), q(a)\}\\ \Phi_{\Pi'}^{hex}(\emptyset, \{p(b), q(a)\})_2 &= \{p(b), q(a)\}\end{aligned}$$

shows $\Phi_{\Pi'}^{hex,\uparrow}(\emptyset) = \{p(b), q(a)\}$. Moreover,

$$\Phi_{\Pi'}^{hex}(\{q(a)\}, \{q(a)\})_2 = \Phi_{\Pi'}^{hex}(\{q(a)\}, \{p(b), q(a)\})_2 = \{p(b), q(a)\}$$

shows $\Phi_{\Pi'}^{hex,\uparrow}(\{q(a)\}) = \{p(b), q(a)\}$. Hence, we have

$$w(\Phi_{\Pi'}^{hex}) = (\{q(a)\}, \{p(b), q(a)\}) = k(\Phi_{\Pi'}^{hex}).$$

Example 4.3.5. Reconsider the normal HEX program Π of Example 4.1.17 consisting of the following rules:

$$a \leftarrow b^{\#}$$
$$b \leftarrow a^{\#}.$$

We noticed in Example 4.1.17 that a and b are both self-supported. However, under Φ_{Π}^{hex}-Kripke-Kleene semantics both atoms are *undefined* and not *false*. We show that Φ_{Π}^{hex}-well-founded semantics yields a stronger result and correctly detects the self-supportedness. We compute $w\left(\Phi_{\Pi}^{hex}\right)$ by iterating $\Phi_{\Pi}^{hex,\downarrow\uparrow}$ over $(\emptyset, \{a, b\})$ as follows:

$$\Phi_{\Pi}^{hex,\downarrow\uparrow}(\emptyset, \{a, b\}) = \left(\Phi_{\Pi}^{hex,\downarrow}(\{a, b\}), \Phi_{\Pi}^{hex,\uparrow}(\emptyset)\right) = (\emptyset, \emptyset)$$
$$\Phi_{\Pi}^{hex,\downarrow\uparrow}(\emptyset, \emptyset) = \left(\Phi_{\Pi}^{hex,\downarrow}(\emptyset), \Phi_{\Pi}^{hex,\uparrow}(\emptyset)\right) = (\emptyset, \emptyset).$$

That is, $w\left(\Phi_{\Pi}^{hex}\right) = (\emptyset, \emptyset)$ and, consequently, a and b are both *false* in $w\left(\Phi_{\Pi}^{hex}\right)$.

Example 4.3.6. Reconsider the normal HEX program Π of Example 4.2.9 consisting of the following rules:

$$p(a) \leftarrow\sim (\exists x\ q(x))^{\#}$$
$$q(b) \leftarrow (\exists x\ q(x))^{\#}.$$

In Example 4.2.9 we have seen that under Φ_{Π}^{hex}-Kripke-Kleene semantics both atoms (i.e., $p(a)$ and $q(b)$) are *undefined*, i.e., $k\left(\Phi_{\Pi}^{hex}\right) = (\emptyset, \{p(a), q(b)\})$. We show that $w\left(\Phi_{\Pi}^{hex}\right)$ is more precise than $k\left(\Phi_{\Pi}^{hex}\right)$, that is, under Φ_{Π}^{hex}-well-founded semantics we can extract more information from Π. We compute $w\left(\Phi_{\Pi}^{hex}\right)$ by iterating $\Phi_{\Pi}^{hex,\downarrow\uparrow}$ over $(\emptyset, \{p(a), q(b)\})$ as follows:

$$\Phi_{\Pi}^{hex,\downarrow\uparrow}(\emptyset, \{p(a), q(b)\}) = \left(\Phi_{\Pi}^{hex,\downarrow}(\{p(a), q(b)\}), \Phi_{\Pi}^{hex,\uparrow}(\emptyset)\right) = (\emptyset, \{p(a)\})$$
$$\Phi_{\Pi}^{hex,\downarrow\uparrow}(\emptyset, \{p(a)\}) = \left(\Phi_{\Pi}^{hex,\downarrow}(\{p(a)\}), \Phi_{\Pi}^{hex,\uparrow}(\emptyset)\right) = (\{p(a)\}, \{p(a)\}).$$

That is, $w\left(\Phi_{\Pi}^{hex}\right) = (\{p(a)\}, \{p(a)\})$ and, consequently, $k\left(\Phi_{\Pi}^{hex}\right) \subsetneq_p w\left(\Phi_{\Pi}^{hex}\right)$.

4.4 Ultimate Semantics

Let Π be a normal HEX program. In Section 4.1 we have seen that the extended van Emden-Kowalski operator T_{Π}^{hex} shares some basic properties with the traditional van Emden-Kowalski operator T_{Π} defined for classical normal programs. That is, (i) T_{Π}^{hex} is a lattice operator, (ii) pre-fixpoints of T_{Π}^{hex} characterize models of Π (see Proposition 4.1.3), and (iii) FLP-answer sets of Π are minimal fixpoints of T_{Π}^{hex} (see Corollary 4.1.4).[2] Arguably, these results indicate that T_{Π}^{hex} is an appropriate operator for applying Approximation Theory. Hence, in what follows we

[2]Recall that FLP-answer sets are minimal models (see Proposition 2.2.4).

translate the definitions and results of Section 3.4 and define ultimate semantics of normal HEX programs.

Given two approximations $\mathcal{A}$ and $\mathcal{B}$ of T_Π^{hex}, we say that $\mathcal{A}$ is *more precise* than $\mathcal{B}$ if $\mathcal{A}(I_1, I_2) \subseteq_p \mathcal{B}(I_1, I_2)$ for each $(I_1, I_2) \in \mathcal{I}_\Pi^c$. The set of all approximations of T_Π^{hex} ordered by $\subseteq_p$ forms a complete lattice. We denote the most precise (i.e., the greatest element with respect to $\subseteq_p$) approximation of T_Π^{hex} by $\mathcal{T}_\Pi^{hex}$. Second, by instantiating Theorem 3.4.1, we immediately conclude the following algebraic characterization of $\mathcal{T}_\Pi^{hex}$ (Denecker et al., 2004).

Definition 4.4.1 (Ultimate approximation)**.** Let Π be a normal HEX program. Define the *ultimate approximation* of T_Π^{hex}, for each $(I_1, I_2) \in \mathcal{I}_\Pi^c$, by

$$\mathcal{T}_\Pi^{hex}(I_1, I_2) = \left(\bigcap_{I_1 \subseteq J \subseteq I_2} T_\Pi^{hex}(J), \bigcup_{I_1 \subseteq J \subseteq I_2} T_\Pi^{hex}(J) \right).$$

Then, according to the theory presented in Chapter 3, we obtain the following definitions.

Definition 4.4.2 (Ultimate semantics)**.** Let Π be a normal HEX program. Define (i) the *ultimate Kripke-Kleene model* of Π by $k(\mathcal{T}_\Pi^{hex})$ (see Section 4.1), (ii) the *ultimate 3-valued answer sets* of Π by the fixpoints of the stable revision operator $\mathcal{T}_\Pi^{hex,\downarrow\uparrow}$ (see Section 4.2), and (iii) the *ultimate well-founded model* of Π by $w(\mathcal{T}_\Pi^{hex})$ (see Section 4.3).

From Proposition 3.3.6 it follows that we can characterize 2-valued ultimate answer sets by fixpoints of $\mathcal{T}_\Pi^{hex,\downarrow}$. That is, I is a *2-valued ultimate answer set* of Π iff I is a fixpoint of T_Π^{hex} and $\mathcal{T}_\Pi^{hex,\downarrow}(I) = I$.

Example 4.4.3. Let Π be the normal HEX program consisting of the following rules (see Example 3.4.5):

$$a \leftarrow a^{\#}$$
$$a \leftarrow\sim a^{\#}.$$

When applying Φ_Π^{hex} to $(\emptyset, \{a\})$, we consider both rules at the same time and evaluate them in the 3-valued interpretation $(\emptyset, \{a\})$. Since in $(\emptyset, \{a\})$ the external atom $a^{\#}$ is neither definitely *true* nor *false*, we obtain $\Phi_\Pi^{hex}(\emptyset, \{a\})_1 = \emptyset$ and $\Phi_\Pi^{hex}(\emptyset, \{a\})_2 = \{a\}$. Hence, we cannot extract any information with Φ_Π^{hex} from $(\emptyset, \{a\})$. In contrast, if we apply $\mathcal{T}_\Pi^{hex}$ to $(\emptyset, \{a\})$ we consider each interpretation $I \in [\emptyset, \{a\}]$ and, consequently, each rule separately: for $I = \emptyset$ we have $I \not\models a^{\#}$ which means that the second rule fires, i.e., $T_\Pi^{hex}(I) = \{a\}$; and for $I = \{a\}$ we have $I \models a^{\#}$ which means that the first rule fires, i.e., again $T_\Pi^{hex}(I) = \{a\}$. Hence, $\mathcal{T}_\Pi^{hex}(\emptyset, \{a\})_1 = T_\Pi^{hex}(\emptyset) \cap T_\Pi^{hex}(\{a\}) = \{a\}$ and $\mathcal{T}_\Pi^{hex}(\emptyset, \{a\})_2 = T_\Pi^{hex}(\emptyset) \cup T_\Pi^{hex}(\{a\}) = \{a\}$, that is, $\mathcal{T}_\Pi^{hex}(\emptyset, \{a\}) = (\{a\}, \{a\})$. Since $T_\Pi^{hex}(\{a\}) = \{a\}$, we have $k\left(\mathcal{T}_\Pi^{hex}\right) = (\{a\}, \{a\})$ and, hence, $k\left(\Phi_\Pi^{hex}\right) \subsetneq_p k\left(\mathcal{T}_\Pi^{hex}\right)$.

The following relations are instantiations of Proposition 3.4.4, and precisely state the relationship between ultimate and Φ_Π^{hex}-semantics.

Proposition 4.4.4. *Let Π be a normal HEX program. Then, for every approximation $\mathcal{A}$ of T_Π^{hex}:*

1. $k(\mathcal{A}) \subseteq_p k(\mathcal{T}_\Pi^{hex})$.
2. $w(\mathcal{A}) \subseteq_p w(\mathcal{T}_\Pi^{hex})$.
3. *Every $\mathcal{A}$-answer set is an ultimate answer set of Π.*
4. *If $k(\mathcal{A})$ is 2-valued, then $k(\mathcal{A}) = k(\mathcal{T}_\Pi^{hex}) = w(\mathcal{T}_\Pi^{hex})$ and it is the unique ultimate answer set of Π.*
5. *If $w(\mathcal{A})$ is 2-valued, then $w(\mathcal{A}) = w(\mathcal{T}_\Pi^{hex})$ and it is the unique ultimate answer set of Π.*

Example 4.4.5. Reconsider the normal HEX program Π of Example 4.2.9 consisting of the following rules:

$$p(a) \leftarrow \sim (\exists x\ q(x))^{\#}$$
$$q(b) \leftarrow (\exists x\ q(x))^{\#}.$$

In Example 4.3.6 we have seen that $w\left(\Phi_\Pi^{hex}\right) = (\{p(a)\}, \{p(a)\})$, i.e., the Φ_Π^{hex}-well-founded model is 2-valued. Hence, by Proposition 4.4.4, $w\left(\Phi_\Pi^{hex}\right) = w\left(\mathcal{T}_\Pi^{hex}\right)$ and $\{p(a)\}$ is the unique ultimate answer set of Π.

As a direct consequence of Proposition 3.4.6, the next result states that if T_Π^{hex} is monotone, then computing $\mathcal{T}_\Pi^{hex}(I_1, I_2)$ reduces to computing T_Π^{hex} on the boundaries of $[I_1, I_2]$.

Proposition 4.4.6. *Let Π be a normal HEX program such that T_Π^{hex} is monotone. Then, for each $(I_1, I_2) \in \mathcal{I}_\Pi^c$, $\mathcal{T}_\Pi^{hex}(I_1, I_2) = \left(T_\Pi^{hex}(I_1), T_\Pi^{hex}(I_2)\right)$.*

Example 4.4.7. Consider the external atom $(\exists x\ q(x))^{\#}$ of Example 4.2.9 (see Example 4.4.5). Recall from Example 4.2.9 that $I \models (\exists x\ q(x))^{\#}$ iff there exists some constant $c \in \Sigma_\Pi^{(0)}$ such that $q(c) \in I$. Clearly, if $q(c) \in I$ then $q(c) \in I'$ for every $I \subseteq I'$, i.e., $I \models (\exists x\ q(x))^{\#}$ implies $I' \models (\exists x\ q(x))^{\#}$. Let Π be the normal HEX program consisting of the following rules:

$$q(a) \leftarrow$$
$$p(a) \leftarrow (\exists x\ q(x))^{\#}.$$

Notice that T_Π^{hex} is monotone and $k\left(\Phi_\Pi^{hex}\right) = (\Lambda_\Pi, \Lambda_\Pi)$ where $\Lambda_\Pi = \{p(a), q(a)\}$. By Proposition 4.4.4, we thus have $k\left(\mathcal{T}_\Pi^{hex}\right) = (\Lambda_\Pi, \Lambda_\Pi)$ which is verified by the following computation (see Proposition 4.4.6):

$$\mathcal{T}_\Pi^{hex}(\emptyset, \Lambda_\Pi) = \left(T_\Pi^{hex}(\emptyset), T_\Pi^{hex}(\Lambda_\Pi)\right) = (\{q(a)\}, \Lambda_\Pi)$$
$$\mathcal{T}_\Pi^{hex}(\{q(a)\}, \Lambda_\Pi) = \left(T_\Pi^{hex}(\{q(a)\}, T_\Pi^{hex}(\Lambda_\Pi))\right) = (\Lambda_\Pi, \Lambda_\Pi)$$
$$\mathcal{T}_\Pi^{hex}(\Lambda_\Pi, \Lambda_\Pi) = \left(T_\Pi^{hex}(\Lambda_\Pi), T_\Pi^{hex}(\Lambda_\Pi)\right) = (\Lambda_\Pi, \Lambda_\Pi).$$

Also, by Proposition 4.4.4, $w\left(\mathcal{T}_\Pi^{hex}\right) = (\Lambda_\Pi, \Lambda_\Pi)$ and Λ_Π is the unique ultimate answer set of Π.

In Figure 4.4 we illustrate the relationships between the different semantics defined in this chapter.

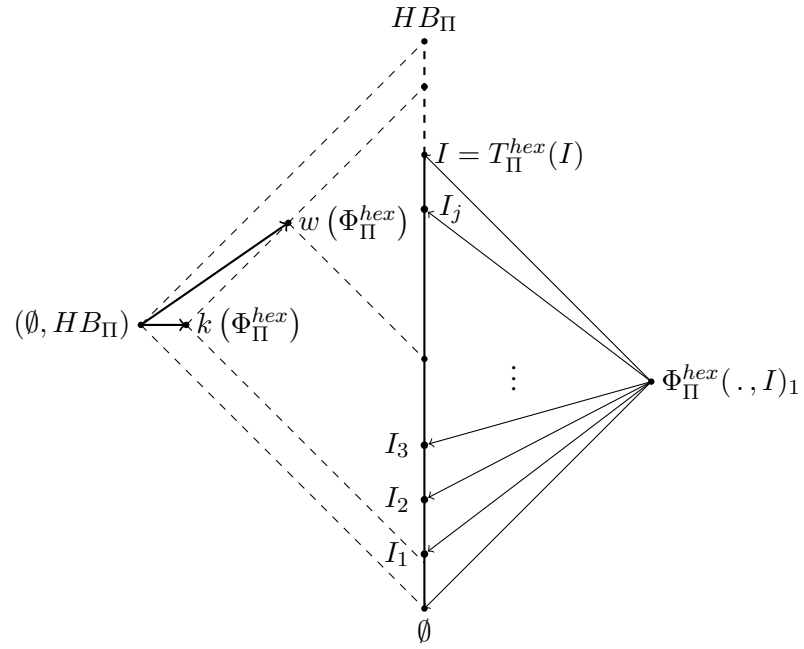

Figure 4.1: Illustration of the relations between the Φ_Π^{hex}-Kripke-Kleene-, the Φ_Π^{hex}-well-founded, and the 2-valued Φ_Π^{hex}-answer-set semantics. On the left side: (i) the *Kripe-Kleene fixpoint* $k\left(\Phi_\Pi^{hex}\right)$ is the least fixpoint of Φ_Π^{hex}; (ii) the *well-founded fixpoint (least 3-valued stable fixpoint)* $w\left(\Phi_\Pi^{hex}\right)$ is the least fixpoint of $\Phi_\Pi^{hex,\downarrow\uparrow}$. On the right side: monotone iteration of the *2-valued* Φ_Π^{hex}*-stable model* I. If we replace Φ_Π^{hex} by $\mathcal{T}_\Pi^{hex}$, we obtain the more precise *ultimate* semantics.

CHAPTER 5

Semantic Properties of Normal HEX Programs

Given a classical normal program Π, we noticed in Example 3.3.7 that I is an answer set of Π iff I is a fixpoint of T_Π and $\Phi^{\downarrow}_\Pi(I) = I$ (Denecker et al., 2000b). Likewise, in Section 5.1 we show that for each positive (i.e., negation-free) normal HEX program Π containing only *monotone* external atoms, the same characterization, in terms of Φ^{hex}_Π, holds.

However, in Section 5.2 we show that if Π contains nonmonotone external atoms, 2-valued Φ^{hex}_Π-answer-set semantics and FLP-answer-set semantics do, in general, not coincide. More precisely, each Φ^{hex}_Π-answer set is also an FLP-answer set (see Theorem 5.2.5), whereas the converse may not hold (see Example 5.2.6). We argue in Section 5.4 that this divergence is due to the well-supportedness (Shen, 2011) of 2-valued Φ^{hex}_Π-answer sets.

Finally, in Section 5.3 we show that for every normal HEX program Π, the (ultimate) well-founded model of Π approximates each FLP-answer set of Π in the sense that each atom *true* (resp., *false*) in $w\left(\Phi^{hex}_\Pi\right)$ (resp., $w\left(\Phi^{hex}_\Pi\right)$), is *true* (resp., *false*) in each FLP-answer set of Π.

5.1 Monotone Normal HEX Programs Have Nice Properties

In this section we show that standard the FLP-answer-set semantics of normal HEX programs containing only monotone external atoms coincides with the 2-valued Φ^{hex}_Π-answer-set semantics.

Let Π be a normal HEX program with language $\mathcal{L}^{hex}_\Pi = (\Sigma_\Pi, \Sigma^{\#}_\Pi)$. Let $\mathbf{i} \in \Sigma^m_\Pi$, and $\mathbf{o} \in \Sigma^n_\Pi$. We say that an external atom $f^{\#} \in \Sigma^{\#,(m,n)}$ is *monotone* if for every $J, I \in \mathcal{I}_\Pi$ such that $J \subseteq I$,

$$f\left(J, \mathbf{i}, \mathbf{o}\right) \leq f\left(I, \mathbf{i}, \mathbf{o}\right).$$

Furthermore, we say that Π is *monotone* if every external atom $a \in \Lambda^{\#}_\Pi$ is monotone.

In the sequel let Π be a monotone normal HEX program. By definition of $\langle\ .\ \rangle_{(I_1,I_2)}$ (see Section 4.1), for every *monotone* external atom $f^{\#}\left[\mathbf{i}\right](\mathbf{o})$ and 3-valued interpretation (I_1, I_2),

we have

$$\left\langle f^{\#}\left[\mathbf{i}\right](\mathbf{o})\right\rangle_{(I_1,I_2)} = \begin{cases} \mathbf{t} & \text{if } f\left(I_1,\mathbf{i},\mathbf{o}\right)=\mathbf{t}, \\ \mathbf{f} & \text{if } f\left(I_2,\mathbf{i},\mathbf{o}\right)=\mathbf{f}, \\ \mathbf{u} & \text{otherwise.} \end{cases} \tag{5.1}$$

Roughly, in the case of a monotone external atom a, it suffices to evaluate a on the boundaries of the interval $[I_1, I_2]$. This property indicates that every monotone external atom behaves similar to an ordinary atom.

Example 5.1.1. In Example 4.1.7 we observed that given an atom $a \in \Lambda_\Pi$, the corresponding external atom $a^{\#}$ (see Example 2.2.1) is monotone.

We now extend the Gelfond-Lifschitz reduct and Gelfond-Lifschitz operator (see Example 3.3.7) to the class of normal HEX programs.

