International Center for Computational Logic,
TU Dresden,
01062 Dresden, Germany

Logic Programs and Three-Valued Consequence Operators

Master's Thesis

Carroline Dewi Puspa Kencana Ramli
Author

Prof. Dr. rer. nat. habil. Steffen Hölldobler
Supervisor

Dresden, Germany August, 2009

Logic Programs and Three-Valued Consequence Operators

Master's Thesis

Carroline Dewi Puspa Kencana Ramli

International Center for Computational Logic,
TU Dresden,
01062 Dresden, Germany

Study Programme: Computational Logic

Prof. Dr. rer. nat. habil. Steffen Hölldobler

Dresden, Germany, August, 2009

Declaration

Author	: Carroline Dewi Puspa Kencana Ramli
Matrikel-Nr	: 3458561
Title	: Logic Programs and Three-Valued Consequence Operators
Degree	: Master of Science
Date of Submission	: 6th of August 2009

I hereby declare that this thesis is my own work, only with the help of the referenced literature and under the careful supervision of my thesis supervisor.

Dresden, Germany, August, 2009 Carroline Dewi Puspa Kencana Ramli

Acknowledgements

First of all, I would like to express my deepest gratitude and appreciation to my supervisor, Prof. Steffen Hölldobler, whose help, stimulating suggestions and encouragement helped me in all the time of research for and writing of this thesis. Through the collaboration in several papers he taught me how to do scientific research and also contributed greatly to improving my English. This thesis is partly a joint work with him that was published in two papers, first one on ICLP 2009 with the title "Logic Programs under Three-Valued Łukasiewicz's Semantics" [HR09a] and the second one on ICANN 2009 with the title "Logics and Networks for Human Reasoning" [HR09b].

I would like to thank the IJCAI 2009 organization and Master of Computational Logic, TUD, for the grant they gave to me to attend the IJCAI 2009 and ICLP 2009 in Pasadena, Los Angeles, United States. I got a lot of experience in the international scientific research community. I also had a chance to present my work at the ICLP 2009 conference as well as receive valuable input from the audience.

I wish to express my special thanks to Prof. Yohanes Stefanus from Faculty of Computer Science, University of Indonesia. He is the one who first introduced Computational Logic to me and he also motivated me to pursue my master degree in this field. Not only that, he has helped me with this thesis by giving comments and suggestions. Until the end, I am indebted to him for his patience with reading my final draft.

I thank Martin Slota for his time and patience to read my thesis, comment and correct my English. I am thankful for his kindness in checking the proofs and giving detailed comments about my work. I have learned a lot from his way of working. I am also heavily indebted for all his encouragement and cheerfulness, especially when I had a difficult time.

In addition, I would like to thank Prof. Luís Moniz Pereira from Universidade Nova de Lisboa who has always motivated me to be a better person and a better student. He has helped me a lot during my early stage in this master, moreover, he gave me a chance to pursue my master thesis at TUD.

I would like to thank the secretaries as well as administrators at UNL, Mrs. Sandra, Mrs. Filipa, Mrs. Gracinda, and at TUD, Mrs. Sylvia, Ms. Julia, for all their help during my study. I specially thank Mrs. Sylvia and Ms. Julia for their vital help, from the beginning of my EMCL application up to the very end of arranging for my thesis defense.

Also, many thanks should go to the European Master in Computational Logic programme and its director, Prof. Steffen Hölldobler, for their support and help during my study. I thank the European Commission for their Erasmus Mundus scholarship to support my study.

I also thank my friends, Mr. Han The Anh, for his help and discussion about the TUD thesis regulations and Mr. Fauzan Tanwil and Ms. Milka Hutagalung for their support in my study and my life, at TUD. I would also to thank Mr. Ari Saptawijaya for his comments in the early draft of my thesis.

Finally, I wish to express very special thanks to my family, who has always been patient and heartedly supported me, listened to me and given good advice, in both my life and my study. Without their encouragement, I would not have been able to finish my thesis.

Abstract

While classical logic is considered not expressive enough to model human reasoning, three-valued logic seems much better suited for this purpose. In [SvL08] Stenning and van Lambalgen show that their consequence operator under Fitting three-valued semantics can appropriately model human reasoning. Their operator is defined similarly to the Fitting operator which has been studied extensively. Even though their definitions and usage are very similar, it turns out that some of their properties are fundamentally different. Thus, in this thesis we deepen the knowledge about the Stenning and van Lambalgen operator, providing formal grounds for further investigation of relations between human reasoning and logic.

First we look for conditions under which both operators are continuous and when they acquire the property of being a contraction. We also introduce a level mapping characterisation of the new operator that puts it within the same framework with other three-valued semantics for logic programs, including the Fitting and well-founded semantics. Then we turn our attention to the underlying three-valued logic used to characterise these operators, dubbed the Fitting semantics. We will see that under this semantics, the model of completed program is not necessarily a model of the program itself. This happens because under Fitting semantics, the law of equivalence does not hold. We show that the Łukasiewicz semantics is a good candidate to replace Fitting semantics since it admits the law of equivalence while not changing the meaning or properties of logic programs. Further, we present the core method, a connectionist model generator for logic programs, that can easily be adapted to handle Stenning and van Lambalgen's approach. Finally, since under the new operator negative information is difficult to express in the program, we propose a number of approaches to add this kind of expressivity to the formalism.