Definition 5.1.2 (Gelfond-Lifschitz reduct)**.** Let Π be a normal HEX program. Define the *Gelfond-Lifschitz reduct* (Gelfond and Lifschitz, 1988) of Π with respect to I by

$$\Pi^I = \left\{H(r) \leftarrow B^+(r) : r \in \Pi : I \models B^{\sim}(r)\right\}.$$

Intuitively, we compute the reduct Π^I by (i) deleting every rule $r \in \Pi$, where $I \models b$ for some negated atom $\sim b \in B^{\sim}(r)$, and (ii) deleting the negative body of the remaining rules. This transformation yields a positive (i.e., negation-free) HEX program Π^I with only monotone external atoms. Consequently, the extended van Emden-Kowalski operator $T^{hex}_{\Pi^I}$ of Π^I is monotone. Hence, by the theorem of Tarski and Knaster (Tarski, 1955), $T^{hex}_{\Pi^I}$ has a least fixpoint, given by $\mathrm{lfp}(T^{hex}_{\Pi^I})$. Therefore, we can extend the classical Gelfond-Lifschitz operator (Gelfond and Lifschitz, 1988) as follows.

Definition 5.1.3 (Extended Gelfond-Lifschitz operator)**.** Let Π be a normal HEX program. Define the *extended Gelfond-Lifschitz operator* of Π, for each $I \in \mathcal{I}_\Pi$, by

$$\Gamma^{hex}_\Pi(I) = \mathrm{lfp}(T^{hex}_{\Pi^I}).$$

The next result states that Γ^{hex}_Π is an extension of the traditional Gelfond-Lifschitz operator (Gelfond and Lifschitz, 1988) Γ_Π.

Proposition 5.1.4. *Let Π be a classical normal program. Then, $\Gamma^{hex}_\Pi = \Gamma_\Pi$.*

Given a monotone normal HEX program Π and an interpretation $I \in \mathcal{I}_\Pi$, we say that I is an Γ^{hex}_Π*-answer set* of Π if $\Gamma^{hex}_\Pi(I) = I$.

As an immediate consequence of Proposition 5.1.4 we obtain the following result.

Proposition 5.1.5. *Let Π be a classical normal program, and let $I \in \mathcal{I}_\Pi$. Then, I is an Γ^{hex}_Π-answer set iff I is an Γ_Π-answer set.*

Example 5.1.6. Consider the monotone normal HEX program Π consisting of the single rule

$$a \leftarrow \top^{\#}, \sim \bot^{\#},$$

where $\top \equiv \mathbf{t}$, and $\bot \equiv \mathbf{f}$ (i.e., $\top(I) = \mathbf{t}$ and $\bot(I) = \mathbf{f}$ for each $I \in \mathcal{I}_\Pi$). Let $I = \{a\}$. We have $\Pi^I = \{a \leftarrow \top^{\#}\}$. Since $T^{hex}_{\Pi^I}(\emptyset) = I$ and $T^{hex}_{\Pi^I}(I) = I$, we have $\Gamma^{hex}_\Pi(I) = I$ and I is thus an Γ^{hex}_Π-answer set of Π.

Example 5.1.7. Let Π be the monotone normal HEX program consisting of the following single rule:

$$a \leftarrow a^{\#}, \sim \bot^{\#}.$$

Let $I = \{a\}$. Since $\Pi^I = \{a \leftarrow a^{\#}\}$, $T^{hex}_\Pi(\emptyset) = \emptyset$ yields $\Gamma^{hex}_\Pi(I) = \emptyset$, i.e., I is not an Γ^{hex}_Π-answer set of Π.

Given a *classical* normal program Π, the operator $\Phi^{\downarrow}_\Pi$ coincides with the Gelfond-Lifschitz operator Γ_Π (see Example 3.3.7). That is, the Fitting approximation (Fitting, 1985) Φ_Π is designed to simulate T_{Π^I}, i.e., for each $I \in \mathcal{I}_\Pi$, $\Phi_\Pi(\,.\,, I)_1 = T_{\Pi^I}$. The next result states that for monotone normal HEX programs, this correspondence remains valid.

Proposition 5.1.8. *Let Π be a monotone normal HEX program. Then, $\Gamma^{hex}_\Pi = \Phi^{hex,\downarrow}_\Pi$.*

Proof (Sketch). Given an interpretation $I \in \mathcal{I}_\Pi$, it is straightforward to prove

$$\Phi^{hex}_\Pi(\,.\,, I)_1 = T^{hex}_{\Pi^I}. \tag{5.2}$$

The rest follows directly from definitions. □

The next result shows that Γ^{hex}_Π-answer-set semantics characterize FLP-answer-set semantics.

Theorem 5.1.9. *Let Π be a monotone normal HEX program, and let $I \in \mathcal{I}_\Pi$. Then, I is an Γ^{hex}_Π-answer set iff I is an FLP-answer set of Π.*

Proof. ($\Rightarrow$) First, we show that I is a model of $f\Pi^I$. By assumption, we have $T^{hex}_{\Pi^I}(I) = I$. Hence, by Proposition 4.1.3, I is a model of Π^I. That is, whenever $I \models B^+(r)$ for some $r \in \Pi^I$, we have $H(r) \in I$. Clearly, this implies that I is also a model of $f\Pi^I$.

Second, we show that I is a minimal such model. Suppose, towards a contradiction, there exists some $I' \subsetneq I$ such that I' is a model of $f\Pi^I$. We claim $T^{hex}_{\Pi^I}(I') \subseteq I'$. Let $a \in T^{hex}_{\Pi^I}(I')$. Then there exists a rule $r \in \Pi$ such that $H(r) = a$, $I' \models B^+(r)$, and $I \models B^{\sim}(r) = \emptyset$. Hence, $I \models B(r)$ (i.e., $r \in f\Pi^I$) and, since I' is a model of $f\Pi^I$, $a \in I'$. This proves the claim and, by Proposition 4.1.3, it follows that I' is a model of Π^I, contradicting the minimality of I.

($\Leftarrow$) Suppose $\Gamma^{hex}_\Pi(I) = I' \neq I$. Since I is a fixpoint of $T^{hex}_{\Pi^I}$, and I' is the least fixpoint of $T^{hex}_{\Pi^I}$, $I' \subseteq I$. Suppose $I' \subsetneq I$. Since $T^{hex}_{\Pi^I}(I') = I'$, I' is a model of Π^I. Let $r \in f\Pi^I$. Notice that $H(r) \leftarrow B^+(r) \in \Pi^I$. Since I' is a model of Π^I, $I \models B^+(r)$, and $I \models H(r)$; moreover, since $I' \subseteq I$ and $I \cap B^-(r) = \emptyset$, $I' \cap B^-(r) = \emptyset$. Hence, since $r \in f\Pi^I$ was arbitrary, I' is a model of $f\Pi^I$, a contradiction to the minimality of I. □

We are now ready to prove that for monotone normal HEX programs Π, the FLP-answer sets and the 2-valued Φ^{hex}_{Π}-answer sets coincide.

Theorem 5.1.10. *Let Π be a monotone normal HEX program, and let $I \in \mathcal{I}_\Pi$. Then, I is an Φ^{hex}_{Π}-answer set iff I is an FLP-answer set of Π.*

Proof. Is an immediate consequence of Proposition 5.1.8 and Theorem 5.1.9. □

5.2 Two-Valued Answer Sets are FLP-Answer Sets

For every classical normal program Π, the answer sets of Π are characterized by the 2-valued Φ_Π-answer sets (see Example 3.3.7). Moreover, since the answer sets of Π coincide with the FLP-answer sets of Π, the 2-valued Φ_Π-answer sets also characterize the FLP-answer sets of Π.

In this section we show that in the case of normal HEX programs, this correspondence does, in general, not hold. Let Π be a (normal or disjunctive) HEX program. Recall from Section 2.2 that the FLP-reduct (Faber et al., 2004, 2011) of Π relative to an interpretation $I \in \mathcal{I}_\Pi$ is the set of all rules in Π such that I satisfies the body of r, i.e., $I \models B(r)$. We now extend this definition to 3-valued interpretations.

Definition 5.2.1 (Three-valued FLP-reduct)**.** Let Π be a normal HEX program, and let $(I_1, I_2) \in \mathcal{I}^c_\Pi$. Define the *3-valued FLP-reduct* of Π relative to (I_1, I_2) by

$$f\Pi^{(I_1,I_2)} = \left\{ r \in \Pi : \langle\, B(r) \,\rangle_{(I_1,I_2)} = \mathbf{t} \right\}.$$

The following proposition summarizes some basic properties of the 3-valued FLP-reduct.

Proposition 5.2.2. *Let Π be a normal or disjunctive HEX program.*

1. *For every $I \in \mathcal{I}_\Pi$, $f\Pi^{(I,I)} = f\Pi^I$.*
2. *For every $(I_1, I_2) \in \mathcal{I}^c_\Pi$, and let $I \in \mathcal{I}_\Pi$ such that $(I_1, I_2) \subseteq_p I$, $f\Pi^{(I_1,I_2)} \subseteq f\Pi^I$.*

Proof. The first assertion is an immediate consequence of (4.3) and (4.5). For the second inclusion, since $(I_1, I_2) \subseteq_p I$, we have by Lemma 4.1.10 that if $\langle\, B(r) \,\rangle_{(I_1,I_2)} = \mathbf{t}$ then $\langle\, B(r) \,\rangle_I = \mathbf{t}$ which proves the assertion. □

The next result shows the correspondence between the 3-valued FLP-reduct and the extended Fitting operator.

Lemma 5.2.3. *Let Π be a normal or disjunctive HEX program, and let $(I_1, I_2) \in \mathcal{I}^c_\Pi$. Then,*

$$\Phi^{hex}_{\Pi}(I_1, I_2)_1 = \Phi^{hex}_{f\Pi^{(I_1,I_2)}}(I_1, I_2)_1.$$

Proof. Follows directly from definitions. □

The next lemma relates $f\Pi^I$ and Π in terms of the extended Fitting operator.

Lemma 5.2.4. *Let Π be a normal or disjunctive HEX program. For every $(I_1, I_2) \in \mathcal{I}^c_\Pi$, and let $I \in \mathcal{I}_\Pi$ such that $(I_1, I_2) \subseteq_p I$, we have*

$$\Phi^{hex}_\Pi(I_1, I_2)_1 = \Phi^{hex}_{f\Pi^I}(I_1, I_2)_1.$$

Proof. It is clear that more rules entail more facts, i.e., if $\Pi' \subseteq \Pi$, then $\Phi^{hex}_{\Pi'}(I_1, I_2)_1 \subseteq \Phi^{hex}_\Pi(I_1, I_2)_1$. Moreover, by Proposition 5.2.2, we have the following inclusions

$$f\Pi^{(I_1,I_2)} \subseteq f\Pi^I \subseteq \Pi.$$

Consequently, we have

$$\Phi^{hex}_{f\Pi^{(I_1,I_2)}}(I_1, I_2)_1 \subseteq \Phi^{hex}_{f\Pi^I}(I_1, I_2)_1 \subseteq \Phi^{hex}_\Pi(I_1, I_2)_1, \tag{5.3}$$

and, by Lemma 5.2.3, $\Phi^{hex}_{f\Pi^{(I_1,I_2)}}(I_1, I_2)_1 = \Phi^{hex}_\Pi(I_1, I_2)_1$ which together with (5.3) entails

$$\Phi^{hex}_{f\Pi^I}(I_1, I_2)_1 = \Phi^{hex}_\Pi(I_1, I_2)_1.$$

□

We are now ready to prove that, given a *normal* HEX program Π, every 2-valued Φ^{hex}_Π-answer set $I \in \mathcal{I}_\Pi$ is also an FLP-answer set of Π.

Theorem 5.2.5. *Let Π be a normal HEX program and $I \in \mathcal{I}_\Pi$. If I is a 2-valued Φ^{hex}_Π-answer set, then I is an FLP-answer set of Π.*

Proof. By assumption, we have $\Phi^{hex,\downarrow}_\Pi(I) = I$, i.e., I is the least fixpoint of $\Phi^{hex}_\Pi(\,.\,, I)_1$. By Lemma 5.2.4, we have $\Phi^{hex}_\Pi(\,.\,, I)_1 = \Phi^{hex}_{f\Pi^I}(\,.\,, I)_1$ and, hence, $\Phi^{hex,\downarrow}_{f\Pi^I}(I) = I$.

Since every 2-valued Φ^{hex}_Π-answer set is a model of Π, I is a model of $f\Pi^I$. It remains to show that I is a minimal model of $f\Pi^I$. Suppose there exists some $I' \subsetneq I$ such that I' is a model of $f\Pi^I$. Then, by Proposition 4.1.3, $T^{hex}_{f\Pi^I}(I') \subseteq I'$. Hence, by Proposition 4.2.2, $\Phi^{hex,\downarrow}_{f\Pi^I}(I) \subseteq I' \subsetneq I$, a contradiction. Consequently, I is a minimal model of $f\Pi^I$ and, thus, an FLP-answer set of Π. □

Recall from Section 3.4 that even in the classical case, not every ultimate answer set is also an Φ_Π-answer set (see Example 3.4.5); hence, in Theorem 5.2.5 we cannot replace Φ^{hex}_Π by $\mathcal{T}^{hex}_\Pi$.

In the next example we show that the converse of Theorem 5.2.5 does, in general, not hold.

Example 5.2.6. Let Π be the normal HEX program consisting of the following rules:

$$a \leftarrow f^{\#}[a, b]$$
$$b \leftarrow g^{\#}[a, b]$$

where f and g are defined as in Table 5.1. Let $I = \{a, b\}$. It is easy to verify that I is a minimal model of $f\Pi^I = \Pi$ and, hence, an FLP-answer set of Π. In contrast, we have

$$\begin{aligned}\mathcal{T}_\Pi^{hex}(\emptyset, I)_1 &= T_\Pi^{hex}(\emptyset) \cap T_\Pi^{hex}(\{a\}) \cap T_\Pi^{hex}(\{b\}) \cap T_\Pi^{hex}(I) \\ &= I \cap \{b\} \cap \{a\} \cap I \\ &= \emptyset.\end{aligned}$$

That is, $\mathcal{T}_\Pi^{hex,\downarrow}(I) = \emptyset$. Consequently, by Proposition 3.3.6, I is not an ultimate answer set of Π and, hence, by Proposition 4.4.4, not an Φ_Π^{hex}-answer set.

J	$f(J, a, b)$	$g(J, a, b)$
$\emptyset$	**t**	**t**
$\{a\}$	**f**	**t**
$\{b\}$	**t**	**f**
$\{a, b\}$	**t**	**t**

Table 5.1: Definition of the interpretation functions f and g of Example 5.2.6.

Intuitively, the divergence of 2-valued answer-set semantics based on Approximation Theory, and FLP-answer-set semantics, is due to the "non-constructiveness" of FLP-semantics. More precisely, given an FLP-answer set I of a normal HEX program Π, in general we cannot define a *level mapping* (Hitzler and Wendt, 2005) on I such that every atom $a \in I$ is justified only by atoms of a lower level. The intuition behind "constructiveness" is formalized by the concept of *well-supportedness* (Shen, 2011; Fages, 1994) given in Section 5.4. Indeed, the FLP-answer set I of Example 5.2.6 is not well-supported (see Example 5.4.4), whereas every 2-valued Φ_Π^{hex}-, and ultimate answer set is well-supported (see Theorem 5.4.7).

5.3 Well-Founded Semantics Approximate FLP-Answer-Set Semantics

In classical logic programming, the well-founded model $w(\Pi)$, of a given classical normal program Π, approximates[1] every answer set of Π in the following sense (Van Gelder et al., 1991): an atom $a \in \Lambda_\Pi$ is well-founded (resp., unfounded) only if $a \in I$ for every (resp., no) answer set I of Π. Moreover, since in the classical case, FLP-answer-set semantics coincide with traditional answer set semantics (Faber et al., 2004, 2011), the well-founded model $w(\Pi)$ also approximates every FLP-answer set of Π.

The results in Approximation Theory guarantee that these relations between well-founded and answer-set semantics (based on approximations of T_Π^{hex}) also hold for normal HEX programs. However, in the previous section we have seen that for this class of programs, (ultimate) answer-set semantics and FLP-answer-set semantics do not coincide. Therefore, the question

[1] Here we use the term "approximates" in the intuitive sense.

naturally arises, whether (ultimate) well-founded semantics of normal HEX programs approximate FLP-answer-set semantics in the sense described above.

The following theorem shows that every atom *true* (resp., *false*) in the (ultimate) well-founded model is also *true* (resp., *false*) in each FLP-answer set. However, as a direct consequence of Theorem 5.2.5, Example 5.3.3 shows that not every atom *true* (resp., *false*) in each FLP-answer is also *true* (resp., *false*) in the (ultimate) well-founded model. Arguably, this indicates that (ultimate) well-founded semantics based on Approximation Theory are only partially compatible with FLP-answer-set semantics.

Theorem 5.3.1. *Let* Π *be a normal HEX program. For each FLP-answer set* I *of* Π, $w\left(\mathcal{T}_\Pi^{hex}\right) \subseteq_p I$.

Proof. Let $((I_1^\alpha, I_2^\alpha))_{\alpha \geq 0}$ be the sequence used to define $w\left(\mathcal{T}_\Pi^{hex}\right)$. Then it suffices to prove that for each ordinal α, $(I_1^\alpha, I_2^\alpha) \subseteq_p I$.

We proceed by (transfinite) induction as follows. Clearly, $(I_1^0, I_2^0) = (\emptyset, \Lambda_\Pi) \subseteq_p I$, which proves the induction base. For the induction step, assume $\alpha = \beta + 1$ and $I_1^\beta \subseteq I \subseteq I_2^\beta$. We have to show

$$\mathcal{T}_\Pi^{hex,\downarrow}(I_2^\beta) \subseteq I \subseteq \mathcal{T}_\Pi^{hex,\uparrow}(I_1^\beta). \tag{5.4}$$

Since I is a fixpoint of T_Π^{hex} and a subset of I_2^β, the first inclusion is an immediate consequence of Proposition 3.3.1.

For the second inclusion, notice that by definition we have $I_2^\alpha = \mathcal{T}_\Pi^{hex,\uparrow}(I_1^\beta)$. In particular, we have

$$I_2^\alpha = \bigcup_{I_1^\beta \subseteq J \subseteq I_2^\alpha} T_\Pi^{hex}(J). \tag{5.5}$$

Let $I' = I_2^\alpha \cap I$. We claim that $I' = I$. By assumption, we have $I_1^\beta \subseteq I$, and since (I_1^β, I_2^β) is $\mathcal{T}_\Pi^{hex}$-prudent, we have $I_1^\beta \subseteq I_2^\alpha$. Consequently, we have $I_1^\beta \subseteq I'$. Moreover, by definition of I', we have $I' \subseteq I_2^\alpha$. Hence, $I_1^\beta \subseteq I' \subseteq I_2^\alpha$ which together with (5.5) entails $T_\Pi^{hex}(I') \subseteq I_2^\alpha$. That is, for every rule $r \in \Pi$, whenever $I' \models B(r)$, $H(r) \in I_2^\alpha$. Since I is, by assumption, an FLP-answer set of Π, for every rule $r \in f\Pi^I$, $H(r) \in I$. Hence, for every rule $r \in f\Pi^I \subseteq \Pi$ such that $I' \models B(r)$, $H(r) \in I_2^\alpha$ and $H(r) \in I$ which is equivalent to $H(r) \in I'$. Thus, I' is a model of $f\Pi^I$. Since I is a minimal model of $f\Pi^I$ and $I' \subseteq I$, we conclude $I' = I$; by definition of I', this implies $I = I_2^\alpha \cap I$ which is equivalent to $I \subseteq I_2^\alpha = \mathcal{T}_\Pi^{hex,\uparrow}(I_1^\beta)$.

The case where α is a limit ordinal is straightforward. □

Corollary 5.3.2. *Let* Π *be a normal HEX program. For each FLP-answer set* I *of* Π, $w\left(\Phi_\Pi^{hex}\right) \subseteq_p I$.

Proof. Is a direct consequence of in Theorem 5.3.1 and (2) in Proposition 4.4.4. □

Example 5.3.3. Reconsider the normal HEX program Π of Example 5.2.6 consisting of the following rules:

$$a \leftarrow f^\#[a, b]$$
$$b \leftarrow g^\#[a, b]$$

where $f^{\#}$ and $g^{\#}$ are defined as in Table 5.1. In Example 5.2.6 we have seen that $I = \{a, b\}$ is an FLP-answer set of Π. However, since $\mathcal{T}_{\Pi}^{hex,\downarrow\uparrow}(\emptyset, \{a, b\}) = (\emptyset, \{a, b\})$,

$$w\left(\mathcal{T}_{\Pi}^{hex}\right) = (\emptyset, \{a, b\}) = w\left(\Phi_{\Pi}^{hex}\right),$$

i.e., a and b are both *undefined* in the (ultimate) well-founded model.

5.4 Two-Valued Answer-Sets are Well-Supported

It is well-known that answer sets (and, consequently, FLP-answer sets) of classical normal programs are characterized by *well-supported* (Fages, 1994) models. Intuitively, a supported model I is well-supported, if I is free of circular justifications. Therefore, no atom contained in any answer set is self-justified.

However, in the case of normal HEX programs, Shen (2011) shows that FLP-answer-set semantics is not free of circular justifications (see Example 5.4.1). In this section, we extend the notion of well-supportedness for classical normal programs (Fages, 1994) to the class of normal HEX programs (Shen, 2011), and show that 2-valued (ultimate) answer sets are well-supported.

The next example shows that standard FLP-answer sets may contain self-supported atoms (Shen, 2011). Following the notation in (Shen, 2011), we use the symbol $\Leftarrow$ to express the meta-property "truth is supported by" and we use standard classical connectives (i.e., $\wedge$, $\vee$, $\neg$) to form "meta-formulas".

Example 5.4.1. Reconsider the normal HEX program Π of Example 5.2.6 consisting of the following rules:

$$a \leftarrow f^{\#}[a, b]$$
$$b \leftarrow g^{\#}[a, b]$$

where $f^{\#}$ and $g^{\#}$ are defined as in Table 5.1. We have seen in Example 5.2.6 that $I = \{a, b\}$, is an FLP-answer set of Π. We argue that a and b are both circular justified in I. Since the first rule is the only rule with a in its head, $a \Leftarrow f^{\#}$. By definition of f (see Table 5.1), we have $f^{\#} \Leftarrow b \vee (\neg a \wedge \neg b)$. Since $\neg a$ cannot be a justification for a in I, we conclude $f^{\#} \Leftarrow b$ and, hence, $a \Leftarrow f^{\#} \Leftarrow b$. Analogously, we have $b \Leftarrow g^{\#} \Leftarrow a$. Consequently, we have the following self-supported loops: $a \Leftarrow f^{\#} \Leftarrow b \Leftarrow g^{\#} \Leftarrow a$, and $b \Leftarrow g^{\#} \Leftarrow a \Leftarrow f^{\#} \Leftarrow b$. That is, a and b are both self-supported. In Example 5.4.4 we will see that I is indeed not well-supported.

We now extend Fages' definition of well-supportedness (Fages, 1994) to the case of normal HEX programs (Shen, 2011). We say that a binary relation $\prec$ is a *well-founded partial order* on a set X, if there exists no infinite decreasing chain $x_1 \succ x_2 \succ x_3 ...$ in X.

Definition 5.4.2 (Well-supportedness)**.** Let Π be a normal HEX program and let $I \in \mathcal{I}_{\Pi}$. We say that I is *well-supported* (Shen, 2011), if there exists a strict well-founded partial order $\prec$ on I such that for every atom $a \in I$, there exists a rule $r \in \Pi$ with $H(r) = a$ and a proper subset $J \subsetneq I$ such that

1. $\langle\, B(r) \,\rangle_{(J,I)} = \mathbf{t}$, and
2. $J \prec H(r)$,

where $J \prec H(r)$ if $j \prec H(r)$ for every $j \in J$.

The next proposition states that this definition of well-supportedness is an extension of Fages' original definition.

Proposition 5.4.3 (Shen (2011))**.** *Let Π be a classical normal program, and let $I \in \mathcal{I}_\Pi$. Then, I is well-supported iff I is well-supported under the definition given in (Fages, 1994).*

Example 5.4.4. We show that the FLP-answer set I of Example 5.4.1 (see also Example 5.2.6) is not well-supported. Let $b \prec a$. Since the rule $b \leftarrow g^{\#}[a, b]$ is the only rule in Π with $H(r) = b$, for I to be well-supported we have to find a proper subset $J \subsetneq I$ such that conditions (1)-(2) are satisfied. Since $\langle\, B(r) \,\rangle_{(\emptyset,I)} \neq \mathbf{t}$, $\{a\} \not\prec b$, and $\{b\} \not\prec b$, we conclude that there exists no such J. The case $a \prec b$ is analogous. If a and b are incomparable, J has to be the empty set, but $\langle\, B(r) \,\rangle_{(\emptyset,I)} \neq \mathbf{t}$ for both rules. Hence, I is not well-supported.

We now define level mappings (Hitzler and Wendt, 2005) which are closely related to the notion of well-supportedness (see Remark 5.4.6).

Definition 5.4.5 (Level-mapping)**.** Let Π be a HEX program. We call every function $\ell : \Lambda_\Pi \longrightarrow \alpha$, for a (countable) ordinal α, a *level mapping* (Hitzler and Wendt, 2005) for Π.

Remark 5.4.6. We can reformulate the definition of well-supportedness in terms of level mappings as follows: given a normal HEX program Π, an interpretation $I \in \mathcal{I}_\Pi$ is well-supported iff there exists a level mapping ℓ for Π such that for each $a \in I$ there exists a rule $r \in \Pi$ with $H(r) = a$ and $\ell(a) > \ell(b)$ for each $b \in B^+(r)$.

We have seen that FLP-answer sets may contain circular justified atoms. Now the question arises, whether 2-valued Φ^{hex}_Π-answer sets are well-supported. The next theorem answers these questions positively.

Theorem 5.4.7. *Let Π be a normal HEX program, and let $I \in \mathcal{I}_\Pi$. If I is a 2-valued Φ^{hex}_Π-answer set, then I is well-supported.*

Proof. By Remark 5.4.6, we have to find an appropriate level mapping ℓ for I. By assumption, we have $\Phi^{hex,\downarrow}_\Pi(I) = I$, and recall that $\Phi^{hex,\downarrow}_\Pi(I)$ is the least fixpoint of $\Phi^{hex}_\Pi(\,.\,, I)_1$ which the limit of the transfinite sequence[2] (see Definition 4.2.1)

$$J_0 \subsetneq J_1 \subsetneq \ldots \subsetneq J_\alpha \subsetneq \ldots \subsetneq I$$

where $J_0 = \emptyset$, $J_\alpha = \Phi^{hex}_\Pi(J_{\alpha-1}, I)_1$ for every $\alpha \geq 1$, and $I = \bigcup_{\alpha \geq 0} J_\alpha$. For each $a \in I$ there exists a successor ordinal $\alpha(a)$ such that $a \in J_{\alpha(a)}$ but $a \notin J_\beta$ for each $\beta < \alpha(a)$. Define, for each $a \in I$, $\ell(a) = \alpha(a)$ (and extend ℓ to Λ_Π arbitrary). Then, by construction of I and by definition of $\Phi^{hex}_\Pi(\,.\,, I)_1$, we have that ℓ is a level mapping fulfilling the conditions of Remark 5.4.6. □

[2] In fact, we prove the more general case where I can be infinite (recall that in this thesis we consider only finite programs and, hence, only finite interpretations).

Intuitively, the well-supportedness of 2-valued answer sets is due to the constructive condition of stability (Proposition 3.3.6) which induces a level mapping on answer sets.

Example 5.4.8. Reconsider the normal HEX program of Example 4.1.15 consisting of the following rules:

$$a \leftarrow\sim \bot^{\#}$$
$$b \leftarrow a^{\#}.$$

In Example 4.1.15 we have seen that $T_{\Pi}^{hex}(\{a,b\}) = \{a,b\}$. First, we show that $I = \Lambda_{\Pi}$ is a 2-valued Φ_{Π}^{hex}-answer set by computing $\Phi_{\Pi}^{hex,\downarrow}$ as follows:

$$\Phi_{\Pi}^{hex}(\emptyset, I)_1 = \{a\} = J_1$$
$$\Phi_{\Pi}^{hex}(\{a\}, I)_1 = I$$
$$\Phi_{\Pi}^{hex}(I, I)_1 = I.$$

That is, $\Phi_{\Pi}^{hex,\downarrow}(I) = I$. Second, we define the strict well-founded partial order $\prec$ on I by $a \prec b$ and, finally, show that I is indeed well-supported: (i) for $a \in I$, let r be the first rule, and observe $\langle\, B(r)\,\rangle_{(\emptyset,I)} = \mathbf{t}$ and $\emptyset \prec H(r)$; (ii) for $b \in I$ let r be the second rule, and observe $\langle\, B(r)\,\rangle_{(\{a\},I)} = \mathbf{t}$ and $\{a\} \prec H(r)$. Consequently, I is well-supported.

CHAPTER 6

Fixpoint Semantics of Disjunctive HEX programs

In Chapter 4 we defined fixpoint semantics for normal (i.e., disjunction-free) HEX programs by exploiting the machinery of Approximation Theory. For that we extended the well-known van Emden-Kowalski operator of classical normal programs to the class of HEX programs, while preserving the applicability of Approximation Theory.[1]

However, for HEX programs containing disjunctive rules, the extended van Emden-Kowalski operator is no longer a lattice operator. More precisely, if a HEX program Π contains disjunctions (i.e., rules with disjunctive heads), then the result of applying T_Π^{hex} to some interpretation I may contain disjunctive facts which are not part of the Herbrand base of Π. In that case, since Approximation Theory studies fixpoints of lattice operators, the theory is not applicable. Hence, we cannot straightforwardly define fixpoint semantics of *disjunctive* HEX programs in terms of approximations of T_Π^{hex}.

In the literature, there exist two well-known extensions of the *classical* van Emden-Kowalski operator T_Π to the class of classical disjunctive programs:

1. Minker and Rajasekar (1990) (see also (Lobo et al., 1992)) introduced, for a *positive* classical disjunctive program Π, the operator T_Π^s based on *hyperresolution* and defined on the complete lattice $\mathfrak{P}(DHB_\Pi)$ ordered by set inclusion, where DHB_Π is the *extended* or *disjunctive* Herbrand base of Π consisting of all disjunctions that can be formed with atoms from HB_Π. Finite subsets of $\mathfrak{P}(DHB_\Pi)$ are called *model states*, so T_Π^s is an operator mapping model states to model states.

2. Fernández and Minker (1995) defined a model theoretic operator T_Π^M on the complete partially ordered set $(\mathfrak{P}_{min}(\mathcal{I}_\Pi), \sqsubseteq)$ where $\mathfrak{P}_{min}(\mathcal{I}_\Pi)$ contains the minimal elements from

[1] For every normal HEX program Π, the operator T_Π^{hex} is a *lattice* operator defined on the complete lattice of all Herbrand interpretations of Π (see Section 4.1).

$\mathfrak{P}(\mathcal{I}_\Pi)$ and $\sqsubseteq$ denotes the subsumption relation (Fernández and Minker, 1995) (see Section 6.1). The operator T_Π^M can be considered as a *non-deterministic* extension of T_Π which maps a set of possible interpretations $\mathcal{J}$ to a set of possible outcomes $T_\Pi^M(\mathcal{J})$.

Although T_Π^s and T_Π^M both extend T_Π and have similar "nice" properties as T_Π (Minker and Rajasekar, 1990; Lobo et al., 1992; Fernández and Minker, 1995; Seipel et al., 1997), we cannot directly apply Approximation Theory to either of T_Π^s or T_Π^M for the following reasons: T_Π^s is defined only for positive disjunctive programs (i.e., programs without negation as failure (Clark, 1978)) and T_Π^M is not a lattice operator.

However, by combining ideas from classical disjunctive logic programming and Approximation Theory, Pelov (2004); Pelov and Truszczyński (2004) extended parts of the Approximation Theory to the case of *non-deterministic* operators. In detail, Pelov (2004) and Pelov and Truszczyński (2004) defined a non-deterministic operator[2] $N_\Pi^{Sel} : \mathcal{I}_\Pi \longrightarrow \mathfrak{P}(\mathcal{I}_\Pi)$ by

$$N_\Pi^{Sel}(I) = Sel\left(T_\Pi(I)\right), \tag{6.1}$$

where Sel is a *selection function* selecting a subset from the models of $T_\Pi(I)$.

We define in this chapter an operator N_Π^{hex}, which is similar though not identical to N_Π^{Sel}, as follows:

1. In contrast to (Pelov, 2004; Pelov and Truszczyński, 2004), we restrict the definition of N_Π^{hex} to the selection function MM selecting, for a given set D of disjunctive facts (or clauses), the *minimal models* of D.
2. In (Pelov and Truszczyński, 2004) the authors argue that the notion of *computation* (Marek et al., 2004) is an adequate formalization of the process of iterating a non-deterministic operator in a bottom-up manner.[3] However, Pelov and Truszczyński showed that if N_Π is defined as in (6.1), then computations[4] of N_Π^{Sel} generally do not yield the expected results. In contrast, we define N_Π^{hex} in such a way that it has similar properties as N_Π^{Sel}, and additionally can be iterated in terms of computations (see Section 7.2).

The rest of this chapter is organized as follows. In Section 6.1 we define, for a disjunctive HEX program Π, the non-deterministic van Emden-Kowalski operator N_Π^{hex} representing one step of logical derivation. To this end, we define the Smyth ordering (Smyth, 1978) $\sqsubseteq$ on $\mathfrak{P}(\mathcal{I}_\Pi)$ as in (Fernández and Minker, 1995), and show that N_Π^{hex} has similar properties as its normal counterpart T_Π^{hex}. In particular, (minimal) pre-fixpoints of N_Π^{hex} characterize (minimal) models of Π.

In Section 6.2 we define the notion of computation (Marek et al., 2004) for N_Π^{hex} and show that for every monotone and positive HEX program Π, N_Π^{hex} is monotone and thus every minimal model of Π is derivable.

[2] To be more precise, Pelov (2004); Pelov and Truszczyński (2004) defined an operator N_Π^{aggr} for logic programs with aggregates.

[3] Notice that $N_\Pi^{hex}(N_\Pi^{hex}(I))$ is not well-defined, i.e., we cannot iterate N_Π^{hex}. The notion of computation (Marek et al., 2004) is one possible way of defining iterations of N_Π^{hex}.

[4] in the sense of (Marek et al., 2004)

In Section 6.3 we define approximations (Pelov, 2004; Pelov and Truszczyński, 2004) of N_Π^{hex}. Particularly, we define the (i) non-deterministic Fitting approximation $\mathcal{F}_\Pi^{hex}$ based on the extended Fitting operator Φ_Π^{hex}, and the (ii) non-deterministic ultimate approximation $\mathcal{N}_\Pi^{hex}$ based on the ultimate approximation $\mathcal{T}_\Pi^{hex}$. We then show that $\mathcal{F}_\Pi^{hex}$ and $\mathcal{N}_\Pi^{hex}$ similarly relate to N_Π^{hex} as Φ_Π^{hex} and $\mathcal{T}_\Pi^{hex}$ relate to T_Π^{hex} according to Approximation Theory; moreover, as in the non-disjunctive case, the ultimate approximation $\mathcal{N}_\Pi^{hex}$ yields, in general, more answer sets than $\mathcal{F}_\Pi^{hex}$.

In Section 6.4 we define 2-valued answer-set semantics in terms of minimal fixpoints of the non-deterministic Fitting approximation $\mathcal{F}_\Pi^{hex}$ and show that these semantics extend the classical answer-set semantics of disjunctive logic programs (Gelfond and Lifschitz, 1991). Moreover, we show that each answer set is derivable by a bottom-up computation of $\mathcal{F}_\Pi^{hex}$. However, since we approximate only the domain of N_Π^{hex}, we do not obtain 3-valued answer-set semantics and, consequently, do not obtain well-founded semantics.[5]

Finally, in Section 6.5 we define 2-valued ultimate answer-set semantics similar in terms of minimal fixpoints of $\mathcal{N}_\Pi^{hex}$, and show that ultimate answer-set semantics are, in general, "weaker" than $\mathcal{F}_\Pi^{hex}$-answer-set semantics (i.e., every $\mathcal{F}_\Pi^{hex}$-answer set is also an ultimate answer set).

6.1 Non-Deterministic Operator

In Section 4.1 we extended the well-known van Emden-Kowalski operator (Van Emden and Kowalski, 1976) of classical logic programs to the class of (disjunctive) HEX programs by (see Definition 4.1.1)

$$T_\Pi^{hex}(I) = \{H(r) : r \in \Pi : I \models B(r)\}.$$

For every normal HEX program Π, T_Π^{hex} is a lattice operator mapping each interpretation $I \in \mathcal{I}_\Pi$ to some interpretation $T_\Pi^{hex}(I) \in \mathcal{I}_\Pi$, where $\mathcal{I}_\Pi$ is the complete lattice of all interpretations of Π ordered by set inclusion.

However, if Π is disjunctive, $T_\Pi^{hex}(I)$ may contain disjunctions. For instance, consider the program $P = \{a \vee b \leftarrow\}$ and observe $T_\Pi^{hex}(\emptyset) = \{a \vee b\} \notin \mathcal{I}_\Pi$. Hence, for a disjunctive HEX programs Π, T_Π^{hex} is, in general, no longer a lattice operator. Consequently, we cannot apply the machinery of Approximation Theory (Denecker et al., 2000a, 2004) to T_Π^{hex}.

In this section we define an immediate consequence operator N_Π^{hex} which handles disjunctions as follows: N_Π^{hex} maps each interpretation $I \in \mathcal{I}_\Pi$ to a collection of interpretations (called *coin* (Seipel et al., 1997) henceforth) $\mathcal{J} \in \mathfrak{P}(\mathcal{I}_\Pi)$ such that every $J \in \mathcal{J}$ represents a possible outcome of applying one step of logical derivation to Π with respect to I.

To this end, we first define the *Smyth ordering* $\sqsubseteq$ (Smyth, 1978), for every coin $\mathcal{J}, \mathcal{K} \in \mathfrak{P}(\mathcal{I}_\Pi)$, by

$$\mathcal{J} \sqsubseteq \mathcal{K} \Leftrightarrow \text{for every } K \in \mathcal{K} \text{ there exists some } J \in \mathcal{J} \text{ such that } J \subseteq K.$$

[5] Disjunctive well-founded semantics is controversial and various proposals exist with some late on (Wang and Zhou, 2003) being considered best so far.

Intuitively, if we interpret $\mathcal{J}$ and $\mathcal{K}$ as possible outcomes of Π (i.e., $N_\Pi^{hex}(J) = \mathcal{J}$ and $N_\Pi^{hex}(K) = \mathcal{K}$ for some $J, K \in \mathcal{I}_\Pi$) and $\mathcal{J} \sqsubseteq \mathcal{K}$, then every $K' \in \mathcal{K}$ is a refinement of some $J' \in \mathcal{J}$, that is, $J' \subseteq K'$.

The ordering $\sqsubseteq$ is reflexive and transitive, but not anti-symmetric. Thus, $\mathfrak{P}(\mathcal{I}_\Pi)$ endowed with $\sqsubseteq$ is not a poset. However, if we consider only the minimal sets in $\mathfrak{P}(\mathcal{I}_\Pi)$, denoted $\mathfrak{P}_{min}(\mathcal{I}_\Pi)$, then $(\mathfrak{P}_{min}(\mathcal{I}_\Pi), \sqsubseteq)$ is a complete poset (Fernández and Minker, 1995) with least element $\{\emptyset\}$.

Example 6.1.1. Given the following two sets of minimal interpretations we have

$$\{\{a\}, \{b\}\} \sqsubseteq \{\{a\}\} \quad \text{and} \quad \{\{a\}\} \not\sqsubseteq \{\{a\}, \{b\}\}.$$

However, if we consider coins with nonminimal interpretations

$$\{\{a\}, \{a, b\}\} \sqsubseteq \{\{a\}\} \quad \text{and} \quad \{\{a\}\} \sqsubseteq \{\{a, b\}, \{a\}\},$$

but $\{\{a\}\} \neq \{\{a\}, \{a, b\}\}$ shows that $\sqsubseteq$ may be not anti-symmetric.

In the sequel let Π be a disjunctive HEX program. Let DHB_Π be the *disjunctive Herbrand base* (Minker and Rajasekar, 1990; Lobo et al., 1992) of Π consisting of all disjunctions of form

$$a_1 \vee \ldots \vee a_k, \quad k \geq 1, \tag{6.2}$$

which can be formed with atoms from Λ_Π. Given a disjunction d of form (6.2), we interpret d as the fact d' defined by

$$a_1 \vee \ldots \vee a_k \leftarrow$$

and say that an interpretation I is a *model* of d if $I \models d'$.