Keywords: logic programming, consequence operator, three-valued logics, Łukasiewicz semantics, Kleene semantics, Fitting semantics, Fitting operator, Stenning and van Lambalgen operator, human modelling, cognitive science, core method

Contents

List of Tables

List of Figures

Chapter 1

Introduction

It has been widely argued in the field of Cognitive Science that logic is inadequate for modelling human reasoning (see e.g. Byrne in [Byr89]). In this context, "logic" is meant to be classical logic and, indeed, classical logic fails to capture some well-documented forms of human reasoning. However, many non-classical logics have been studied and widely used in the field of Artificial Intelligence. These logics try to capture many assumptions or features that occur in common sense reasoning like, for example, the closed world assumption [Rei78, Lif85] or non-monotonic reasoning [BG94, NBR02, SvL05].

Recently, Stenning and van Lambalgen [SvL08] have suggested that completed logic programs can adequately model many human reasoning tasks. This is somewhat surprising since the completion of a logic program is simply a classical theory. They describe conditionals as law-like relationships between antecedents and consequents, allowing the antecedent to hide an endless number of unstated assumptions. This implies that there may very well be exceptions to the law but these exceptions do not falsify the law. For example, in case we have a law-like condition "if A then B", it is interpreted as "if A, *and nothing abnormal is the case*, then B", where what is abnormal is provided by the context. In this sense, closed world assumption plays an important part in reasoning with conditionals. Hence, the conditionals will be seen to be due to a special form of a closed world assumption and can be formalized in logic programming. By contrast, Byrne [Byr89] assumed that conditionals can be represented by classical implication.

In their work, Stenning and van Lambalgen used the common three-valued semantics introduced by Fitting [Fit85]. This semantics combines Kleene strong three-valued logic for negation, conjunction, disjunction and implication with complete equivalence, which was also introduced by Kleene [Kle52]. Complete equivalence was used by Fitting to ensure that a formula of the form $F \leftrightarrow F$ is mapped to true under an interpretation which maps F to neither true nor false (see [Fit85], p.300). Under the Fitting semantics, the law of equivalence ($F \leftrightarrow G$ is semantically equivalent to $(F \leftarrow G) \wedge (G \leftarrow F)$) does not hold anymore. This is somewhat surprising as Fitting suggests a completion-based approach [Cla78], where the if-halves of the definitions in a logic program are completed by adding their corresponding only-if-halves. Under the Fitting semantics, a completed definition $p \leftrightarrow q$ may be mapped to true under an interpretation which maps neither $p \leftarrow q$ nor $q \leftarrow p$ to true.

Stenning and van Lambalgen also introduced a new three-valued operator for logic programs that they claim is suitable for modelling human reasoning. Their operator is only slightly different from the one defined by Fitting [Fit85]. We present below two examples to illustrate the difference between the Fitting and the Stenning and van Lambalgen operator.

Suppose we want to model an agent driving a car. One rule would be that he may cross an intersection if the traffic light shows green and there is no unusual situation:

$$cross \leftarrow green, \neg unusual\_situation. \tag{1.1}$$

An unusual situation occurs if an ambulance wants to cross the intersection from a different direction:

$$unusual\_situation \leftarrow ambulance\_crossing. \tag{1.2}$$

In addition, suppose that the green light is indeed on:

$$green \leftarrow \top. \tag{1.3}$$

Let $\mathcal{P}_{crossing}$ be the set of clauses 1.1, 1.2 and 1.3. The least fixed point of Fitting operator for $\mathcal{P}_{crossing}$ is

$$lfp(\Phi_{F,\mathcal{P}_{crossing}}) = \langle \{green, cross\}, \{unusual\_situation, ambulance\_crossing\} \rangle .$$

We see that in the least fixed point, *cross* is mapped to true. Hence, not knowing anything about an ambulance, our agent will assume that no ambulance is present, hit the accelerator, and speed into the intersection. One should observe that not knowing anything about an ambulance may be caused by the fact that the agent's camera is blurred or the agent's microphone is damaged. His assumption that no ambulance is present is made by default. On the other hand,

$$lfp(\Phi_{SvL,\mathcal{P}_{crossing}}) = \langle \{green\}, \emptyset \} \rangle .$$

In this case, *crossing* is neither true nor false. Hence, the agent doesn't know whether he may cross the intersection. Inspecting his rules he may find that in order to satisfy the conditions for the first rule, he must verify that no ambulance is crossing. In doing so, he may extend $\mathcal{P}_{crossing}$ to $\mathcal{P}'_{crossing} = \mathcal{P}_{crossing} \cup \{ambulance\_crossing \leftarrow \bot\}$ yielding

$$lfp(\Phi_{SvL,\mathcal{P}'_{crossing}}) = \langle \{green, cross\}, \{unusual\_situation, ambulance\_crossing\} \rangle .$$

Now, the agent can safely cross the intersection.