The next result follows directly from definitions.

Proposition 6.1.2. *Let Π be a HEX program, and let $I \in \mathcal{I}_\Pi$. Then, I is a model of Π iff I is a model of $T_\Pi^{hex}(I)$.*

We now define the non-deterministic (or disjunctive) pendant to the van Emden-Kowalski operator T_Π^{hex}.

Definition 6.1.3 (Non-deterministic van Emden-Kowalski operator)**.** Let Π be a disjunctive HEX program, and let $D \in \mathfrak{P}(DHB_\Pi)$. By the *non-deterministic van Emden-Kowalski operator* of Π we mean the operator $N_\Pi^{hex} : \mathcal{I}_\Pi \longrightarrow \mathfrak{P}_{min}(\mathcal{I}_\Pi)$ defined by

$$N_\Pi^{hex}(I) = MM\left(I \cup T_\Pi^{hex}(I)\right), \tag{6.3}$$

where $MM(D)$ is the set of all minimal models of D, and $\mathfrak{P}(\mathcal{I}_\Pi)$ is ordered by $\sqsubseteq$.

Intuitively, $N_\Pi^{hex}(I) = \mathcal{J}$ consists of all interpretations $J \in \mathcal{J}$ representing minimal possible outcomes (i.e., models) of Π after one step of logical derivation. Notice that for every $J \in N_\Pi^{hex}(I)$ we have $I \subseteq J$, that is, when applying N_Π^{hex} to I we assume each $a \in I$ to be *true*.

Example 6.1.4. Consider the classical disjunctive program Π consisting of the following single fact:

$$a \vee b \leftarrow .$$

Then, $N_\Pi^{hex}(\{\emptyset\}) = MM(\{a \vee b\}) = \{\{a\}, \{b\}\}$ (i.e., either a is *true* and b is *false* or vice versa) represents the possible outcomes of Π with respect to $\emptyset$ (i.e., a and b are *false*). Furthermore, we have

$$N_\Pi^{hex}(\{a\}) = \{\{a\}\} \quad \text{and} \quad N_\Pi^{hex}(\{b\}) = \{\{b\}\},$$

that is, if we consider a (resp., b) as *true* and b (resp., a) as *false*, then applying N_Π^{hex} to $\{a\}$ (resp., $\{b\}$) and $a \vee b \leftarrow$ correctly entails that $\{a\}$ (resp., $\{b\}$) is the only outcome compatible with the assumption.

Let $I \in \mathcal{I}_\Pi$. We say that I is a *fixpoint* of N_Π^{hex} if $I \in N_\Pi^{hex}(I)$. We denote the set of all minimal fixpoints of N_Π^{hex} by $\mathrm{mfp}(N_\Pi^{hex})$. Since every $J \in N_\Pi^{hex}(I)$ has to contain I as a subset, we immediately conclude the following characterization of fixpoints of N_Π^{hex}.

Proposition 6.1.5. *Let Π be a HEX program, and let $I \in \mathcal{I}_\Pi$. Then, I is a fixpoint of N_Π^{hex} iff $N_\Pi^{hex}(I) = \{I\}$.*

Moreover, Proposition 6.1.5 implies the following weaker characterization of fixpoints of N_Π^{hex}. We say that I is a *pre-fixpoint* of N_Π^{hex} if $N_\Pi^{hex}(I) \sqsubseteq \{I\}$. Then we can show the following result.

Proposition 6.1.6. *Let Π be a HEX program, and let $I \in \mathcal{I}_\Pi$. Then, I is a fixpoint of N_Π^{hex} iff I is a pre-fixpoint of N_Π^{hex}.*

Proof. We only have to show that every pre-fixpoint is indeed a fixpoint of N_Π^{hex}. Let $I \in \mathcal{I}_\Pi$ and assume $N_\Pi^{hex}(I) \sqsubseteq \{I\}$, that is, there exists some $J \in N_\Pi^{hex}(I)$ such that $J \subseteq I$. Since $I \subseteq J$ by definition of N_Π^{hex}, we have $J = I$. Consequently, $I \in N_\Pi^{hex}(I)$. □

In Proposition 4.1.3 we have seen that if Π is disjunction-free, then the pre-fixpoints of T_Π^{hex} characterize models of Π. The next result shows a similar characterization of models of disjunctive HEX programs.

Proposition 6.1.7. *Let Π be a HEX program, and let $I \in \mathcal{I}_\Pi$. Then, I is a model of Π iff I is a pre-fixpoint of N_Π^{hex}.*

Proof. ($\Rightarrow$) By Proposition 6.1.2, I is a model of Π iff I is a model of $T_\Pi^{hex}(I)$. Consequently, if I is a model of Π, I is also a model of $I \cup T_\Pi^{hex}(I)$ and, clearly, a minimal such model. Hence, $I \in MM(I \cup T_\Pi^{hex}(I)) = N_\Pi^{hex}(I)$.

($\Leftarrow$) Assume $I \in N_\Pi^{hex}(I)$ and I is not a model of Π. Then, there exists a rule $r \in \Pi$ such that $I \models B(r)$, but $I \cap H(r) = \emptyset$. Since $I \models B(r)$, we have $H(r) \in T_\Pi^{hex}(I)$. Hence, for every $J \in MM(I \cup T_\Pi^{hex}(I)) = N_\Pi^{hex}(I)$ we have $I \subsetneq J$, a contradiction. □

As an immediate consequence of Propositions 6.1.6 and 6.1.7 we obtain the following characterization of minimal models of Π.

Corollary 6.1.8. *Let* Π *be a HEX program, and let* $I \in \mathcal{I}_\Pi$*. Then,* I *is a minimal model of* Π *iff* $I \in \mathrm{mfp}(N^{hex}_\Pi)$.

Example 6.1.9. Consider the classical disjunctive program Π consisting of the following propositional rules:

$$a \vee b \leftarrow$$
$$b \vee c \leftarrow$$
$$a \vee c \leftarrow .$$

Then, $\{a, b\}$, $\{a, c\}$ and $\{b, c\}$ are the FLP-answer sets (and therefore minimal models) of Π. Since the FLP-answer sets are not mutually disjoint, this example shows that disjunctions under FLP-answer set semantics are, in general, not interpreted mutually exclusive.

Likewise, applying N^{hex}_Π to the empty set yields

$$N^{hex}_\Pi(\emptyset) = \{\{a, b\}, \{a, c\}, \{b, c\}\},$$

and it is easy to verify that each $J \in N^{hex}_\Pi(\emptyset)$ is a (pre-)fixpoint of N^{hex}_Π. For instance, we have

$$N^{hex}_\Pi(\{a, b\}) = MM\left(\{a, b\} \cup T^{hex}_\Pi(\{a, b\})\right) = MM\left(\{a, b, a \vee b, b \vee c, a \vee c\}\right) = \{\{a, b\}\}.$$

6.2 Iterating the Non-Deterministic Operator by Computations

The results of Section 6.1 show the similarities between the non-deterministic van Emden-Kowalski operator N^{hex}_Π and its deterministic counterpart T^{hex}_Π. Therefore, we can consider N^{hex}_Π as an "extension" of T^{hex}_Π to the class of disjunctive HEX programs. However, an important property of T^{hex}_Π which N^{hex}_Π does not satisfy is that T^{hex}_Π is *iterable* in the sense that, for each interpretation I, $T^{hex}_\Pi(T^{hex}_\Pi(I))$ is well-defined. In contrast, since the domain and range of N^{hex}_Π do not coincide, it is not obvious how to "iterate" N^{hex}_Π.

Pelov and Truszczyński (2004) proposed the following notion of computation (Marek et al., 2004) as an appropriate formalization of this process.

Definition 6.2.1 (Computation)**.** Let Π be a disjunctive HEX program. By a *computation* (Marek et al., 2004) (with respect to N^{hex}_Π) we mean a sequence $I^\uparrow = (I_i)_{i \geq 0} \in \mathcal{I}^{\mathbb{N}}_\Pi$ where $I_0 = \emptyset$ and, for every $n \geq 0$,

1. $I_n \subseteq I_{n+1}$, and
2. $I_{n+1} \in N^{hex}_\Pi(I_n)$.

We define the *result* of $I^\uparrow$ by $I^\infty = \bigcup_{i \geq 0} I_i$ and say that an interpretation $I \in \mathcal{I}_\Pi$ is *derivable* if there exists a computation $I^\uparrow$ with result $I^\infty = I$.

Remark 6.2.2. In this thesis we consider only *finite* (i.e., *terminating*) computations.

Intuitively, every computation $I^\uparrow$ defines a non-deterministic *bottom-up* construction of its result I^∞. The next result shows that each derivable interpretation I is a fixpoint and, hence, a model of Π.

Proposition 6.2.3. *Let Π be a HEX program and I^∞ be the result of some computation $I^\uparrow$. Then, I^∞ is a fixpoint of N_Π^{hex}.*

Proof. Since Π is finite, by definition of I^∞, $I^\infty = I_k$ for some $k \geq 0$ and $I_{k+1} = I_k \in N_\Pi^{hex}(I_k)$ (see Remark 6.2.2). □

Corollary 6.2.4. *Let Π be a HEX program, and let I^∞ be the result of some computation $I^\uparrow$. Then, I^∞ is a model of Π.*

Proof. Is an immediate consequence of Proposition 6.2.3, Proposition 6.1.7 and Proposition 6.1.6. □

Example 6.2.5. Let Π be the classical program of Example 6.1.9. Then, $I^\uparrow = (I_i)_{i\geq 0}$ with $I_0 = \emptyset$ and $I_k = \{a, b\}$, $k \geq 1$, is a computation with result $I^\infty = \{a, b\}$. Moreover, it is easy to verify that each FLP-answer set of Π is derivable.

The next example shows that a derivable model I may have more than one computation and that, in general, there exists no canonical computation for I.

Example 6.2.6. Let Π be the classical disjunctive program consisting of the following propositional rules:

$$\begin{aligned} a \vee b \leftarrow \\ a \leftarrow b \\ b \leftarrow a. \end{aligned}$$

Then we can define the following two distinct computations $(I_i)_{i\geq 0}$ and $(J_i)_{i\geq 0}$ with results $I^\infty = J^\infty = \{a, b\}$. Let $I_0 = J_0 = \emptyset$. First, define $I_1 = \{a\}$, $J_1 = \{b\}$ and notice $I_1, J_1 \in N_\Pi^{hex}(\emptyset) = \{\{a\}, \{b\}\}$. Second, define $I_2 = J_2 = \{a, b\}$ and observe $N_\Pi^{hex}(I_1) = N_\Pi^{hex}(J_1) = \{\{a, b\}\}$. Finally, define $I_k = J_k = \{a, b\}$ for every $k \geq 3$. It follows that $I^\uparrow = (I_i)_{i\geq 0}$ and $J^\uparrow = (J_i)_{i\geq 0}$ are computations and $I^\infty = J^\infty = \{a, b\}$ as desired.

Let us analyze Example 6.2.6 in more detail. The first rule is a fact stating that a or b is *true*. To conclude from this fact and the two other rules that a and b both have to be *true*, informally, one has to make the following case distinction: (i) if a is *true*, then b is *true* (last rule), and (ii) if b is *true*, then a is *true* (second rule). That is, intuitively, $N_\Pi^{hex}(\emptyset) = \{\{a\}, \{b\}\}$ is a coin consisting of sets of *assumptions* (not of *true* atoms). Hence, when we apply in the second iteration N_Π^{hex}, e.g., to $I_1 = \{a\}$, we intuitively assume that a is *true* and derive the outcome $\{a, b\}$ containing a. That is, we have to "remember" the chosen assumptions by adding the assumed atoms to every outcome obtained after applying N_Π^{hex}. This intuition is the motivation of the definition of N_Π^{hex} given in (6.3).

We say that N_Π^{hex} is *monotone* if for every $J, I \in \mathcal{I}_\Pi$ such that $J \subseteq I$, $N_\Pi^{hex}(J) \sqsubseteq N_\Pi^{hex}(I)$. The following example shows that if Π contains negation as failure or nonmonotone external atoms, N_Π^{hex} may be nonmonotone.

Example 6.2.7. Let Π be the classical normal program consisting of the following single rule:

$$a \leftarrow\sim b.$$

Then, $N^{hex}_{\Pi}(\emptyset) = \{\{a\}\}$ and $N^{hex}_{\Pi}(\{b\}) = \{\{b\}\}$, but $\{\{a\}\} \not\sqsubseteq \{\{b\}\}$. Furthermore, notice that for the positive HEX program $\Pi^{\#} = \left\{a \leftarrow (not\ b)^{\#}\right\}$, where $I \models (not\ b)^{\#}$ iff $b \notin I$, we have $N^{hex}_{\Pi} = N^{hex}_{P^{\#}}$.

However, the next result shows that if Π is positive and monotone (i.e., does not contain negation as failure and each external atom is monotone, see Section 5.1), N^{hex}_{Π} is monotone. Before we prove the proposition we state the following lemma.

Lemma 6.2.8. *Let Π be a HEX program. For every $D, E \in DHB_{\Pi}$ such that $D \subseteq E$, $MM(D) \sqsubseteq MM(E)$.*

Proposition 6.2.9. *Let Π be a positive monotone HEX program. Then, N^{hex}_{Π} is monotone.*

Proof. Since Π is positive, T^{hex}_{Π} is monotone. Hence, for each $J, I \in \mathcal{I}_{\Pi}$ such that $J \subseteq I$, $J \cup T^{hex}_{\Pi}(J) \subseteq I \cup T^{hex}_{\Pi}(I)$ and therefore, by Lemma 6.2.8,

$$MM\left(J \cup T^{hex}_{\Pi}(J)\right) \sqsubseteq MM\left(I \cup T^{hex}_{\Pi}(I)\right)$$

which directly entails $N^{hex}_{\Pi}(J) \sqsubseteq N^{hex}_{\Pi}(I)$. □

The following theorem shows that there exists for every minimal model I of Π a computation $I^{\uparrow}$ with result $I^{\infty} = I$.

Theorem 6.2.10. *Let Π be a positive monotone HEX program. Then, every minimal model I of Π is derivable.*

Proof. By Corollary 6.1.8, I is a minimal model of Π iff $I \in \mathrm{mfp}(N^{hex}_{\Pi})$. We define a computation $I^{\uparrow}$ with result $I^{\infty} \subseteq I$ and show $I^{\infty} = I$. Let $I_0 = \emptyset$. Since I is a fixpoint of N^{hex}_{Π}, we have $N^{hex}_{\Pi}(I) = \{I\}$ by Proposition 6.1.5. Furthermore, since N^{hex}_{Π} is monotone and $I_0 \subseteq I$, we have

$$N^{hex}_{\Pi}(I_0) \sqsubseteq N^{hex}_{\Pi}(I) = \{I\},$$

that is, there exists some $I_1 \in N^{hex}_{\Pi}(I_0)$ such that $I_0 \subseteq I_1 \subseteq I$. Generally, we obtain, for every $n \geq 0$, $I_{n+1} \in N^{hex}_{\Pi}(I_n)$ for some $I_n \subseteq I_{n+1} \subseteq I$. Clearly, this construction yields a computation $I^{\uparrow}$ with result $I^{\infty} \subseteq I$. By Proposition 6.2.3, I^{∞} is a fixpoint of N^{hex}_{Π}. Consequently, since I is a minimal fixpoint of N^{hex}_{Π}, $I^{\infty} = I$. Hence, I is derivable. □

Corollary 6.2.11. *Let Π be a positive monotone HEX program, and let $I \in \mathcal{I}_{\Pi}$. If $I \in \mathrm{mfp}(N^{hex}_{\Pi})$, then I is derivable.*

Proof. Is an immediate consequence of Theorem 6.2.10 and Corollary 6.1.8. □

Arguably, Theorem 6.2.10 shows that the notion of computation adequately formalizes the intuitive meaning of "iterating" N^{hex}_{Π}. However, if N^{hex}_{Π} is not positive or contains nonmonotone external atoms, the notion of computation of N^{hex}_{Π} is no longer well-defined.

In the next section we define approximations of N^{hex}_{Π} similar to approximations of T^{hex}_{Π} in the case of normal HEX programs (see Chapter 4).

6.3 Non-Deterministic Approximations and Computations

Let us briefly recall some basic definitions and intuitions of Approximation Theory combined with normal HEX programs (see Chapter 4).

Given a normal HEX program Π, the extended van Emden-Kowalski operator $T_\Pi^{hex} : \mathcal{I}_\Pi \longrightarrow \mathcal{I}_\Pi$ characterizes (supported) models of Π and is, in general, nonmonotone. An approximation of T_Π^{hex} is any operator $\mathcal{A}_\Pi^{hex} : \mathcal{I}_\Pi^c \longrightarrow \mathcal{I}_\Pi^c$ which is *monotone* with respect to the precision ordering $\subseteq_p$ and coincides with T_Π^{hex} on $\mathcal{I}_\Pi$.

Moreover, recall from Definition 4.1.8 that the first argument of the extended Fitting approximation Φ_Π^{hex} is defined, for each $(I_1, I_2) \in \mathcal{I}_\Pi^c$, by

$$\Phi_\Pi^{hex}(I_1, I_2)_1 = \left\{ H(r) : r \in \Pi : \langle\, B(r) \,\rangle_{(I_1,I_2)} = \mathbf{t} \right\},$$

that is, $\Phi_\Pi^{hex}(I_1, I_2)_1$ contains every disjunction $H(r) \in DHB_\Pi$ such that the body of r is *true* with respect to the 3-valued interpretation (I_1, I_2). Furthermore, the first argument of the ultimate approximation $\mathcal{T}_\Pi^{hex}$ is defined (see Definition 4.4.1), for each $(I_1, I_2) \in \mathcal{I}_\Pi^c$, by

$$\mathcal{T}_\Pi^{hex}(I_1, I_2)_1 = \bigcap_{I_1 \subseteq J \subseteq I_2} T_\Pi^{hex}(J).$$

Analogously, in this section we define approximations of the non-deterministic immediate consequence operator N_Π^{hex} (Pelov and Truszczyński, 2004; Pelov, 2004). Since N_Π^{hex} is not a lattice operator, Approximation Theory is not directly applicable. However, we can define the following non-deterministic approximations of N_Π^{hex}.

Definition 6.3.1 (Non-deterministic approximations)**.** Let Π be a disjunctive HEX program. Define, for each $(I_1, I_2) \in \mathcal{I}_\Pi^c$, (i) the *non-deterministic Fitting approximation* of N_Π^{hex} by

$$\mathcal{F}_\Pi^{hex}(I_1, I_2) = MM\left(I_1 \cup \Phi_\Pi^{hex}(I_1, I_2)_1\right), \tag{6.4}$$

(ii) and the *non-deterministic ultimate approximation* of N_Π^{hex} by

$$\mathcal{N}_\Pi^{hex}(I_1, I_2) = MM\left(I_1 \cup \mathcal{T}_\Pi^{hex}(I_1, I_2)_1\right). \tag{6.5}$$

The next result shows that $\mathcal{F}_\Pi^{hex}$ and $\mathcal{N}_\Pi^{hex}$ similarly relate to N_Π^{hex} as Φ_Π^{hex} and $\mathcal{T}_\Pi^{hex}$ relate to T_Π^{hex}.

Proposition 6.3.2. *Let Π be a HEX program, $I \in \mathcal{I}_\Pi$, $(J_1, J_2), (I_1, I_2) \in \mathcal{I}_\Pi^c$, and $\mathcal{A}_\Pi^{hex} \in \left\{\mathcal{F}_\Pi^{hex}, \mathcal{N}_\Pi^{hex}\right\}$. Then,*

1. *$\mathcal{A}_\Pi^{hex}(I, I) = N_\Pi^{hex}(I)$, and*
2. *$(J_1, J_2) \subseteq_p (I_1, I_2)$ implies $\mathcal{A}_\Pi^{hex}(J_1, J_2) \sqsubseteq \mathcal{A}_\Pi^{hex}(I_1, I_2)$.*

Proof. (1) Since $\mathcal{T}_\Pi^{hex}(I, I) = \Phi_\Pi^{hex}(I, I) = T_\Pi^{hex}(I)$, the equality holds. (2) Follows directly from the facts that Φ_Π^{hex} and $\mathcal{T}_\Pi^{hex}$ are both monotone with respect to $\subseteq_p$, and MM is monotone (see Lemma 6.2.8). □

In the sequel let $\mathcal{A}_\Pi^{hex} \in \{\mathcal{F}_\Pi^{hex}, \mathcal{N}_\Pi^{hex}\}$. In Section 6.2 we defined the notion of computation for N_Π^{hex} and showed its adequacy in the case that N_Π^{hex} is monotone. Proposition 6.3.2 shows that $\mathcal{A}_\Pi^{hex}$ is monotone which implies that we can reasonably adapt the notion of computation to approximations $\mathcal{F}_\Pi^{hex}$ and $\mathcal{N}_\Pi^{hex}$ of N_Π^{hex}. Therefore, we now define computations of $\mathcal{A}_\Pi^{hex}$ with respect to some interpretation I.