The second example is taken from [Byr89]. Byrne has confronted individuals with sentences like

> *If Marian has an essay to write, she will study late in the library.*
>
> *She does not have an essay to write.*
>
> *If she has textbooks to read, she will study late in the library.*

The individuals are then asked to draw conclusions. In this example, only 4% of the individuals conclude that Marian will not study late in the library.

Although Byrne uses these and similar examples to conclude that (classical) logic is inadequate for human reasoning, Stenning and van Lambalgen have argued in [SvL08] that the use of three-valued logic programs under completion semantics is indeed adequate for human reasoning. They represent the scenario by

$$\begin{aligned} \mathcal{P}_{study}: \qquad l &\leftarrow e, \neg ab_1. \\ e &\leftarrow \bot. \\ ab_1 &\leftarrow \bot. \\ l &\leftarrow t, \neg ab_2. \\ ab_2 &\leftarrow \bot. \end{aligned}$$

where l denotes that Marian will study late in the library, e denotes that she has an essay to write, t denotes that she has a textbook to read, and ab denotes abnormality. In this case, we find $lfp(\Phi_{SvL,\mathcal{P}_{study}}) = \langle \emptyset, \{ab_1, ab_2, e\}\rangle$, from which we conclude that it is unknown whether Marian will study late in the library. On the other hand, $lfp(\Phi_{F,\mathcal{P}_{study}}) = \langle \emptyset, \{ab_1, ab_2, e, t, l\}\rangle$. Using the Fitting operator one would conclude that Marian will not study late in the library. Thus, this operator leads to a wrong answer with respect to the discussed scenario from human reasoning, whereas the Stenning and van Lambalgen operator does not.

Further, Stenning and van Lambalgen stated that the least fixed point of their operator is always a model of the program completion and also a minimal model of the program (Lemma 4 (1.) in [SvL08]). This claim are, however, not correct. We found out that the least fixed point may not even be a model of the program [HR09b] and that this stems from the fact that the Fitting semantics does not admit the law of equivalence.

These findings motivated further research that is presented in this thesis. In the first part, we will compare the Fitting operator to the Stenning and van Lambalgen operator. Their definitions and usage are very similar but it will turn out that some of their properties are fundamentally different. Also, while the former was studied extensively in the literature, this is not the case with the latter, so our contribution is also in deepening the knowledge about the new operator. We will look for conditions under which both operators are continuous and also when they acquire the property of being a contraction. We will introduce a level mapping characterisation of the new operator that puts it within the same framework with other semantics examined in [HW02], including the Fitting and the well-founded semantics.

The second part of this thesis is related to the properties of Fitting three-valued semantics. We will show that the Łukasiewicz semantics [Łuk20] may be a good candidate to replace Fitting semantics since it admits the law of equivalence while not changing the meaning or properties of logic programs.

In [HK94], a connectionist model generator for propositional logic programs using recurrent networks with feed-forward core was presented. It was later called the *core method* [BH06]. The core method has been extended and applied to a variety of programs including computing the immediate consequence operators associated with logic programs. Turning the feed-forward core into a recurrent network allows to compute or approximate the least model of a logic program [HS99]. Kalinke has applied the core method to logic programs un-

der the Fitting semantics. In the third part of this thesis, we present the core method for Stenning and van Lambalgen's approach.

Lastly, since under the new operator negative information is difficult to express in the program, we will propose a number of approaches to add this kind of expressivity to the formalism. Their full realization is, however, left for future research.

1.1 Thesis Structure

We begin with the preliminaries for the notation and terminology that we use through the thesis in Chapter 2. Then we start with the first part which is about the relations between the Fitting operator and the Stenning and van Lambalgen operator. In Chapter 3 we discuss about the property of continuity. We continue with the contraction property in Chapter 4. Finally, in Chapter 5 we review the correlation between Fitting semantics, Stenning and van Lambalgen semantics and the well-founded semantics.

In the second part, we discuss Łukasiewicz semantics when used in the context of logic programs. In Chapter 6, we first introduce the background of three-valued logics and then we investigate the possibility of replacing the Fitting semantics by the Łukasiewicz semantics in logic programs and its effects on the properties of fixed points of both operators.

Chapter 7 constitutes the final part of this thesis in which we present the connectionist system for the Stenning and van Lambalgen operator based on the core method.

To lead the future work, we will discuss in Chapter 8 about "negative facts". We will propose a number of definitions for negative facts, compare them and discuss the advantages and disadvantages of each of them. In the final Chapter 9 we summarize our findings and point to some future work.

Chapter 2

Preliminaries

This chapter introduces general notation and terminology that we use throughout this thesis. It is based on [Llo84] with some extensions. Note that these preliminaries are for logic programming in general, whereas the background of more specific topics (e.g. the metric theory and lattice theory) will be given later on when it is needed.

The chapter starts with the introduction of the syntax of well-formed formulae of a first order theory. Then it introduces the more specific syntax of logic programs and continues with the description of a declarative semantics for logic programs. At the end it introduces the definition of Fitting immediate consequence operator and Stenning and van Lambalgen immediate consequence operator.

2.1 First-Order Language

We consider an *alphabet* consisting of finite or countably infinite disjoint sets of constants, function symbols and predicate symbols, an infinite set of variables, the connectives $\neg$, $\vee$, $\wedge$, $\leftarrow$, $\leftrightarrow$, and punctuation symbols "(", "," and ")". Additionally, the alphabet also contains the special symbols $\top$ and $\bot$ denoting a valid and an unsatisfiable formula, respectively.