Definition 6.3.3 (Computation)**.** Let Π be a disjunctive HEX program,, and let $I \in \mathcal{I}_\Pi$. By an $\mathcal{A}_\Pi^{hex}$*-*I*-computation* (Marek et al., 2004) we mean a sequence $J^{I,\uparrow} = (J_i)_{i\geq 0} \in \mathcal{I}_\Pi^{\mathbb{N}}$ such that $J_0 = \emptyset$ and, for every $n \geq 0$,

1. $J_n \subseteq J_{n+1} \subseteq I$, and
2. $J_{n+1} \in \mathcal{A}_\Pi^{hex}(J_n, I)$.

We call $J^{I,\infty} = \bigcup_{i\geq 0} J_i$ the *result* of computation $J^{I,\uparrow}$ and say that an interpretation $J \in \mathcal{I}_\Pi$, $J \subseteq I$, is $\mathcal{A}_\Pi^{hex}$*-*I*-derivable* if there exists a computation $J^{I,\uparrow}$ with result $J^{I,\infty} = J$.

Condition (1) states that every J_n has to be a subset of I, i.e., we consider only those outcomes compatible with the assumption that every atom in I is not *false*.

Example 6.3.4. Let Π be the classical disjunctive program consisting of the following propositional rules:

$$a \vee b \leftarrow$$
$$c \leftarrow a, \sim b.$$

We show that $I = \{a, c\}$ is $\mathcal{F}_\Pi^{hex}$-I-derivable. Let $I_0 = \emptyset$. We compute

$$\mathcal{F}_\Pi^{hex}(I_0, I) = MM\left(\Phi_\Pi^{hex}(I_0, I)_1\right) = MM(\{a \vee b\}) = \{\{a\}, \{b\}\},$$

and define $I_1 = \{a\} \in \mathcal{F}_\Pi^{hex}(I_0, I)$. For the next iteration, we compute

$$\mathcal{F}_\Pi^{hex}(I_1, I) = MM\left(I_1 \cup \Phi_\Pi^{hex}(I_1, I)_1\right) = MM(\{a, a \vee b, c\}) = \{I\},$$

and define $I_2 = I$. Finally, since we have

$$\mathcal{F}_\Pi^{hex}(I_2, I) = \mathcal{F}_\Pi^{hex}(I, I) = N_\Pi^{hex}(I) = \{I\},$$

we define $I_n = I$ for every $n \geq 2$, and obtain a $\mathcal{F}_\Pi^{hex}$-I-computation $J^{I,\uparrow}$ with result $J^{I,\infty} = I$. Hence, I is $\mathcal{F}_\Pi^{hex}$-I-derivable. Similar calculations show that $J^{I,\uparrow}$ is also a $\mathcal{N}_\Pi^{hex}$-I-computation.

6.4 Answer-Set Semantics

In this section we translate the definitions of Section 4.2 to the case of disjunctive HEX programs, and the corresponding non-deterministic operators as follows. Let Π be a disjunctive HEX program.

1. Instead of considering T_Π^{hex}, we consider the non-deterministic van Emden-Kowalski operator N_Π^{hex} as an appropriate operator of describing one-step of logical derivation.
2. Instead of iterating $\Phi_\Pi^{hex}(\,.\,, I)_1$ to the least fixpoint $\Phi_\Pi^{hex,\downarrow}(I)$, we "iterate" $\mathcal{F}_\Pi^{hex}(\,.\,, I)$ in terms of $\mathcal{F}_\Pi^{hex}$-I-computations to the *minimal* fixpoints $\mathcal{F}_\Pi^{hex,\downarrow}(I)$.
3. Since disjunctive rules are non-deterministic, we consider minimal models instead of least models, and minimal fixpoints instead of least fixpoints.

With these intuitions in mind, we now define (2-valued) answer-set semantics.

Definition 6.4.1 (Answer-set semantics)**.** Let Π be a disjunctive HEX program,, and let $I \in \mathcal{I}_\Pi$. We say that I is an $\mathcal{F}_\Pi^{hex}$*-answer set* if $I \in \mathcal{F}_\Pi^{hex,\downarrow}(I)$ where

$$\mathcal{F}_\Pi^{hex,\downarrow}(I) = \mathrm{mfp}\left(\mathcal{F}_\Pi^{hex}(\,.\,, I)\right).$$

Example 6.4.2. Let Π be the disjunctive HEX program consisting of the following rules:

$$p(a) \vee q(a) \leftarrow$$
$$\bot \leftarrow\sim p \subseteq^{\#} q, \sim \bot,$$

where, for each $I \in \mathcal{I}_\Pi$, $I \models p \subseteq^{\#} q$ iff $p^I \subseteq q^I$ (see Example 2.2.2). Intuitively, the first rule states that $a \in p^I$ or $a \in q^I$, whereas the second rule states that p^I has to be a subset of q^I.[6] We show that $I = \{q(a)\}$ is the only $\mathcal{F}_\Pi^{hex}$-answer set. First, we compute $\mathcal{F}_\Pi^{hex}(\emptyset, I) = \{\{p(a)\}, I\}$ which shows that $\emptyset$ is not a fixpoint of $\mathcal{F}_\Pi^{hex}(\,.\,, I)$ (i.e., $\emptyset \notin \mathcal{F}_\Pi^{hex}(\emptyset, I)$). Second, we compute $\mathcal{F}_\Pi^{hex}(I, I) = N_\Pi^{hex}(I) = \{I\}$ and conclude $I \in \mathcal{F}_\Pi^{hex,\downarrow}(I)$, that is, I is an $\mathcal{F}_\Pi^{hex}$-answer set.

We now show that $I' = \{p(a)\}$ is not an $\mathcal{F}_\Pi^{hex}$-answer set. Since $I' \not\models p \subseteq^{\#} q$ and $I' \not\models \bot$, the second rule "fires", that is, we obtain $N_\Pi^{hex}(I') = \{\{p(a), \bot\}\}$. Therefore, we conclude that I' is not a fixpoint of $\mathcal{F}_\Pi^{hex}(\,.\,, I')$ and, hence, not an $\mathcal{F}_\Pi^{hex}$-answer set. Finally, since $\mathcal{F}_\Pi^{hex}$-answer sets are minimal (see Theorem 6.4.8), I is the only $\mathcal{F}_\Pi^{hex}$-answer set.

Example 6.4.3. Consider the disjunctive HEX program consisting of the following rules:

$$q(a) \vee q(b) \leftarrow$$
$$q(a) \leftarrow (\exists x\ q(x))^{\#},$$

where $(\exists x\ q(x))^{\#}$ is defined as in Example 4.2.9. We show that $I = \{q(a)\}$ is an $\mathcal{F}_\Pi^{hex}$-answer set. Since $\mathcal{F}_\Pi^{hex}(\emptyset, I) = \{\{q(a)\}, \{q(b)\}\}$, the empty set is not a fixpoint of $\mathcal{F}_\Pi^{hex}(\,.\,, I)$. In contrast, we have

$$\mathcal{F}_\Pi^{hex}(I, I) = MM(\{q(a) \vee q(b), q(a)\}) = \{\{q(a)\}\},$$

which shows that I is indeed a minimal fixpoint of $\mathcal{F}_\Pi^{hex}$ and, hence, an $\mathcal{F}_\Pi^{hex}$-answer set. Finally, it is easy to verify that $\{q(b)\}$ and $\{q(a), q(b)\}$ are not $\mathcal{F}_\Pi^{hex}$-answer sets.

[6] Notice that the second rule is a constraint (see Example 2.1.2).

In analogy to Proposition 4.1.3 we have the following result.

Proposition 6.4.4. *Let Π be a HEX program, and let $I \in \mathcal{I}_\Pi$. If I is an $\mathcal{F}_\Pi^{hex}$-answer set of Π, then I is a fixpoint of N_Π^{hex}.*

Proof. Follows directly from Proposition 6.3.2. □

We now show that the $\mathcal{F}_\Pi^{hex}$-answer-set semantics indeed generalize the answer-set semantics (Gelfond and Lifschitz, 1991) of classical disjunctive programs. To this end, we first prove the following result (see Example 3.3.7).

Lemma 6.4.5. *Let Π be a classical disjunctive program, and let $I \in \mathcal{I}_\Pi$. Then, $\mathcal{F}_\Pi^{hex}(\,.\,, I) = N_{\Pi^I}^{hex}$.*

Proof. Observe that (5.2) also holds for disjunctive programs. □

Theorem 6.4.6. *Let Π be a classical disjunctive program, and let $I \in \mathcal{I}_\Pi$. Then, I is an answer set of Π (in the classical sense (Gelfond and Lifschitz, 1991)) iff $I \in \mathcal{F}_\Pi^{hex,\downarrow}(I)$.*

Proof. We have the following equivalences: I is an answer set of Π iff I is a minimal model of Π^I iff $I \in \mathrm{mfp}\left(N_{\Pi^I}^{hex}\right)$ (Corollary 6.1.8) iff $I \in \mathrm{mfp}\left(\mathcal{F}_\Pi^{hex}(\,.\,, I)\right)$ (Lemma 6.4.5) iff $I \in \mathcal{F}_\Pi^{hex,\downarrow}(I)$ (see proof of Proposition 4.37 in Pelov (2004)). □

The next result states that results of $\mathcal{A}_\Pi^{hex}$-I-computations are fixpoints of $\mathcal{A}_\Pi^{hex}(\,.\,, I)$.

Proposition 6.4.7. *Let Π be a HEX program, $I \in \mathcal{I}_\Pi$, and $J^{I,\infty}$ be the result of some $\mathcal{A}_\Pi^{hex}$-I-computation $J^{I,\uparrow}$. Then, $J^{I,\infty}$ is a fixpoint of $\mathcal{A}_\Pi^{hex}(\,.\,, I)$.*

Proof. Similar to the proof of Proposition 6.2.3. □

At this point, it is not clear whether $\mathcal{F}_\Pi^{hex}$-answer sets are minimal, which is a key requirement when defining semantics of logic programs. The next theorem shows that $\mathcal{F}_\Pi^{hex}$-answer sets are indeed minimal models.

Theorem 6.4.8. *Let Π be a HEX program, and let $I \in \mathcal{I}_\Pi$. If I is an $\mathcal{F}_\Pi^{hex}$-answer set of Π, then I is a minimal model of Π.*

Proof. Similar to the proof of Theorem 4.33 in (Pelov, 2004). □

6.5 Ultimate Answer-Set Semantics

In this section we define ultimate answer-set semantics of disjunctive HEX programs. In the sequel let Π be a disjunctive HEX program. Recall from Definition 6.3.1 that the non-deterministic ultimate approximation of N_Π^{hex} is defined by

$$\mathcal{N}_\Pi^{hex}(I_1, I_2) = MM\left(I_1 \cup \mathcal{T}_\Pi^{hex}(I_1, I_2)_1\right),$$

where $\mathcal{T}_\Pi^{hex}$ is the ultimate approximation of the extended van Emden-Kowalski operator T_Π^{hex} (see Definition 4.1.1).

Definition 6.5.1 (Ultimate answer-set semantics)**.** Let Π be a disjunctive HEX program. We say that an interpretation $I \in \mathcal{I}_\Pi$ is an *ultimate answer set* of Π if

$$I \in \mathcal{N}_\Pi^{hex,\downarrow}(I) = \mathrm{mfp}\left(\mathcal{N}_\Pi^{hex}(\,.\,, I)\right).$$

Since the results of Chapter 4 are only valid in the case that T_Π^{hex} is a lattice operator (i.e., Π is normal), we have to explicitly state the next result.

Lemma 6.5.2. *Let Π be a HEX program, and let $(I_1, I_2) \in \mathcal{I}_\Pi^c$. Then, $\Phi_\Pi^{hex}(I_1, I_2)_1 \subseteq \mathcal{T}_\Pi^{hex}(I_1, I_2)_1$.*

The next result shows that the non-deterministic ultimate approximation $\mathcal{N}_\Pi^{hex}$ is "more precise" than the non-deterministic Fitting approximation $\mathcal{F}_\Pi^{hex}$.

Proposition 6.5.3. *Let Π be a HEX program. Then, for each $(I_1, I_2) \in \mathcal{I}_\Pi^c$,*

$$\mathcal{F}_\Pi^{hex}(I_1, I_2) \sqsubseteq \mathcal{N}_\Pi^{hex}(I_1, I_2).$$

Proof. Follows directly from the facts that MM is monotone (Lemma 6.2.8) and $\Phi_\Pi^{hex}(I_1, I_2)_1 \subseteq \mathcal{T}_\Pi^{hex}(I_1, I_2)_1$ (Lemma 6.5.2). □

The next result shows that, as in the normal case (see Proposition 4.4.4), the ultimate answer-set semantics are "weaker" in the sense that every $\mathcal{F}_\Pi^{hex}$-answer set is also an ultimate answer set.

Theorem 6.5.4. *Let Π be a HEX program, and let $I \in \mathcal{I}_\Pi$. If I is an $\mathcal{F}_\Pi^{hex}$-answer set, then I is an ultimate answer set of Π.*

Proof. Let $I \in \mathcal{F}_\Pi^{hex,\downarrow}(I) = \mathrm{mfp}(\mathcal{F}_\Pi^{hex}(\,.\,, I))$. Then, by Proposition 6.3.2 and Proposition 6.4.4, we have $\mathcal{N}_\Pi^{hex}(I, I) = \mathcal{F}_\Pi^{hex}(I, I) = N_\Pi^{hex}(I) = \{I\}$ which shows that I is a fixpoint of $\mathcal{N}_\Pi^{hex}(\,.\,, I)$. To prove minimality, suppose there exists some $J \subsetneq I$ such that J is a fixpoint of $\mathcal{N}_\Pi^{hex}(\,.\,, I)$. Then, J is a pre-fixpoint of $\mathcal{N}_\Pi^{hex}(\,.\,, I)$ and, hence, by Proposition 6.5.3,

$$\mathcal{F}_\Pi^{hex}(J, I) \sqsubseteq \mathcal{N}_\Pi^{hex}(J, I) \sqsubseteq \{J\},$$

that is, J is a pre-fixpoint of $\mathcal{F}_\Pi^{hex}(\,.\,, I)$ and, hence, a fixpoint; a contradiction to the minimality of I. Consequently, $I \in \mathcal{N}_\Pi^{hex,\downarrow}(I)$. □

The next example shows that the converse of Theorem 6.5.4 does, in general, not hold.

Example 6.5.5. Reconsider the normal HEX program Π of Example 4.4.3 consisting of the following rules:

$$a \leftarrow a^{\#}$$
$$a \leftarrow\sim a^{\#}.$$

Since $\Phi_\Pi^{hex}(\emptyset, \{a\})_1 = \emptyset$, we have $\mathcal{F}_\Pi^{hex}(\emptyset, \{a\}) = \{\emptyset\}$, that is, $\emptyset$ is a minimal fixpoint of $\mathcal{F}_\Pi^{hex}(\,.\,, \{a\})$. Hence, $\mathcal{F}_\Pi^{hex,\downarrow}(\{a\}) = \{\emptyset\}$ and $\{a\}$ is thus not an $\mathcal{F}_\Pi^{hex}$-answer set. In contrast, since $\mathcal{T}_\Pi^{hex}(\emptyset, \{a\})_1 = \{a\}$, we have $\mathcal{N}_\Pi^{hex,\downarrow}(\{a\}) = \{\{a\}\}$ which shows that $\{a\}$ is an ultimate answer set.

Example 6.5.6. Reconsider the disjunctive HEX program Π of Example 6.4.2 consisting of the following rules:

$$p(a) \vee q(a) \leftarrow$$
$$\bot \leftarrow \sim p \subseteq^{\#} q, \sim \bot.$$

We have seen in Example 6.4.2 that $I = \{q(a)\}$ is the only $\mathcal{F}_{\Pi}^{hex}$-answer set. By using $\mathcal{N}_{\Pi}^{hex}$ instead of $\mathcal{F}_{\Pi}^{hex}$, the same calculation shows that I is also an ultimate answer set of Π. In detail, $\mathcal{N}_{\Pi}^{hex}(\emptyset, I) = \{\{p(a)\}, I\}$ shows that $\emptyset$ is not a fixpoint of $\mathcal{N}_{\Pi}^{hex}(\,.\,, I)$; moreover, $\mathcal{N}_{\Pi}^{hex}(I, I) = N_{\Pi}^{hex}(I) = MM\,(\{q(a), p(a) \vee q(a)\}) = \{I\}$ shows that I is a minimal fixpoint of $\mathcal{N}_{\Pi}^{hex}(\,.\,, I)$ and, hence, an ultimate answer set of Π.

CHAPTER 7

Semantic Properties of Disjunctive HEX Programs

In this chapter we show the following semantic properties of the (ultimate) answer-set semantics defined in Section 6.4 and Section 6.5. In Section 7.1 we show that the answer-set semantics of disjunctive HEX programs extends the *2-valued* answer-set semantics of normal HEX programs as defined in Section 4.2.

Recall from Section 6.4 and Section 6.5 that (ultimate) answer sets are defined in terms of *minimal* fixpoints. This definitions are non-constructive. In Section 7.2 we show that every (ultimate) answer-set is derivable by a bottom-up computation. Moreover, given an answer set I, we define an algorithm for constructing a computation with result I.

In Section 7.3 we show that (ultimate) answer sets are supported. That is, in every (ultimate) answer set I each atom is justified by some rule of the program.

Finally, in Section 7.4 we show that every answer set is an FLP-answer set. The converse, however, generally does not hold (see Example 7.4.2).

7.1 Disjunctive Answer-Set Semantics extend Normal Answer-Set Semantics

In Chapter 4 we defined (2-, and 3-valued ultimate) answer-set semantics of normal HEX programs (see Section 4.2 and Section 4.4) by applying Approximation Theory (Denecker et al., 2000a, 2002, 2004). Moreover, in Section 6.4 we defined 2-valued answer-set semantics for the class of disjunctive HEX programs by adapting concepts from Approximation Theory to the class of non-deterministic functions. Clearly, every normal HEX program is, formally, also a disjunctive HEX program. Hence, the question arises whether the 2-valued answer-set semantics of disjunctive HEX programs extends the 2-valued answer-set semantics of normal HEX programs. Theorem 7.1.4 answers this question positively, i.e., we show that, given a normal HEX program Π, the 2-valued Φ_{Π}^{hex}-answer sets coincide with the $\mathcal{F}_{\Pi}^{hex}$-answer sets, and the

ultimate answer sets with respect to $\mathcal{T}_\Pi^{hex}$ coincide with the ultimate answer sets with respect to $\mathcal{N}_\Pi^{hex}$.

Example 7.1.1. Reconsider the normal HEX program Π of Example 4.2.9 consisting of the following rules:

$$p(a) \leftarrow\sim (\exists x\ q(x))^{\#}$$
$$q(b) \leftarrow (\exists x\ q(x))^{\#}.$$

In Example 4.2.9 we have seen that $I = \{p(a)\}$ is the only Φ_Π^{hex}-answer set. We show that I is also an $\mathcal{F}_\Pi^{hex}$-answer set, i.e., $I \in \mathcal{F}_\Pi^{hex,\downarrow}(I) = \mathrm{mfp}\left(\mathcal{F}_\Pi^{hex}(\,.\,, I)\right)$. First, we have

$$\mathcal{F}_\Pi^{hex}(\emptyset, I) = MM\left(\Phi_\Pi^{hex}(\emptyset, I)_1\right) = MM(I) = \{I\}.$$

This shows that $\emptyset$ is not a fixpoint of $\mathcal{F}_\Pi^{hex}(\,.\,, I)$. Second, we have

$$\mathcal{F}_\Pi^{hex}(I, I) = N_\Pi^{hex}(I) = MM\left(I \cup T_\Pi^{hex}(I)\right) = MM(I) = \{I\}$$

which shows that I is indeed a minimal fixpoint of $\mathcal{F}_\Pi^{hex}(\,.\,, I)$ and, consequently, an $\mathcal{F}_\Pi^{hex}$-answer set.

Let Π be a HEX program. Recall from Section 6.3 that $\mathcal{F}_\Pi^{hex}$ and $\mathcal{N}_\Pi^{hex}$ are defined in terms of the non-deterministic operator MM selecting, from a given set $D \subseteq DHB_\Pi$ of disjunctions, the minimal models of D. However, if D contains only atoms (i.e., $D \in \mathcal{I}_\Pi$), MM is deterministic. This is formally stated in the next lemma.

Lemma 7.1.2. *Let Π be a HEX program. Then, for each $I \in \mathcal{I}_\Pi$, $MM(I) = \{I\}$.*

From Lemma 7.1.2 we immediately conclude the following result.

Corollary 7.1.3. *Let Π be a normal HEX program. Then, for each $(I_1, I_2) \in \mathcal{I}_\Pi^c$,*

1. *$\mathcal{F}_\Pi^{hex}(I_1, I_2) = \left\{I_1 \cup \Phi_\Pi^{hex}(I_1, I_2)_1\right\}$, and*
2. *$\mathcal{N}_\Pi^{hex}(I_1, I_2) = \left\{I_1 \cup \mathcal{T}_\Pi^{hex}(I_1, I_2)_1\right\}$.*

We now prove the assertion stated in the beginning of this section.

Theorem 7.1.4. *Let Π be a normal HEX program, and let $I \in \mathcal{I}_\Pi$. Then we have the following equivalences:*

1. *I is an Φ_Π^{hex}-answer set iff I is an $\mathcal{F}_\Pi^{hex}$-answer set.*
2. *I is an ultimate answer set with respect to $\mathcal{T}_\Pi^{hex}$ iff I is an an ultimate answer set with respect to $\mathcal{N}_\Pi^{hex}$.*

Proof. (1) ($\Rightarrow$) Suppose I is an Φ^{hex}_{Π}-answer set, i.e., $I = \mathrm{lfp}\left(\Phi^{hex}_{\Pi}(\,.\,, I)_1\right)$. First, since $\Phi^{hex}_{\Pi}(I, I)_1 = I$, we have $\mathcal{F}^{hex}_{\Pi}(I, I) = \{I\}$. Second, suppose there exists some $J \subsetneq I$ such that $J \in \mathcal{F}^{hex}_{\Pi}(J, I) = \{J \cup \Phi^{hex}_{\Pi}(J, I)_1\}$. Then, $\Phi^{hex}_{\Pi}(J, I)_1 \subseteq J$ which implies $\Phi^{hex}_{\Pi}(J, I)_1 = J$,[1] a contradiction to the assumption that I is an Φ^{hex}_{Π}-answer set.

($\Leftarrow$) Since, by assumption, I is $\mathcal{F}^{hex}_{\Pi}$-I-derivable (see Section 7.2), there exists a $\mathcal{F}^{hex}_{\Pi}$-I-computation $J^{I,\uparrow} = (J_i)_{i\geq 0}$ with result $J^{I,\infty} = I$. Moreover, by Corollary 7.1.3, $\mathcal{F}^{hex}_{\Pi}(\,.\,, I)$ is deterministic, and since $\Phi^{hex}_{\Pi}(\,.\,, I)_1$ is monotone with respect to set inclusion, it is easy to see that $J_i = \Phi^{hex,i}_{\Pi}(\emptyset, I)$ for every $i \geq 0$. Finally, since I is the result of $J^{I,\uparrow}$, we conclude that I is the least fixpoint of $\Phi^{hex}_{\Pi}(\,.\,, I)_1$.

(2) Analogous to (1).

□

7.2 Answer Sets are Derivable

Recall from Section 4.2 (see Proposition 4.2.7) and Section 4.4 that 2-valued (ultimate) answer sets of normal HEX programs have a constructive fixpoint characterization. However, in the case of disjunctive HEX programs, the definition of (ultimate) answer sets given in Section 6.4 is non-constructive. Nevertheless, in this section we show that each (ultimate) answer set is derivable, i.e., has a bottom-up computation (see Definition 6.3.3). Moreover, we present an algorithm for constructively computing (ultimate) answer sets.

Example 7.2.1. Let Π be the disjunctive HEX program of Example 6.4.2 consisting of the following rules:

$$p(a) \vee q(a) \leftarrow$$
$$\bot \leftarrow \sim p \subseteq^{\#} q, \sim \bot,$$

In Example 6.4.2 we have seen that $I = \{q(a)\}$ is the only $\mathcal{F}^{hex}_{\Pi}$-answer set. We define a $\mathcal{F}^{hex}_{\Pi}$-I-computation $J^{I,\uparrow}$ with result $J^{I,\infty} = I$ as follows. Let $J_0 = \emptyset$, and $J_i = I$ for every $i \geq 1$. Since $I \in \mathcal{F}^{hex}_{\Pi}(\emptyset, I)$ and $I \in \mathcal{F}^{hex}_{\Pi}(I, I) = \{I\}$, $J^{I,\uparrow} = (J_i)_{i\geq 0}$ is a $\mathcal{F}^{hex}_{\Pi}$-I-computation with result $J^{I,\infty} = I$.

Let Π be a disjunctive HEX program, and let $\mathcal{A}^{hex}_{\Pi} \in \left\{\mathcal{F}^{hex}_{\Pi}, \mathcal{N}^{hex}_{\Pi}\right\}$. Algorithm 2 presents a guess & check algorithm for constructively computing the $\mathcal{A}^{hex}_{\Pi}$-answer sets of Π (see Figure 7.2); in Theorem 7.2.2 we prove the soundness and the completeness of Algorithm 2.

Theorem 7.2.2 (Soundness, Completeness)**.** *Let Π be a HEX program, let $\mathcal{A}^{hex}_{\Pi} \in \{\mathcal{F}^{hex}_{\Pi}, \mathcal{N}^{hex}_{\Pi}\}$, and let $I \in \mathcal{I}_{\Pi}$. Then, I is an $\mathcal{A}^{hex}_{\Pi}$-answer set of Π iff $I \in ComputateAnswerSets(I, \mathcal{A}^{hex}_{\Pi})$ where $ComputateAnswerSets(I, \mathcal{A}^{hex}_{\Pi})$ is defined as in Algorithm 2.*

Proof. ($\Rightarrow$) We have to show that if I is an $\mathcal{A}^{hex}_{\Pi}$-answer set, then (i) $I = Computation(I, \mathcal{A}^{hex}_{\Pi})$ where $Computation(I, \mathcal{A}^{hex}_{\Pi})$ is defined as in Algorithm 1, and (ii) I is a minimal fixpoint of

[1] Notice that $\Phi^{hex}_{\Pi}(\,.\,, I)_1$ is monotone with respect to set inclusion (see Section 4.2). Hence, we have $J \subseteq \Phi^{hex}_{\Pi}(J, I)_1$.

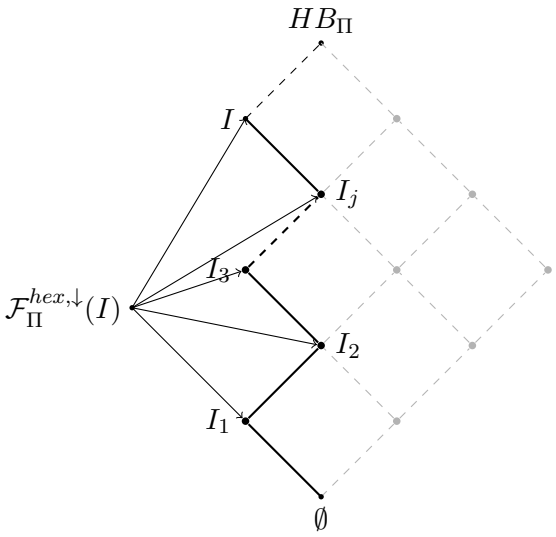

Figure 7.1: Non-deterministic computation of $I \in \mathrm{mfp}\left(\mathcal{F}_\Pi^{hex}(\,.\,, I)_1\right)$.

$\mathcal{A}_\Pi^{hex}(\,.\,, I)$. Part (ii) follows directly from the definition of an $\mathcal{A}_\Pi^{hex}$-answer set (see Definition 6.4.1). For part (i) we have to show that for each $\mathcal{A}_\Pi^{hex}$-I-computation $J^{I,\uparrow}$, $J^{I,\infty} = I$. Let $J_0 = \emptyset$. Since I is a fixpoint of $\mathcal{A}_\Pi^{hex}(\,.\,, I)$, we have $\mathcal{A}_\Pi^{hex}(I, I) = \{I\}$. Furthermore, since $\mathcal{A}_\Pi^{hex}(\,.\,, I)$ is monotone (Proposition 6.3.2) and $(J_0, I) \subseteq_p I$, we have

$$\mathcal{A}_\Pi^{hex}(J_0, I) \sqsubseteq \mathcal{A}_\Pi^{hex}(I, I) \sqsubseteq \{I\},$$

that is, there exists some $J_1 \in \mathcal{A}_\Pi^{hex}(J_0, I)$ such that $J_0 \subseteq J_1 \subseteq I$. Generally, we obtain, for every $n \geq 0$, $J_{n+1} \in \mathcal{A}_\Pi^{hex}(J_n, I)$ for some $J_n \subseteq J_{n+1} \subseteq I$. Clearly, this construction yields a computation $J^{I,\uparrow}$ with result $J^{I,\infty} \subseteq I$. By Proposition 6.4.7, $J^{I,\infty}$ is a fixpoint of $\mathcal{A}_\Pi^{hex}(\,.\,, I)$. Consequently, since I is, by assumption, a minimal fixpoint of $\mathcal{A}_\Pi^{hex}(\,.\,, I)$, $J^{I,\infty} = I$. Hence, we have $I = Computation(I, \mathcal{A}_\Pi^{hex})$.

($\Leftarrow$) Let $J^{I,\infty} = Computation(I, \mathcal{A}_\Pi^{hex})$. By definition of Algorithm 1, $J^{I,\infty}$ is the result of some $\mathcal{A}_\Pi^{hex}$-I-computation. Hence, if $J^{I,\infty} = I$, then I is a fixpoint of $\mathcal{A}_\Pi^{hex}(\,.\,, I)$. Moreover, in line numbers 7-11 of Algorithm 2 we check whether I is a minimal fixpoint of $\mathcal{A}_\Pi^{hex}(\,.\,, I)$. Since, by assumption, $I \in ComputeAnswerSets(I, \mathcal{A}_\Pi^{hex})$, we have $I \in \mathrm{mfp}(\mathcal{A}_\Pi^{hex}(\,.\,, I))$ which proves that I is an $\mathcal{A}_\Pi^{hex}$-answer set. □

However, in general, there exists no *canonical* $\mathcal{A}_\Pi^{hex}$-I-computation, as the following example illustrates.

Algorithm 1 Construct an $\mathcal{A}_{\Pi}^{hex}$-I-computation with result $J^{I,\infty}$.

Input: An interpretation $I \in \mathcal{I}_{\Pi}$, and an approximation $\mathcal{A}_{\Pi}^{hex} \in \{\mathcal{F}_{\Pi}^{hex}, \mathcal{N}_{\Pi}^{hex}\}$.
Output: The result $J^{I,\infty}$ of an $\mathcal{A}_{\Pi}^{hex}$-I-computation.

procedure COMPUTATION(I,$\mathcal{A}_{\Pi}^{hex}$)
  $J_0 \leftarrow \emptyset$
  $i \leftarrow 0$
  repeat
    $J_{i+1} \leftarrow$ Select *any* $J_{i+1} \in \mathcal{A}_{\Pi}^{hex}(J_i, I)$ such that $J_{i+1} \subseteq I$
    $i \leftarrow i + 1$
  until $J_i = J_{i-1}$
  $J^{I,\infty} \leftarrow J_i$
  return $J^{I,\infty}$
end procedure

Algorithm 2 Compute the $\mathcal{A}_{\Pi}^{hex}$-answer sets by guess & check.

Input: A HEX program Π, and an approximation $\mathcal{A}_{\Pi}^{hex} \in \{\mathcal{F}_{\Pi}^{hex}, \mathcal{N}_{\Pi}^{hex}\}$.
Output: The (2-valued) $\mathcal{A}_{\Pi}^{hex}$-answer sets of Π.

procedure COMPUTATEANSWERSETS(Π,$\mathcal{A}_{\Pi}^{hex}$)
  $\mathcal{AS} \leftarrow \emptyset$
  for $I \in \mathcal{I}_{\Pi}$ **do**
    $J^{I,\infty} \leftarrow Computation(I, \mathcal{A}_{\Pi}^{hex})$
    if $J^{I,\infty} = I$ **then**
      $\mathrm{mfp} \leftarrow \mathbf{t}$
      for $J \subsetneq I$ **do** ▷ $I \in \mathrm{mfp}(\mathcal{A}_{\Pi}^{hex}(\,.\,, I))$?
        if $J \in \mathcal{A}_{\Pi}^{hex}(J, I)$ **then**
          $\mathrm{mfp} \leftarrow \mathbf{f}$
        end if
      end for
      if $\mathrm{mfp} = \mathbf{t}$ **then**
        $\mathcal{AS} \leftarrow \mathcal{AS} \cup \{I\}$
      end if
    end if
  end for
  return $\mathcal{AS}$
end procedure

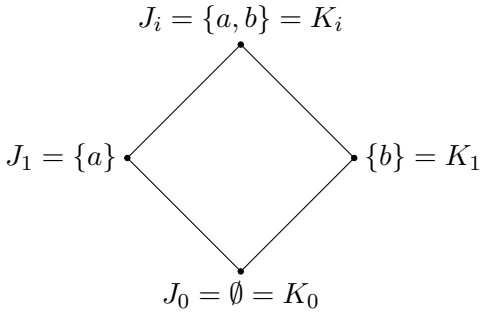

Figure 7.2: Computations $J^{I,\uparrow}$ and $K^{I,\uparrow}$ of Example 7.2.3.

Example 7.2.3. Reconsider the classical disjunctive program Π of Example 6.2.6 consisting of the following propositional rules:

$$\begin{aligned} a \vee b \leftarrow \\ a \leftarrow b \\ b \leftarrow a. \end{aligned}$$

We show that $I = \{a, b\}$ is an $\mathcal{F}^{hex}_{\Pi}$-answer set and that there exist two equally adequate $\mathcal{F}^{hex}_{\Pi}$-I-computations $J^{I,\uparrow}$ and $K^{I,\uparrow}$ with results $J^{I,\infty} = K^{I,\infty} = I$. Let $J_0 = K_0 = \emptyset$. Applying $\mathcal{F}^{hex}_{\Pi}$ to $(\emptyset, I)$ yields $\mathcal{F}^{hex}_{\Pi}(\emptyset, I) = \{\{a\}, \{b\}\}$. Let $J_1 = \{a\}$ and $K_1 = \{b\}$, and observe $J_1, K_1 \subseteq I$. Finally, we have $\mathcal{F}^{hex}_{\Pi}(J_1, I) = \mathcal{F}^{hex}_{\Pi}(K_1, I) = \{I\}$. Let $J_i = K_i = I$ for every $i \geq 2$. This yields two computations $J^{I,\uparrow} = (J_i)_{i \geq 0}$ and $K^{I,\uparrow} = (K_i)_{i \geq 0}$ with results $J^{I,\infty} = K^{I,\infty} = I$ as desired (see Figure 7.2).

7.3 Answer Sets are Supported

A basic requirement for semantics of logic programs is that every intended model contains only atoms justified by the program (Apt et al., 1988), i.e., only *supported* atoms (see Example 3.2.3). In this section we show that the (ultimate) answer-set semantics as defined in Section 6.4 and Section 6.5 is supported.

In the sequel let Π be a disjunctive HEX program. We say that an interpretation $I \in \mathcal{I}_{\Pi}$ is (i) *weakly supported* (Brass and Dix, 1997) if for every atom $a \in I$ there exists some rule $r \in \Pi$ such that $a \in H(r)$ and $I \models B(r)$, and (ii) *supported* (Brass and Dix, 1997) if for every atom $a \in I$ there exists some rule $r \in \Pi$ such that $I \models B(r)$ and $I \not\models H(r) - \{a\}$. Clearly, if I is supported, then I is weakly supported.

Example 7.3.1. Consider the classical disjunctive program Π consisting of the following single rule:

$$a \vee b \leftarrow\sim c.$$

Then, for instance, $\{a\}$ and $\{b\}$ are (weakly) supported, $\{a, b\}$ is weakly supported but not supported, and $\{c\}$ is not (weakly) supported.

We now prove that every $\mathcal{F}^{hex}_{\Pi}$-, and ultimate answer set of Π is (weakly) supported.

Theorem 7.3.2. *Let Π be a disjunctive HEX program, and let $\mathcal{A}^{hex}_{\Pi} \in \{\mathcal{F}^{hex}_{\Pi}, \mathcal{N}^{hex}_{\Pi}\}$. For every $I \in \mathcal{I}_{\Pi}$, if I is an $\mathcal{A}^{hex}_{\Pi}$-answer set, then I is supported.*

Proof. Suppose that I is not supported, i.e., there exists some atom $a \in I$ such that for every $r \in \Pi$ with $a \in H(r)$, $I \not\models B(r)$ or there exists some $a' \in H(r)$, $a' \neq a$, such that $a' \in I$. Let $I' = I - \{a\}$. We claim that I' is a model of $\Phi^{hex}_{\Pi}(I', I)_1$. Let $d \in \Phi^{hex}_{\Pi}(I', I)_1$ be an arbitrary disjunction, and let $r \in \Pi$ be a rule such that $H(r) = d$ and $\langle B(r) \rangle_{(I',I)} = \mathbf{t}$. Then, since $(I', I) \subseteq_p I$, we have, by Lemma 4.1.10, $I \models B(r)$. We distinguish the following two cases: (i) if $a \notin H(r)$ then $I' \models H(r)$ by definition of I' (notice that I is, by assumption, a model of Π and thus $I \models H(r)$); and (ii) if $a \in H(r)$ then, by assumption, there exists some $a' \in H(r)$, $a' \neq a$, such that $a' \in I'$, i.e., $I' \models H(r)$. Consequently, I' is indeed a model of $\Phi^{hex}_{\Pi}(I', I)_1$. Therefore, by definition of $\mathcal{F}^{hex}_{\Pi}$, I' is a fixpoint of $\mathcal{F}^{hex}_{\Pi}(\,.\,, I)$, a contradiction to the minimality of I.

Since $d \in \mathcal{T}^{hex}_{\Pi}(I', I)_1$ only if $d \in \bigcap_{I' \subseteq J \subseteq I} T^{hex}_{\Pi}(J)$ only if there exists a rule $r \in \Pi$ with $H(r) = d$ and $I \models B(r)$, the case for $\mathcal{N}^{hex}_{\Pi}$ is analogous. □

Corollary 7.3.3. *Let Π be a disjunctive HEX program, and let $\mathcal{A}^{hex}_{\Pi} \in \{\mathcal{F}^{hex}_{\Pi}, \mathcal{N}^{hex}_{\Pi}\}$. For every $I \in \mathcal{I}_{\Pi}$, if I is an $\mathcal{A}^{hex}_{\Pi}$-answer set, then I is weakly supported.*

7.4 Answer Sets are FLP-Answer Sets

Given a *normal* HEX program Π, in Section 5.2 we have seen that every 2-valued Φ^{hex}_{Π}-answer set is an FLP-answer set (see Theorem 5.2.5); however, the converse does, in general, not hold (see Example 5.2.6). We argued in Section 5.4 that this divergence is due to the well-supportedness (Shen, 2011) of 2-valued Φ^{hex}_{Π}-answer-set semantics.

Now the question naturally arises whether every $\mathcal{F}^{hex}_{\Pi}$-answer set is an FLP-answer set. The next theorem gives a positive answer to this question.

Theorem 7.4.1. *Let Π be a disjunctive HEX program, and let $I \in \mathcal{I}_{\Pi}$. If I is an $\mathcal{F}^{hex}_{\Pi}$-answer set, then I is an FLP-answer set of Π.*

Proof. By assumption, I is a fixpoint of $\mathcal{F}^{hex}_{\Pi}(\,.\,, I)$. Hence, by Proposition 6.3.2, we have

$$\mathcal{F}^{hex}_{\Pi}(I, I) = N^{hex}_{\Pi}(I) = \{I\},$$

which implies that I is a model of Π (Proposition 6.1.7), and thus a model of $f\Pi^I$.