Next we turn to the definition of the first order language given by an alphabet.

Definition 2.1 (Term)**.** The set of *terms* is the smallest set defined by the following rules:

1. A variable is a term.
2. A constant is a term.
3. If f is an n-ary function symbol ($n \geq 1$) and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.

A *ground term* is a term not containing variables.

We will use upper case letters to denote variables and lower case letters to denote constants, function- and predicate symbols.

Definition 2.2 (Formula)**.** The set of *(well-formed) formulae* is the smallest set defined by the following rules:

1. If p is an n-ary predicate symbol and $t_1, \ldots, t_n$ are terms, then $p(t_1, \ldots, t_n)$ is a formula (called an *atomic formula* or simply an *atom*).
2. If F and G are formulae, then so are $(\neg F)$, $(F \wedge G)$, $(F \vee G)$, $(F \leftarrow G)$ and $(F \leftrightarrow G)$.

A *propositional formula* is a formula where all predicate symbols are of arity 0. A *ground formula* is a formula with every term grounded. A *literal* is an atom A or its negation $\neg A$.

To avoid having formulae cluttered with brackets, we adopt the following precedence hierarchy to order the connectives:

$$\neg > \{ \vee , \wedge \} > \leftarrow > \leftrightarrow$$

We are now ready to define a language as follows:

Definition 2.3 (Language)**.** A *language* $\mathcal{L}$ given by an alphabet $\mathcal{A}$ consists of the set of all formulae constructed from the symbols of $\mathcal{A}$.

2.2 Syntax of Logic Programs

A logic program is a declarative, relational style of programming based on first-order logic. In this section, we define the syntax of logic programs.

Definition 2.4 (Clause)**.** A *(program) clause* is a formula of the form

$$A \leftarrow B_1 \wedge \cdots \wedge B_m \quad (m \geq 1)$$

A is an atom, and each $B_i,\ 1 \leq i \leq m$, is either a literal or $\top$. A is called the *head* and $B_1 \wedge \cdots \wedge B_m$ the *body* of the clause. We usually write $B_1 \wedge \cdots \wedge B_m$ simply as $B_1, \ldots, B_m$ and we finish each clause with a dot as in Prolog.

We define a *definite clause* as a clause where every $B_i,\ 1 \leq i \leq m$, is either an atom or $\top$.

One should observe that the body of a clause must not be empty. A *positive fact* is a clause of the form

$$A \leftarrow \top.$$

In addition, Stenning and van Lambalgen allow for so-called *negative facts* of the form

$$A \leftarrow \bot.$$

Definition 2.5 (Logic Program)**.** A *(logic) program* $\mathcal{P}$ is a finite set of clauses. We say a program is *definite* if all of its clauses are definite. Moreover, an *extended (logic) program* is a finite set of clauses and negative facts. A *propositional program* is a program where all clauses are propositional. A program is *ground* if all its clauses are ground.

We assume that each non-propositional program contains at least one constant symbol. Moreover, the language $\mathcal{L}$ underlying a program $\mathcal{P}$ shall contain precisely the predicate, function and constant symbols occurring in $\mathcal{P}$, and no others.

2.3 Semantics of Logic Programs

The declarative semantics of a logic program is given by a model-theoretic semantics of formulae in the underlying language. This sections discusses interpretations and models, concentrating particularly on the important class of Herbrand interpretations.

Definition 2.6 (Herbrand Universe)**.** The *Herbrand universe* $\mathcal{U}_\mathcal{L}$ for a language $\mathcal{L}$ is the set of all ground terms that can be formed from the constants and function symbols appearing in $\mathcal{L}$. By $\mathcal{U}_\mathcal{P}$ we denote the Herbrand universe for the language underlying the program $\mathcal{P}$.

Definition 2.7 (Ground Instance)**.** A *ground instance* of a formula F is any ground formula that results from F by substituting all variables by terms in $\mathcal{U}_\mathcal{L}$. We denote by *ground*($\mathcal{P}$) the set of all ground instances of clauses in program $\mathcal{P}$.

In many cases, *ground*($\mathcal{P}$) is infinite. In the sequel, we will consider *ground*($\mathcal{P}$) as a substitute for $\mathcal{P}$, thus ignoring unification issues.

Definition 2.8 (Herbrand Base)**.** The *Herbrand base* $\mathcal{B}_\mathcal{L}$ for a language $\mathcal{L}$ is the set of all ground atoms that can be formed by using predicate symbols from $\mathcal{L}$ and ground terms from $\mathcal{U}_\mathcal{L}$ as arguments. By $\mathcal{B}_\mathcal{P}$ we denote the Herbrand base for the language underlying the program $\mathcal{P}$.

Since in this work we only use Herbrand interpretations, we drop the qualification "Herbrand". Also, we are mainly concerned with three-valued logics, so we will introduce the notion of a three-valued interpretation and a three-valued model for logic programs.