Suppose there exists some interpretation $J \subsetneq I$ such that J is a model of $f\Pi^I$. We claim that there exists a $\mathcal{F}^{hex}_{\Pi}$-I-computation $K^{I,\uparrow} = (K_i)_{i \geq 0}$ with $K^{I,\infty} \subseteq J$. By Lemma 5.2.4, we have, for each $(I_1, I_2) \in \mathcal{I}^c_{\Pi}$,

$$\Phi^{hex}_{\Pi}(I_1, I_2)_1 = \Phi^{hex}_{f\Pi^I}(I_1, I_2)_1.$$

Hence, we have, for each $(I_1, I_2) \in \mathcal{I}^c_\Pi$,

$$\mathcal{F}^{hex}_\Pi(I_1, I_2) = \mathcal{F}^{hex}_{f\Pi^I}(I_1, I_2). \tag{7.1}$$

Let $K_0 = \emptyset$. Since $(K_0, I) \subseteq_p (J, I) \subseteq_p J$ we have, by Lemma 4.1.10 and (7.1),

$$\mathcal{F}^{hex}_\Pi(K_0, I) = \mathcal{F}^{hex}_{f\Pi^I}(K_0, I) \sqsubseteq \mathcal{F}^{hex}_{f\Pi^I}(J, I) \sqsubseteq \mathcal{F}^{hex}_{f\Pi^I}(J, J) = N^{hex}_{f\Pi^I}(J) \sqsubseteq \{J\}, \tag{7.2}$$

where the last equation follows from the assumption that J is a model of $f\Pi^I$ together with Proposition 6.1.7. Hence, there exists some $K_1 \subseteq I$ such that $K_1 \in \mathcal{F}^{hex}_\Pi(K_0, I)$. With the same argument as in (7.2) we obtain $\mathcal{F}^{hex}_\Pi(K_1, I) \sqsubseteq \{J\}$ which entails that there exists some $K_2 \subseteq I$ such that $K_2 \in \mathcal{F}^{hex}_\Pi(K_1, I)$. Iterating this process yields a $\mathcal{F}^{hex}_\Pi$-I-computation $K^{I,\uparrow}$ with $K^{I,\infty} \subseteq J \subsetneq I$, a contradiction to Theorem 7.2.2. Consequently, I is a minimal model of $f\Pi^I$ and, hence, an FLP-answer set of Π. □

In contrast to Theorem 7.4.1, the next example shows that, in general, not every FLP-answer set is an $\mathcal{F}^{hex}_\Pi$-answer set.

Example 7.4.2. Reconsider the normal HEX program Π of Example 5.2.6 consisting of the following rules:

$$a \leftarrow f^{\#}[a, b]$$
$$b \leftarrow g^{\#}[a, b]$$

where f and g are defined as in Table 5.1. We have seen in Example 5.2.6 that $I = \{a, b\}$ is an FLP-answer set of Π, but not an ultimate answer set with respect to $\mathcal{T}^{hex}_\Pi$. Hence, since Π is normal, by Theorem 7.1.4 I is not an ultimate answer set with respect to $\mathcal{N}^{hex}_\Pi$.

CHAPTER 8

Related Work

In this chapter we consider related work. In Section 8.1 we compare in detail our approach used in Chapter 6, and the approach used by (Pelov and Truszczyński, 2004; Pelov, 2004).

In Section 8.2 we argue that the disjunctive Fitting operator defined by (Calimeri et al., 2006) is, in general, not appropriate for defining (ultimate) semantics of *arbitrary* classical disjunctive programs based on Approximation Theory.

In Section 8.3 we compare our 2-valued (ultimate) answer-set semantics of normal HEX programs, as defined in Section 4.2, with strong and weak answer-set semantics (Eiter et al., 2004a, 2008).

Finally, in Section 8.4 we compare our semantics with well-supported semantics as defined in (Shen, 2011).

8.1 Comparison to Pelov and Truszczyński (2004)

In Chapter 6 we used *non-deterministic* operators for defining 2-valued (ultimate) answer-set semantics of *disjunctive* HEX programs. The approach we used is motivated by the work of Pelov and Truszczyński (2004) and Pelov (2004, Chapter 4.4). However, the approaches are not entirely identical, so in this section we elaborate these differences in detail.

Pelov (2004) and Pelov and Truszczyński (2004) extend partially concepts of Approximation Theory to the class of non-deterministic operators by combining ideas from disjunctive logic programming (Minker and Rajasekar, 1990; Lobo et al., 1992; Fernández and Minker, 1995; Seipel et al., 1997) and Approximation Theory. Since aggregates can be simulated by external atoms (see Section 3.1 in (Eiter et al., 2005)), we translate the definitions given in (Pelov and Truszczyński, 2004; Pelov, 2004) to the language of HEX programs and define, for a given disjunctive HEX program Π, the *non-deterministic immediate consequence operator* (Pelov and Truszczyński, 2004; Pelov, 2004), for each $I \in \mathcal{I}_\Pi$, by

$$N_\Pi^{Sel}(I) = Sel\left(T_\Pi^{hex}(I)\right), \tag{8.1}$$

where $Sel : \mathfrak{P}(DHB_\Pi) \longrightarrow \mathfrak{P}(\mathcal{I}_\Pi)$ is a *selection function* satisfying the following axioms:[1]

1. $Sel(D) \subseteq \{I \in \mathcal{I}_\Pi : I \models D\}$
2. $Sel(D) \sqsubseteq \{I \in \mathcal{I}_\Pi : I \models D\}$
3. $I \subseteq \Lambda_D$ for every $I \in Sel(D)$.

Clearly, the function MM (see (6.3)), selecting the minimal models of D, is a selection function (see Proposition 2 in (Pelov and Truszczyński, 2004)). Since we only used MM in this thesis (see Chapter 6), we focus on N_Π^{MM} in the sequel.

Pelov and Truszczyński (2004) proposed the notion of *computation* (Marek et al., 2004) as an appropriate formalization of the process of "iterating" non-deterministic operators. In Chapter 6 we successfully applied this notion to non-deterministic (ultimate) approximations, and proved in Section 7.2 that (ultimate) answer sets are *derivable*. However, the following example, presented in (Pelov and Truszczyński, 2004), shows that the definition of N_Π^{MM} is not compatible with the notion of computation.

Example 8.1.1 (Pelov and Truszczyński (2004), Example 3)**.** Consider the classical disjunctive program Π consisting of the following propositional rules:

$$\begin{aligned} a \vee b \vee c \leftarrow & \\ a \leftarrow b & \\ b \leftarrow c & \\ c \leftarrow a. & \end{aligned}$$

Observe that $I = \{a, b, c\}$ is the only model of Π. Let $I_0 = \emptyset$. By applying N_Π^{MM} to I_0, we obtain $N_\Pi^{MM}(I_0) = \{\{a\}, \{b\}, \{c\}\}$. However, since $N_\Pi^{MM}(\{a\}) = \{\{c\}\}$, $N_\Pi^{MM}(\{b\}) = \{\{a\}\}$, and $N_\Pi^{MM}(\{c\}) = \{\{b\}\}$, there is no $I_1 \in N_\Pi^{MM}(I_0)$ such that there exists some $I_2 \in N_\Pi^{MM}(I_1)$ such that $I_1 \subseteq I_2$. Hence, there exists no computation $I^\uparrow = (I_i)_{i \geq 0}$ with result $I^\infty = I$ (Pelov and Truszczyński, 2004). In contrast, we have $N_\Pi^{hex}(I_0) = N_\Pi^{MM}(I_0)$, but $N_\Pi^{hex}(\{a\}) = \{\{a, c\}\}$, $N_\Pi^{hex}(\{b\}) = \{\{a, b\}\}$, and $N_\Pi^{hex}(\{c\}) = \{\{b, c\}\}$; Furthermore, $N_\Pi^{hex}(\{a, c\}) = N_\Pi^{hex}(\{a, b\}) = N_\Pi^{hex}(\{b, c\}) = I$ which shows that I is derivable by some computations with respect to N_Π^{hex}.

Let us analyze Example 8.1.1, and the difference between N_Π^{MM} and N_Π^{hex}, more precisely. Recall from Section 6.1 that N_Π^{hex} is designed in such a way that the assumptions made about disjunctions are contained in every outcome of N_Π^{hex} applied to an interpretation (see Example 6.1.4, and the discussion after Example 6.2.6). That is, each computation of I (with respect to N_Π^{hex}) represents one path of possible assumptions. For instance, $I_1 = \{a\} \in N_\Pi^{hex}(I_0)$ represents the assumption that in $a \vee b \vee c$ the atom a is *true*, whereas b and c are *false*. From this assumption, we derive with the last rule that c is also *true*; since $a \vee b \vee c \leftarrow$ does not contain the information that in the current computation we assume a to be *true*, we have to add a to the outcome, resulting in $I_2 = \{a, c\} \in N_\Pi^{hex}(I_1)$. In contrast, after applying N_Π^{MM} to I_1,

[1] Recall from Section 6.1 that DHB_Π is the set of all disjunctions which can be formed with atoms from Λ_Π.

the immediate consequence operator N_{Π}^{MM} "forgets" that a has been assumed, and only derives $\{c\}$ (i.e., N_{Π}^{MM} treats disjunctive facts in the same way as non-disjunctive facts).

Finally, it is worth noting that we additionally defined *ultimate* answer-set semantics which is more precise than $\mathcal{F}_{\Pi}^{hex}$-answer-set semantics (see Section 6.5).

8.2 Disjunctive Fitting Operator

Calimeri et al. (2006) extend the Fitting operator (Fitting, 1985) to the class of classical disjunctive programs as follows. Given a classical disjunctive program Π, define the *disjunctive extended Fitting operator* (Calimeri et al., 2006), for each $(I_1, I_2) \in \mathcal{I}_{\Pi}^{c}$, by

$$\Phi_{\Pi,\vee}(I_1, I_2) = \left(I_1', I_2'\right),$$

such that

$$I_1' = \left\{a \in H(r) : r \in \Pi : \langle\, B(r)\,\rangle_{(I_1,I_2)} = \mathbf{t} \text{ and } \left\langle\, a'\,\right\rangle_{(I_1,I_2)} = \mathbf{f} \text{ for every } a' \in H(r) - \{a\}\right\}$$
$$I_2' = \left\{a \in H(r) : r \in \Pi : \langle\, B(r)\,\rangle_{(I_1,I_2)} \geq \mathbf{u} \text{ and } \left\langle\, a'\,\right\rangle_{(I_1,I_2)} \leq \mathbf{u} \text{ for every } a' \in H(r) - \{a\}\right\}.$$

The operator $\Phi_{\Pi,\vee}$ is an approximation of the *disjunctive van Emden-Kowalski operator* defined, for each $I \in \mathcal{I}_{\Pi}$, by

$$T_{\Pi,\vee}(I) = \left\{a \in H(r) : r \in \Pi : I \models B(r) \text{ and } I \not\models a' \text{ for every } a' \in H(r) - \{a\}\right\}.$$

Notice that $T_{\Pi,\vee}$ is a lattice operator defined on the complete lattice $\mathcal{I}_{\Pi}$. Hence, in principle, Approximation Theory (see Chapter 3) is applicable. That is, by instantiating Theorem 3.4.1, we can define, for each $(I_1, I_2) \in \mathcal{I}_{\Pi}^{c}$, the *disjunctive ultimate approximation* by

$$\mathcal{T}_{\Pi,\vee}(I_1, I_2) = \left(\bigcap_{I_1 \subseteq J \subseteq I_2} T_{\Pi,\vee}(J), \bigcup_{I_1 \subseteq J \subseteq I_2} T_{\Pi,\vee}(J)\right).$$

Moreover, we obtain the stable revision operators $\Phi_{\Pi,\vee}^{\downarrow\uparrow}$ and $\mathcal{T}_{\Pi,\vee}^{\downarrow\uparrow}$ (see Section 3.3) and, hence, disjunctive versions of the (ultimate) fixpoint semantics defined in Chapter 4.

However, the next example shows that, in general, $T_{\Pi,\vee}$ is not an adequate operator for applying Approximation Theory and defining fixpoint semantics of *arbitrary* classical disjunctive programs.

Example 8.2.1. Reconsider the classical disjunctive program Π of Example 8.1.1 consisting of the following propositional rules:

$$\begin{aligned} a \vee b \vee c \leftarrow & \\ a \leftarrow b & \\ b \leftarrow c & \\ c \leftarrow a. & \end{aligned}$$

Observe that $I = \{a, b, c\}$ is an ultimate answer set (with respect to $\mathcal{N}_\Pi^{hex}$) of Π, i.e., a subset minimal fixpoint of $\mathcal{N}_\Pi^{hex}(\,.\,, I)$. In contrast, we have $T_{\Pi,\vee}(\{a, b\}) = \{a, c\}$, $T_{\Pi,\vee}(\{a, c\}) = \{b, c\}$, and $T_{\Pi,\vee}(\{b, c\}) = \{a, b\}$ and, hence, $\mathcal{T}_{\Pi,\vee}(\emptyset, I)_1 = \bigcap_{J \subseteq I} T_{\Pi,\vee}(J) = \emptyset$. That is, we have $\mathcal{T}^{\downarrow}_{\Pi,\vee}(I) = \emptyset$ which proves that I is not an ultimate answer set with respect to $\mathcal{T}_{\Pi,\vee}$. Since $\mathcal{T}_{\Pi,\vee}$ yields the most answer sets, I is not an $\Phi_{\Pi,\vee}$-answer set.

Calimeri et al. (2006) did not apply Approximation Theory (i.e., did not define 3-valued answer-set semantics), but defined only the least fixpoint of $\Phi_{\Pi,\vee}$ with respect to some interpretation $(I_1, I_2) \in \mathcal{I}^c_\Pi$. Moreover, Calimeri et al. (2006) studied syntactically restricted programs – the question whether Approximation Theory in combination with $\Phi_{\Pi,\vee}$ can be reasonably applied to these syntactically restricted classes of disjunctive programs remains open.

8.3 Strong and Weak Answer-Set Semantics

Description logic programs[2] (Eiter et al., 2004a, 2008) are predecessors of HEX programs (Eiter et al., 2005). Formally, a *description logic program* (or *dl-program*) $\mathcal{KB}$ consists of a description logic knowledge base L and an extended classical normal program Γ possibly containing so called *dl-atoms*. Roughly, a dl-atom is a bi-directional link between the logic program Γ and the knowledge base L. Important is here that the truth of a dl-atom d in an interpretation I with respect to L, in symbols $I \models_L d$, can be represented by a Boolean function. Hence, each dl-atom d occurring in $\mathcal{KB}$ can be simulated by an external atom $d^\#_L$ defined, for every interpretation I, by (see Section 3.2 in Eiter et al. (2005))

$$d_L(I) = \begin{cases} \mathbf{t} & \text{if } I \models_L d \\ \mathbf{f} & \text{otherwise.} \end{cases}$$

Given a dl-program $\mathcal{KB}$, we denote by $\Pi_{\mathcal{KB}}$ the *normal* HEX program obtained from $\mathcal{KB}$ by replacing each dl-atom d with $d^\#_L$. For ease of exposition, we will drop the reference to $\mathcal{KB}$ from the notation and simply write Π instead of $\Pi_{\mathcal{KB}}$. Under this translation, we immediately obtain FLP-answer-set semantics and fixpoint semantics of dl-programs. Additionally, Eiter et al. (2004a, 2008) and Eiter et al. (2004b, 2011) defined strong and weak answer-set semantics of arbitrary dl-programs, and well-founded semantics of *monotone* dl-programs, respectively. Let $\mathcal{KB}$ be a dl-program and Π be the corresponding normal HEX program. By $\Lambda^{\#,m}_\Pi$ we mean the set of all external atoms $a \in \Lambda^\#_\Pi$ which are known to be monotone, and define $\Lambda^{\#,?}_\Pi = \Lambda^\#_\Pi - \Lambda^{\#,m}_\Pi$. Given an interpretation $I \in \mathcal{I}_\Pi$, define the *strong Gelfond-Lifschitz reduct* (Eiter et al., 2004a, 2008) of Π with respect to I and L by

$$s\Pi^I_L = \left\{ H(r) \leftarrow B^+(r) - \Lambda^{\#,?}_\Pi : r \in \Pi : I \models B^+(r) \cap \Lambda^{\#,?}_\Pi \text{ and } I \models B^\sim(r) \right\}.$$

Intuitively, we compute the reduct $s\Pi^I_L$ by (i) deleting every rule $r \in \Pi$ such that either $I \not\models a$ for some $a \in B^+(r) \cap \Lambda^{\#,?}_\Pi$, or $I \models b$ for some $\sim b \in B^\sim(r)$, and (ii) deleting from each remaining

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

rule $r \in \Pi$ all literals in $B^{\sim}(r) \cup \left(B^{+}(r) \cap \Lambda_{\Pi}^{\#,?}\right)$. Notice that $s\Pi_L^I$ is a positive (i.e., negation-free) monotone normal HEX program, which implies that $T^{hex}_{s\Pi_L^I}$ has a least fixpoint. Finally, we say that I is a *strong answer set* (Eiter et al., 2004a, 2008) of Π if $I = \mathrm{lfp}(T^{hex}_{s\Pi_L^I})$.

The next example shows that not every strong answer set is also an (ultimate) answer set of Π.

Example 8.3.1. Consider the normal HEX program Π consisting of the following single rule:

$$p(a) \leftarrow\sim (not\ p(a))^{\#} \tag{8.2}$$

where, for each $I \in \mathcal{I}_\Pi$, $I \models (not\ p(a))^{\#}$ iff $I \not\models p(a)$. For readers familiar with the formalism of dl-programs, it is worth noticing that Π has the same behavior as the the dl-program $\mathcal{KB} = (\emptyset, \Gamma)$, where Γ consists of the following single rule:

$$p(a) \leftarrow\sim DL[S \uplus p; \neg S](a) \tag{8.3}$$

where $d = DL[S \uplus p; \neg S](a)$ is a dl-atom, S is a concept, and $\uplus$ is an update operator (see Eiter et al. (2008)). However, we proceed with (8.2). We show that $I = \{p(a)\}$ is a strong answer set of Π. Since we have $I \models\sim (not\ p(a))^{\#}$, the strong Gelfond-Lifschitz reduct $s\Pi_L^I$ is equal to the single fact $p(a) \leftarrow$. Hence, I is the least fixpoint of $T^{hex}_{s\Pi_L^I}$ and, thus, a strong answer set of Π. In contrast, we have $\mathcal{T}_\Pi^{hex}(\emptyset, I)_1 = \emptyset$ which shows $\mathcal{T}_\Pi^{hex,\downarrow}(I) = \emptyset$. That is, I is not an ultimate answer set of Π and, thus, not an Φ_Π^{hex}-answer set (see Proposition 4.4.4).

We now define weak answer-set semantics. Let $\mathcal{KB}$ be a dl-program, and let Π be the corresponding normal HEX program. Given an interpretation $I \in \mathcal{I}_\Pi$, define the *weak Gelfond-Lifschitz reduct* (Eiter et al., 2004a, 2008) of Π with respect to I and L by

$$w\Pi_L^I = \left\{ H(r) \leftarrow B^{+}(r) - \Lambda_\Pi^{\#} : r \in \Pi : I \models B^{+}(r) \cap \Lambda_\Pi^{\#} \text{ and } I \models B^{\sim}(r) \right\}.$$

The only difference to the strong Gelfond-Lifschitz reduct is that in $w\Pi_L^I$ we do not only delete possibly nonmonotone atoms from $\Lambda_\Pi^{\#,?}$, but every external atom from $\Lambda_\Pi^{\#}$. Clearly, $w\Pi_L^I$ is a positive normal program without external atoms and, hence, $T^{hex}_{w\Pi_L^I}$ is monotone and has a least fixpoint. We say that I is a *weak answer set* (Eiter et al., 2004a, 2008) of Π if $I = \mathrm{lfp}(T^{hex}_{w\Pi_L^I})$.