Definition 2.9 (Interpretation)**.** An *interpretation* I of a program $\mathcal{P}$ is a mapping from the Herbrand base $\mathcal{B}_\mathcal{P}$ to the set of truth values $\{\top, \bot, u\}$. We represent interpretations by pairs $\langle I^\top, I^\bot \rangle$, where the set $I^\top$ contains all atoms which are mapped to $\top$, the set $I^\bot$ contains all atoms which are mapped to $\bot$, and $I^\top \cap I^\bot = \emptyset$. All atoms which occur neither in $I^\top$ nor $I^\bot$ are mapped to u. We say an interpretation I is *total* if $I^\top \cup I^\bot = \mathcal{B}_\mathcal{P}$.

The truth value of arbitrary formulae under some interpretation can be determined from a truth table as usual. We give in Table 2.1 the truth values for connectives that correspond to the logic that Fitting used in [Fit85]. Later in Chapter 6 we will consider other three-valued logics and compare the properties of logic programs and operators with respect to them.

Definition 2.10 (Truth Value of a Formula)**.** The logical value of ground formulae can be derived from Table 2.1 in the usual way. A formula F is then *true under interpretation* I, denoted by $I(F) = \top$, if all its ground instances are true in I; it is *false under* I, denoted by $I(F) = \bot$, if there is a ground instance of F that is false in I; and otherwise it is *undefined under* I, denoted by $I(F) = u$.

Two formulae F and G are said to be *semantically equivalent* denoted by $F \equiv G$ if F and G have same truth value under all interpretations.

Definition 2.11 (Model)**.** Let I be an interpretation. I is a *model* of a formula F if $I(F) = \top$. For a program $\mathcal{P}$, we say I is a model of $\mathcal{P}$ if I is a model for every clause in $\mathcal{P}$. Similarly, for a set of clauses $\mathcal{C} \subseteq \mathcal{P}$, we say I is a model of $\mathcal{C}$ if I is a model for every clause in $\mathcal{C}$.

Table 2.1: Truth values for three-valued logic

	$\neg$
$\top$	$\bot$
$\bot$	$\top$
u	u

		$\wedge$	$\vee$	$\rightarrow$	$\leftrightarrow$
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$
$\bot$	$\top$	$\bot$	$\top$	$\top$	$\bot$
u	$\top$	u	$\top$	$\top$	$\bot$
$\top$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\top$
u	$\bot$	$\bot$	u	u	$\bot$
$\top$	u	u	$\top$	u	$\bot$
$\bot$	u	$\bot$	u	$\top$	$\bot$
u	u	u	u	u	$\top$

An information ordering among interpretations will also be useful in our further considerations.

Definition 2.12 (Ordering Among Interpretations)**.** Let $I = \langle I^\top, I^\bot \rangle$ and $J = \langle J^\top, J^\bot \rangle$ be two interpretations. We write $I \subset J$ if and only if $I^\top \subset J^\top$ and $I^\bot \subset J^\bot$ and we write $I \subseteq J$ if and only if $I^\top \subseteq J^\top$ and $I^\bot \subseteq J^\bot$. If $\mathcal{I}$ is a collection of interpretations, then an interpretation I in $\mathcal{I}$ is called *minimal* in $\mathcal{I}$ if and only if there is no interpretation J in $\mathcal{I}$ such that $J \subset I$. An interpretation I is called *least* in $\mathcal{I}$ if and only if $I \subseteq J$ for any interpretation J in $\mathcal{I}$. A model M of a program $\mathcal{P}$ is called minimal (resp. least) if it is minimal (resp. least) among all models of $\mathcal{P}$.

2.4 Program Completion

Usually a logic program is assumed to contain complete information in the sense that anything not stated is false. Therefore, we derive $\neg A$ from failure to derive A which is often called *negation as failure*. To handle negation as failure in a logic program, Clark [Cla78] introduces program completion, in which, loosely speaking, "if" is interpreted as "if and only if".

Definition 2.13 (Program Completion)**.** Let $\mathcal{P}$ be a logic program. Consider the following transformation:

1. Replace all clauses in *ground*($\mathcal{P}$) with the same head (ground atom) $A \leftarrow Body_1,\ A \leftarrow Body_2, \ldots$ by the single expression $A \leftarrow Body_1 \vee Body_2 \vee \cdots$.
2. If a ground atom A is not the head of any clause in *ground*($\mathcal{P}$) then add $A \leftarrow \bot$.
3. Replace all occurrences of $\leftarrow$ by $\leftrightarrow$.

The resulting set of formulae is called *completion of* $\mathcal{P}$ and is denoted by *comp*($\mathcal{P}$). One should observe that in step 1 there may be infinitely many clauses with the same head resulting in a countable disjunction. However, its semantic behavior is unproblematic.

2.5 Consequence Operators

Let $\mathcal{P}$ be a logic program. A *consequence operator* of $\mathcal{P}$ is a function which maps an interpretation of $\mathcal{P}$ to another interpretation of $\mathcal{P}$. It expresses the consequences of $\mathcal{P}$ when the bodies of its clauses are interpreted under the input interpretation. In [Fit85], Fitting has defined an immediate consequence operator as follows.

Definition 2.14 (Fitting Immediate Consequence Operator)**.** Let I be an interpretation and $\mathcal{P}$ a program. The *Fitting immediate consequence operator* is defined as follows: $\Phi_{F,\mathcal{P}}(I) = \langle J^\top, J^\bot \rangle$, where

$$J^\top = \{\, A \mid \text{there exists } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ with } I(\mathit{Body}) = \top \,\} \text{ and}$$
$$J^\bot = \{\, A \mid \text{for all } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ we find } I(\mathit{Body}) = \bot \,\}.$$

In their quest for models of human reasoning, Stenning and van Lambalgen [SvL08] have introduced an immediate consequence operator for propositional programs, which differs slightly from the Fitting operator. Here, we extend the operator to first-order programs.

Definition 2.15 (Stenning and van Lambalgen Immediate Consequence Operator)**.** Let I be an interpretation and $\mathcal{P}$ be an extended program. The *Stenning and van Lambalgen immediate consequence operator* is defined as follows: $\Phi_{SvL,\mathcal{P}}(I) = \langle J^\top, J^\bot \rangle$, where

$$J^\top = \{\, A \mid \text{there exists } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ with } I(\mathit{Body}) = \top \,\} \text{ and}$$
$$J^\bot = \{\, A \mid \text{there exists } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ and}$$
$$\text{for all } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ we find } I(\mathit{Body}) = \bot \,\}$$

The difference from the Fitting operator is only in the first line of the definition of $J^\bot$.

Chapter 3

Continuity Property

Fixed points play a fundamental role in several areas of computer science and logic for justifying induction and recursive definitions. The theory of fixed points is concerned with the conditions which guarantee that a mapping $F : X \to X$ of a set X into itself admits one or more fixed points, that is, points $x \in X$ for which $F(x) = x$. In the area of logic programming, many semantics of logic programs are determined by a fixed point equation in which the domain and the range of the operator is the set of interpretations.

We will start by introducing in Section 3.1 some basic notions of Order Theory that studies various order relations on sets as well as the behaviour of mappings on these sets. We will use two of important results of this theory concerning a specific type of ordered sets, dubbed a *complete partial order*. In Section 3.2 we will show that the space of (three-valued) interpretations on which our consequence operators operate satisfies the conditions for being a complete partial order (with respect to the $\subseteq$ ordering).

The first result of Order Theory we will in our context use is the Knaster-Tarski Fixed Point Theorem which states that each monotonic mapping on a complete partial order has a least fixed point. In Sections 3.3 and 3.4 we will show that the Fitting and Stenning and van Lambalgen operator, respectively, is indeed monotonic and hence is guaranteed to have a least fixed point.

The second result of Order Theory we will use is the Kleene Fixed Point Theorem which states that the least fixed point of a continuous mapping can be computed by iterating the operator up to the first limit ω times starting from the empty interpretation $\langle \emptyset, \emptyset \rangle$. Continuity is a strictly stronger condition than monotonicity and neither of the operators satisfies it in general. For the Fitting operator this was shown in [Fit85] and we show the same for the Stenning and van Lambalgen operator.

Finally, for complete partial orders continuity is equivalent to monotonicity, so in Section 3.5 we study the case when the interpretation space is finite. This allows us to conclude that both the Fitting and Stenning and van Lambalgen operators for a propositional program are continuous. Then we generalize this claim to ground programs for which the space of interpretations may be infinite if they contain a function symbol. We provide a proof based on a transformation to propositional programs.

3.1 Order Theory

This section introduces the prerequisite concepts and results of Order Theory that we will use later on. The reader can refer to [Llo84, DP02] for a more complete and elaborate study of these topics with many further references.

Definition 3.1 (Relation)**.** Let S be a set. A *(binary) relation* R on S is a subset of $S \times S$. We will use the infix notation xRy to denote $(x, y) \in R$.

Definition 3.2 (Partial Order)**.** A relation $\leq$ on a set S is a *partial order* if the following conditions are satisfied:

1. Reflexivity: $(\forall x \in S)(x \leq x)$.
2. Antisymmetry: $(\forall x, y \in S)((x \leq y) \wedge (y \leq x) \Rightarrow (x = y))$.
3. Transitivity: $(\forall x, y, z \in S)((x \leq y) \wedge (y \leq z) \Rightarrow (x \leq z))$.

We also say that S is a *partially ordered set* with respect to the partial order $\leq$.

Partially ordered sets are very common in mathematics. For example, given a set S, the set of all subsets of S, denoted by 2^S, is a partial order with respect to the set inclusion $\subseteq$. Upper and lower bounds also play important roles in Order Theory.

Definition 3.3 (Upper Bound and Lower Bound)**.** Let S be a set with a partial order $\leq$. Then $a \in S$ is an *upper bound* of a subset X of S if for every $x \in X$ we have $x \leq a$. Similarly, $b \in S$ is a *lower bound* of X if for every $x \in X$ we have $b \leq x$.

Definition 3.4 (Least Upper Bound and Greatest Lower Bound)**.** Let S be a set with a partial order $\leq$. We say $a \in S$ is the *least upper bound* of a subset X of S if a is an upper bound of X and for every upper bound a' of X we have $a \leq a'$. If it exists, we denote the least upper bound of X by $lub(X)$.