Proposition 8.3.2 (Eiter et al. (2004a), Theorem 11)**.** *Let $\mathcal{KB}$ be a dl-program, and let $\Pi = \Pi_{\mathcal{KB}}$ be defined as above. For every $I \in \mathcal{I}_\Pi$, if I is a strong answer set of Π, then I is a weak answer set of Π.*

Example 8.3.3. Reconsider the normal HEX program Π of Example 8.3.1. We have seen in Example 8.3.1 that Π is the translation of the dl-program $\mathcal{KB} = (\emptyset, \Gamma)$ where Γ consists of the single rule (8.3). Moreover, we have seen that $I = \{p(a)\}$ is a strong answer set of Π. Hence, by Proposition 8.3.2, I is also a weak answer set of Π which shows that not every weak answer set is also an (ultimate) answer set of Π.

Example 8.3.1 and Example 8.3.3 show that our (ultimate) answer-set semantics of Chapter 4 which base on Approximation Theory do not coincide with strong or weak answer-set semantics. In the next section we argue that this divergence is due to the well-supportedness (Shen, 2011; Fages, 1994) of our semantics (see Section 5.4).

8.4 Well-Supported Semantics

In the sequel let $\mathcal{KB} = (L, \Gamma)$ be a dl-program. For every ground literal ℓ, and interpretations $I_1, I_2 \in \mathcal{I}_\Pi$, $I_1 \subseteq I_2$, define the relation "I_1 *up to* I_2 *satisfies* ℓ" (Shen, 2011), denoted $(I_1, I_2) \models \ell$, as follows:

1. For a ground atom $a \in HB_\Pi$, $(I_1, I_2) \models a$ if $a \in I_1$, and $(I_1, I_2) \models\!\sim a$ if $a \notin I_2$.
2. For a ground dl-atom d, $(I_1, I_2) \models d$ if $J \models d$ for every $J \in [I_1, I_2]$, and $(I_1, I_2) \models\!\sim d$ if $J \not\models d$ for every $J \in [I_1, I_2]$.

Consequently, the up to satisfaction relation coincides with the 3-valued evaluation function $\langle\ .\ \rangle_{(I_1,I_2)}$ (see (4.2) and (4.4)), i.e., we have, for each $(I_1, I_2) \in \mathcal{I}^c_\Pi$ and each literal ℓ, the following equivalence:

$$(I_1, I_2) \models \ell \Leftrightarrow \langle\, \ell \,\rangle_{(I_1,I_2)} = \mathbf{t}.$$

Hence, we can rephrase Shen's definition of strongly well-supportedness (see Definition 4 in (Shen, 2011)) as follows. We say that an interpretation I of $\mathcal{KB}$ is *strongly well-supported* if there exists a strict well-founded partial order $\prec$ on I such that for every $a \in I$ there exists a rule $r \in \Gamma$ with $H(r) = a$ and a proper subset $J \subsetneq I$ such that $\langle\, B(r) \,\rangle_{(I_1,I_2)} = \mathbf{t}$, and for every $b \in J$, $b \prec a$. The equivalence between the notion of strongly well-supportedness and our notion of well-supportedness (see Definition 5.4.2) is evident.

Moreover, we can rewrite Shen's *extended van Emden-Kowalski operator* $T_{\mathcal{KB}}$ (see Definition 5 in (Shen, 2011)), for each $(I_1, I_2) \in \mathcal{I}^c_\Pi$, as follows:

$$T_{\mathcal{KB}}(I_1, I_2) = \left\{ H(r) : r \in \Pi : \langle\, B(r) \,\rangle_{(I_1,I_2)} = \mathbf{t} \right\}.$$

Then we have the following equivalence (see Definition 4.1.8).

Proposition 8.4.1. *Let $\mathcal{KB}$ be a dl-program, and let $\Pi = \Pi_{\mathcal{KB}}$ be defined as in Section 8.3. Then, for each $(I_1, I_2) \in \mathcal{I}^c_\Pi$, $T_{\mathcal{KB}}(I_1, I_2) = \Phi^{hex}_\Pi(I_1, I_2)_1$.*

Let I be an interpretation of $\mathcal{KB}$. We say that I is a *strongly well-supported answer set* (Shen, 2011) of $\mathcal{KB}$ if $I = \mathrm{lfp}\,(T_{\mathcal{KB}}(\,.\,, I))$. The following proposition is an immediate consequence of Proposition 8.4.1 and Proposition 4.2.7.

Theorem 8.4.2. *Let $\mathcal{KB}$ be a dl-program, let $\Pi = \Pi_{\mathcal{KB}}$ be defined as in Section 8.3, and let I be an interpretation of $\mathcal{KB}$. Then, I is a strongly well-supported answer set of $\mathcal{KB}$ iff I is a 2-valued Φ^{hex}_Π-answer set.*

Considering the equivalences described above, it comes as no surprise that 2-valued Φ_{Π}^{hex}-answer-set semantics are well-supported (see Theorem 5.4.7). Moreover, the equivalences show that Shen's (strongly) well-supported answer-set semantics is naturally captured within the more general framework of Approximation Theory. However, the use of Approximation Theory allowed us to additionally define the whole class of 3-valued (ultimate) answer-set semantics (which contain well-founded semantics) which lead to a more general approach than the one proposed by (Shen, 2011).

The next example shows that a dl-program consisting of a single rule is not well-supported. That is, it shows that strong and weak answer sets (as defined in Section 8.3) are, in general, not free of circular justifications.

Example 8.4.3. Reconsider the normal HEX program Π of Examples 8.3.1 and 8.3.3 consisting of the single rule r given in (8.2). We have seen in Example 8.3.1 that Π can be interpreted as a translation of the dl-program $\mathcal{KB}$ consisting of the single rule (8.3). By definition of d, we have the following self-supported loop:[3] $p(a) \Leftarrow \neg d^{\#} \Leftarrow p(a)$. Formally, for $I = \{p(a)\}$ to be well-supported, we have to find a subset $J \subsetneq I$ such that $\langle\, B(r) \,\rangle_{(J,I)} = \mathbf{t}$ and $J \prec p(a)$ (see Section 5.4). Since $\emptyset$ is the only proper subset of I, and $\langle\, B(r) \,\rangle_{(\emptyset,I)} \neq \mathbf{t}$, we conclude that I is not well-supported.

Finally, it is worth mentioning that Shen and Wang (2012) extended the strongly well-supported answer-set semantics to the class of *general logic programs* (Bartholomew et al., 2011) where the head and bodies of rules can be arbitrary first-order formulas. They formulated the extension in terms of *well-justified FLP-answer sets*, i.e., FLP-answer sets (see Section 2.2) without circular justifications. The application of Approximation Theory to the class of general logic programs remains an open issue for future research.

[3] Recall from Section 5.4 that $\Leftarrow$ denotes the informal statement "truth is supported by" (Shen, 2011).

CHAPTER 9

Conclusions

For logic programs with negation, a range of different purely declarative semantics have been proposed in the last two decades, among them the Kripke-Kleene semantics (Fitting, 1985), the well-founded semantics (Van Gelder et al., 1991), and the answer-set semantics (Gelfond and Lifschitz, 1991). These semantics have been adopted to a large number of extensions of classical logic programs. However, since one has to adopt each semantics separately, this process is cumbersome in general. Moreover, it is not always clear which of the possible extensions to accept as the "canonical" one.

Approximation Theory (Denecker et al., 2000a, 2002, 2004), on the other hand, uniformly characterizes the mentioned semantics within an abstract algebraic framework by studying the fixpoints of the associated (monotone or nonmonotone) one-step provability operators in terms of monotone approximations. Thus, the framework makes it convenient to extend the semantics to a new class of programs.

The goal of this thesis was to lift the Kripke-Kleene-, well-founded-, and (3-valued) answer-set semantics to the class of HEX programs (Eiter et al., 2005) by applying Approximation Theory, and to compare them with the standard FLP-answer-set semantics. This was in particular relevant, because HEX programs constitute a powerful extension of classical disjunctive programs, and are able to represent various other formalisms (e.g., dl-programs, and logic programs with aggregates).

The main contributions were that we uniformly defined the full class of the mentioned 3-valued semantics for normal (i.e., disjunction-free) HEX programs by applying Approximation Theory. Moreover, we defined ultimate versions of these semantics, which are the most precise one with respect to Approximation Theory. Furthermore, we showed that the recently defined strongly well-supported answer-set semantics (Shen, 2011) are equivalent to our 2-valued answer-set semantics. However, since we used the machinery of Approximation Theory, our approach is much more general and elegant from a mathematical and practical point of view. Finally, we obtained 2-valued (ultimate) answer-set semantics for disjunctive HEX programs and showed that they can be constructively characterized in terms of bottom-up computations.

As a result of our investigation, we obtained constructive and uniform semantics for a general class of logic programs with nice properties. More precisely, in the case of normal HEX programs, it turned out that our 2-valued answer-set semantics are well-supported which is considered to be a key requirement. Moreover, to the best of our knowledge, the introduction of well-founded semantics for this class of programs is novel. Finally, our 2-valued (ultimate) answer-set semantics turned out to be computable in a bottom-up manner, which led to a constructive algorithm for computing the (ultimate) answer sets of a given program.

However, there remain open issues. For the class of disjunctive HEX programs, we introduced only 2-valued answer-set semantics, but no well-founded semantics. Therefore, one line of future research is to define well-founded semantics at least for syntactically restricted classes of disjunctive HEX programs. Moreover, Approximation Theory has been extended to algebraically capture the notions of strong and uniform equivalence (Truszczyński, 2006); it is interesting to apply these results to the class of HEX programs by using the results obtained in this thesis. Furthermore, from a practical point of view, it will be interesting to implement the introduced semantics and compare them with the standard one. Finally, it will be interesting to apply the machinery of Approximation Theory to a more general class of HEX programs where the head and body of a rule can contain arbitrary formulas (so called *general logic programs* (Bartholomew et al., 2011)).

Bibliography

Apt, K. R., Blair, H. A., and Walker, A. (1988). Towards a theory of declarative knowledge. In Minker, J., editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann Publishers.

Baral, C. (2003). *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, Cambridge.

Bartholomew, M., Lee, J., and Meng, Y. (2011). First-order extension of the FLP stable model semantics via modified circumscription. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, pages 724–730.

Basol, S., Fink, M., Erdem, O., and Ianni, G. (2010). HEX programs with action atoms. In Hermenegildo, M. and Schaub, T., editors, *Technical Communications of the 26th International Conference on Logic Programming, Leibniz International Proceedings in Informatics (LIPIcs)*, pages 24–33. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl.

Brass, S. and Dix, J. (1997). Characterizations of the disjunctive stable semantics by partial evaluation. *The Journal of Logic Programming*, 32(3):207–228.

Brewka, G., Eiter, T., and Truszczyński, M. (2011). Answer set programming at a glance. *Communiications of the ACM*, 54(12):92–103.

Calimeri, F., Faber, W., Pfeifer, G., and Leone, N. (2006). Pruning operators for disjunctive logic programming systems. *Fundamenta Informaticae*, 71(2-3):183–214.

Clark, K. L. (1978). Negation as failure. In Minker, J. and Gallaire, H., editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York.

Davey, B. A. and Priestley, H. A. (2002). *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, 2 edition.

Denecker, M., Marek, V., and Truszczyński, M. (2000a). Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning. In Minker, J., editor, *Logic-Based Artificial Intelligence,* volume 597 of *The Springer International Series in Engineering and Computer Science*, pages 127–144. Kluwer Academic Publishers, Norwell, Massachusetts.

Denecker, M., Marek, V., and Truszczyński, M. (2000b). Uniform semantic treatment of default and autoepistemic logics. In Cohn, A. G., Guinchiglia, F., and Selman, B., editors, *Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 74–84. Morgan Kaufmann Publishers, Los Altos, California.

Denecker, M., Marek, V., and Truszczyński, M. (2002). Ultimate approximations in nonmonotonic knowledge representation systems. In Fensel, D., Guinchiglia, F., McGuinness, D. L., and Williams, M.-A., editors, *Proceedings of the 8th International Conference on Principles of Knowledge Representation and Reasoning (KR 2002)*. Morgan Kaufmann Publishers.

Denecker, M., Marek, V., and Truszczyński, M. (2003). Uniform semantic treatment of default and autoepistemic logics. *Artificial Intelligence*, pages 79–122.

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

Eiter, T., Brewka, G., Dao-Tran, M., Fink, M., Ianni, G., and Krennwallner, T. (2009a). Combining nonmonotonic knowledge bases with external sources. In Ghilardi, S. and Sebastiani, R., editors, *Proceedings of the 7th International Symposium on Frontiers of Combining,* volume 5749 of *Lecture Notes in Computer Science*, pages 18–42. Springer, Trento, Italy.

Eiter, T., Faber, W., Leone, N., and Pfeifer, G. (2000). Declarative problem-solving using the DLV system. In Minker, J., editor, *Logic-Based Artificial Intelligence*, pages 79–103. Kluwer Academic Publishers, Norwell, Massachusetts, USA.

Eiter, T., Gottlob, G., and Mannila, H. (1997). Disjunctive datalog. *ACM Transactions on Database Systems*, 22(3):364–418.

Eiter, T., Ianni, G., and Krennwallner, T. (2009b). Answer set programming: a primer. In *Reasoning Web. Semantic Technologies for Information Systems,* volume 5689 of *Lecture Notes in Computer Science*, pages 40–110. Springer, Heidelberg.

Eiter, T., Ianni, G., Lukasiewicz, T., and Schindlauer, R. (2011). Well-founded semantics for description logic programs in the semantic web. *ACM Transactions on Computational Logic*, 12(2):11:1–11:41.

Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., and Tompits, H. (2008). Combining answer set programming with description logics for the Semantic Web. *Artificial Intelligence*, 172(12-13):1495–1539.

Eiter, T., Ianni, G., Schindlauer, R., and Tompits, H. (2005). A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 90–96.

Eiter, T., Lukasiewicz, T., Schindlauer, R., and Tompits, H. (2004a). Combining answer set programming with description logics for the semantic web. In Dubois, D., Welty, C., and

Williams, M.-A., editors, *Proceedings of the 9th International Conference of the Principles of Knowledge Representation and Reasoning (KR 2004)*, pages 1–11.

Eiter, T., Lukasiewicz, T., Schindlauer, R., and Tompits, H. (2004b). Well-founded semantics for description logic programs in the semantic web. In Antoniou, G. and Boley, H., editors, *Rules and Rule Markup Languages for the Semantic Web,* volume 3323 of *Lecture Notes in Computer Science*, pages 81–97. Springer, Berlin.

Faber, W., Leone, N., and Pfeifer, G. (2004). Recursive aggregates in disjunctive logic programs: semantics and complexity. In Alferes, J. and Leite, J., editors, *Proceedings of the 9th European Conference on Logics in Artificial Intelligence,* volume 3229 of *Lecture Notes in Computer Science*, pages 200–212. Springer, Berlin.

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

Fages, F. (1994). Consistency of Clark's completion and existence of stable models. *Journal of Methods of Logic in Computer Science*, 1(1):51–60.

Fernández, J. A. and Minker, J. (1995). Bottom-up computation of perfect models for disjunctive theories. *The Journal of Logic Programming*, 25(1):33–51.

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

Fitting, M. (1991). Bilattices and the semantics of logic programming. *The Journal of Logic Programming*, 11(2):91–116.

Fitting, M. (1994). Tableaux for logic programming. *Journal of Automated Reasoning*, 13(2):175–188.

Fitting, M. (2002). Fixpoint semantics for logic programming - a survey. *Theoretical Computer Science*, 278(1-2):25–51.

Gebser, M., Kaufmann, B., and Schaub, T. (2012). Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence*, 187-188(C):52–89.

Gelfond, M. and Lifschitz, V. (1988). The stable model semantics for logic programming. In Kowalski, R. and Bowen, K., editors, *Proceedings of the 5th International Conference and Symposium on Logic Programming (ICLP/SLP 1988)*, pages 1070–1080. MIT Press, Cambridge.

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

Ginsberg, M. L. (1988). Multivalued logics: a uniform approach to reasoning in artificial intelligence. *Computational Intelligence*, 4(3):265–316.

Giunchiglia, E., Lierler, Y., and Maratea, M. (2006). Answer set programming based on propositional satisfiability. *Journal of Automated Reasoning*, 36(4):345–377.

Heymans, S. and Toma, I. (2008). Ranking services using fuzzy HEX programs. In Calvanese, D. and Lausen, G., editors, *Proceedings of the 2nd International Conference on Web Reasoning and Rule Systems (RR 2008)* volume 5341 of *Lecture Notes in Computer Science*, pages 181–196. Springer-Verlag, Berlin/Heidelberg.

Hinman, P. G. (2005). *Fundamentals of Mathematical Logic*. A K Peters Ltd., Wellesley, Massachusetts.

Hitzler, P. and Wendt, M. (2005). A uniform approach to logic programming semantics. *Theory and Practice of Logic Programming*, 5(1–2):93–121.

Hoehndorf, R., Loebe, F., Kelso, J., and Herre, H. (2007). Representing default knowledge in biomedical ontologies: application to the integration of anatomy and phenotype ontologies. *BMC Bioinformatics*, 8(1):377.

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

Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., and Scarcello, F. (2006). The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 7(3):499–562.

Lifschitz, V. (2002). Answer set programming and plan generation. *Artificial Intelligence*, 138:39–54.

Lifschitz, V. (2010). Thirteen definitions of a stable model. In Blass, A., Dershowitz, N., and Reisig, W., editors, *Fields of Logic and Computation*, pages 488–503. Springer-Verlag, Berlin.

Lloyd, J. W. (1987). *Foundations of Logic Programming*. Springer-Verlag, Berlin, 2 edition.

Lobo, J., Minker, J., and Rajasekar, A. (1992). *Foundations of Disjunctive Logic Programming*. The MIT Press, Cambridge.

Marek, V., Niemelä, I., and Truszczyński, M. (2004). Logic programs with monotone cardinality atoms. In Lifschitz, V. and Niemelä, I., editors, *Proceedings of the 7th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2004),* volume 2923 of *Lecture Notes in Computer Science*, pages 154–166. Springer-Verlag, Berlin.

Marek, V. and Truszczyński, M. (1999). Stable models and an alternative logic programming paradigm. In Apt, K. R., Marek, V., Truszczyński, M., and Warren, D. S., editors, *The Logic Programming Paradigm: a 25-Year Perspective*, pages 375–398. Springer, Berlin.

McCarthy, J. (1959). Programs with common sense. In *Proceedings of the Symposium of the National Physical Laboratory on the Mechanisation of Thought Processes*, pages 77–84. Defense Technical Information Center.

Minker, J. and Rajasekar, A. (1990). A fixpoint semantics for disjunctive logic programs. *The Journal of Logic Programming*, 9(1):45–74.

Moore, R. C. (1985). Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25:75–94.

Niemelä, I., Simons, P., and Soininen, T. (1999). Stable model semantics of weight constraint rules. In Gelfond, M., Leone, N., and Pfeifer, G., editors, *Proceedings of the 5th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 1999),* volume 1730 of *Lecture Notes in Computer Science*, pages 317–331. Springer-Verlag, Berlin.

Pelov, N. (2004). *Semantics of logic programs with aggregates*. PhD thesis, Katholieke Universiteit Leuven, Leuven.

Pelov, N. and Truszczyński, M. (2004). Semantics of disjunctive programs with monotone aggreggates - an operator-based approach. In Delgrande, J. P. and Schaub, T., editors, *Proceedings of the 10th International Workshop on Non-Monotonic Reasoning (NMR 2004)*, pages 327–334.

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

Reiter, R. (1980). A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81–132.

Seipel, D., Minker, J., and Ruiz, C. (1997). Model generation and state generation for disjunctive logic programs. *The Journal of Logic Programming*, 32(1):49–69.

Shen, Y.-D. (2011). Well-supported semantics for description logic programs. In Walsh, T., editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, pages 1081–1086. AAAI Press, California.

Shen, Y.-D. and Wang, K. (2012). FLP Semantics without circular justifications for general logic programs. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI 2012)*.

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

Smyth, M. B. (1978). Power domains. *Journal of Computer and System Sciences*, 16:23–36.

Tarski, A. (1955). A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309.

Truszczyński, M. (2006). Strong and uniform equivalence of nonmonotonic theories – an algebraic approach. *Annals of Mathematics and Artificial Intelligence*, 48(3-4):245–265.

Van Emden, M. and Kowalski, R. (1976). The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742.

Van Gelder, A., Ross, K. A., and Schlipf, J. S. (1991). The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):619–649.

Wang, K. and Zhou, L. (2003). Comparisons and computations of well-founded semantics for disjunctive logic programs. *ACM Transactions of Computational Logic*, 6(2):295–327.