Similarly, $b \in S$ is the *greatest lower bound* of a subset X of S if b is a lower bound of X and for every lower bound b' of X we have $b' \leq b$. If it exists, we denote the greatest lower bound of X by $glb(X)$.

Note that if it exists, the least upper bound of X is unique. This is because if two least upper bounds a_1, a_2 of X existed, then a_1, a_2 are upper bounds of X and $a_1 \leq a_2$ and $a_2 \leq a_1$. Consequently, by antisymmetry, $a_1 = a_2$. For the same reasons, if it exists, the greatest lower bound of X is unique.

We are now almost ready to introduce the notion of a complete partial order. The final missing piece is the definition of a directed set which will also be needed to define the property of continuity.

Definition 3.5 (Directed Set)**.** Let S be a partially ordered set and X be a non-empty subset of S. X is *directed* if for every $x, y \in X$ there exists some $z \in X$ such that $x \leq z$ and $y \leq z$.

Definition 3.6 (Complete Partial Order)**.** A partially ordered set C is a *complete partial order* if

1. C has the least element and

2. for every directed subset X of C there exists the least upper bound of X in C.

As mentioned above, the next section will demonstrate that the space of three-valued interpretations with respect to the $\subseteq$ ordering forms a complete partial order.

We now introduce the monotonicity and continuity property for mappings on partial orders.

Definition 3.7 (Monotonic Mapping)**.** Let S be partially ordered set and $f : S \to S$ be a mapping. We say f is *monotonic* if for every $x, y \in S$ such that $x \leq y$ we have $f(x) \leq f(y)$.

Definition 3.8 (Continuous Mapping)**.** Let S be partially ordered set and $f : S \to S$ be a mapping. We say f is *continuous* if for every directed subset X of S we have $f(\mathit{lub}(X)) = \mathit{lub}(f(X))$ where $f(X) = \{\, f(x) \mid x \in X \,\}$.

The following proposition shows that the set of continuous mappings is included in the set of monotonic mappings.

Proposition 3.9. Every continuous mapping is monotonic.

Proof. Consider a continuous mapping $f : S \to S$ and some $x, y \in S$ such that $x \leq y$. Then the set $X = \{\, x, y \,\}$ is a directed subset of S and by continuity of f we obtain

$$\sup(f(X)) = f(\sup(X))$$

Since $\sup(X) = y$, we further obtain $\sup(\{\, f(x), f(y) \,\}) = f(y)$ and consequently $f(x) \leq f(y)$ as desired. □

Now we are ready to formulate the two important results of Order Theory relevant to this work. The first is the Knaster-Tarski Fixed Point Theorem which ensures that every monotonic mapping has a least fixed point.

Theorem 3.10 (Knaster-Tarski Fixed Point Theorem)**.** Let C be a complete partial order and f be a monotonic mapping on C. Then f has a least fixed point.

Proof. See [DP02] page 187 Theorem 8.22. □

Furthermore, monotonicity enables us to characterize the least fixed point of f using transfinite induction as follows:

Proposition 3.11. Let C be a complete partial order with the least element $\bot$, f be a monotonic mapping on C, x be the least fixed point of f and

$$\begin{aligned} x_0 &= \bot \ , \\ x_\alpha &= f(x_{\alpha-1}) \text{ for every non-limit ordinal } \alpha > 0 \ , \\ x_\alpha &= \mathit{lub}\{\, x_\beta \mid \beta < \alpha \,\} \text{ for every limit ordinal } \alpha \ . \end{aligned}$$

Then for some ordinal γ we find $x = x_\gamma$.

Proof. First we prove by transfinite induction that for every ordinal α we have $x_\alpha \leq x$:

1° For $\alpha = 0$ the claim follows by the definition of $\bot$.

2° Let α be an ordinal such that the claim holds for all $\beta < \alpha$. We will consider two cases:

a) If α is a non-limit ordinal, then $x_\alpha = f(x_{\alpha-1})$ and by the inductive assumption we have $x_{\alpha-1} \leq x$. Hence, by the monotonicity of f we directly obtain

$$x_\alpha = f(x_{\alpha-1}) \leq f(x) = x \ .$$

b) If α is a limit ordinal, then $x_\alpha = \mathit{lub}\{\, x_\beta \mid \beta < \alpha \,\}$. Further, for every $\beta < \alpha$ we have $x_\beta \leq x$ and hence x is an upper bound of the set of the set $\{\, x_\beta \mid \beta < \alpha \,\}$. Consequently, since x_α is the least upper bound of that set, we obtain $x_\alpha \leq x$.

Further, it cannot be the case that $x_\alpha < x$ for all ordinals α, since then C would have a higher cardinality than any ordinal. □

The second important result is the Kleene Fixed Point Theorem which guarantees that for every continuous mapping, the least fixed point can be computed by iterating the mapping up to ω times starting from the least element of the complete partial order.

Theorem 3.12 (Kleene Fixed Point Theorem)**.** Let C be a complete partial order with the least element $\bot$ and f be a continuous mapping on C. Then the least fixed point of f is $\mathit{lub}(\{\, f^n(\bot) \mid n \geq 0 \,\})$.

Proof. See [DP02] page 183 Theorem 8.15 (ii). □

Finally, we turn to the situation when the complete partial order is finite. First we show that the pairwise upper bound property of a directed set X can be extended to finite subsets of X. From this is also follows that every finite directed set contains its own least upper bound.

Lemma 3.13. Let X be a directed set and Y be a finite subset of X. Then X contains an upper bound of Y.

Proof. Suppose $Y = \{\, y_1, y_2, \ldots, y_n \,\}$. Then we can construct a sequence $\{x_i\}_{i=2}^n$ of elements of X such that

$$\begin{aligned} &y_1 \leq x_2 \text{ and } y_2 \leq x_2 \ ; \\ &y_i \leq x_i \text{ and } x_{i-1} \leq x_i \qquad \text{for each } i \in \{\, 3, 4, \ldots, n \,\}\,. \end{aligned}$$

By induction on i it follows that $x_i \leq x_n$ for every $i \in \{\, 2, 3, \ldots, n \,\}$ and by applying transitivity we obtain $y_i \leq x_n$ for each $i \in \{\, 1, 2, \ldots, n \,\}$. Hence, x_n is an upper bound of Y in X. □

Corollary 3.14. Any finite directed set contains its own least upper bound.

Proof. Let X be a finite directed set. Then by Lemma 3.13 it contains its own upper bound x. Consider some other upper bound y of X. Then since $x \in X$, we have $x \leq y$ and so x is also the least upper bound of X. □

The previous observations enable us to show that continuity is equivalent to monotonicity in the case of a finite complete partial order. Consequently, the conclusion of the Kleene Fixed Point Theorem can be applied to any monotonic mapping on a finite complete partial order.

Proposition 3.15. Let C be a finite complete partial order and f be a monotonic mapping on C. Then f is continuous.

Proof. Let $X \subseteq C$ be some directed set. We need to prove that $lub(f(X)) = f(lub(X))$. Since C is finite, X must also be finite and so by Corollary 3.14 X must contain its own least upper bound $lub(X) = x \in X$. So we now need to prove that

$$lub(f(X)) = lub(\{\, f(y) \mid y \in X \,\}) = f(x) \ .$$

To show that $f(x)$ is an upper bound of $f(X)$, take some $y \in X$. We have $y \leq x$ because x is the least upper bound of X. Hence $f(y) \leq f(x)$ by the monotonicity of f and we are done.

It remains to show that $f(x)$ is the least upper bound of $f(X)$. But since $f(x) \in f(X)$, for any upper bound x_0 of $f(X)$ we immediately obtain $f(x) \leq x_0$. This observation finishes our proof. □

Corollary 3.16. Let C be a finite complete partial order with the least element $\bot$ and f be a monotonic mapping on C. Then the least fixed point of f is $lub(\{\, f^n(\bot) \mid n \geq 0 \,\})$.

Proof. Follows from Proposition 3.15 and Theorem 3.12. □

3.2 Complete Partial Order of Interpretations

In this section we will show that the set of all interpretations forms a complete partial order with respect to set inclusion $\subseteq$. In the following, whenever X is some set of interpretations, we will use the following notation:

$$X^\top = \{\, I^\top \mid \langle I^\top, I^\bot \rangle \in X \,\}$$
$$X^\bot = \{\, I^\bot \mid \langle I^\top, I^\bot \rangle \in X \,\}$$

The next result shows that the least upper bound of a directed set of interpretations always exists.

Proposition 3.17. Let X be a directed set of interpretations. Then the interpretation $I = \langle \bigcup X^\top, \bigcup X^\bot \rangle$ is the least upper bound of X.

Proof. First we will prove that I is an interpretation. Suppose it is not. Then there is some atom A such that $A \in \bigcup X^\top$ and also $A \in \bigcup X^\bot$. But then for some $J_1, J_2 \in X$ we have $A \in J_1^\top$ and $A \in J_2^\bot$. Further, since X is directed, it contains some interpretation J_3 such that $J_1 \subseteq J_3$ and $J_2 \subseteq J_3$. It follows that $A \in J_3^\top$ and $A \in J_3^\bot$, which contradicts the fact that J_3 is an interpretation.

We will now show that I is an upper bound of X. This follows easily from the definition of I: for any $J \in X$ we have both $J^\top \subseteq \bigcup X^\top$ and $J^\bot \subseteq \bigcup X^\bot$. Hence, $J \subseteq I$.

It remains to show that I is also the least upper bound of X. Suppose to the contrary that I_0 is some upper bound of X such that $I \not\subseteq I_0$. Then for some atom A we have $I(A) \neq u$ and $I(A) \neq I_0(A)$. Assume without loss of generality that $I(A) = \top$ (the case $I(A) = \bot$ follows by symmetric considerations). Then $A \in \bigcup X^\top$, so for some $J \in X$ we have $A \in J^\top$. But then $J \not\subseteq I_0$ contrary to the fact that I_0 is an upper bound of X. This contradiction finishes our proof. □

As a Corollary we obtain that the set of interpretations is a complete partial order. Notice it is not a complete lattice since it has no greatest element (but many maximal elements, e.g. $\langle \mathcal{B}_\mathcal{P}, \emptyset \rangle, \langle \emptyset, \mathcal{B}_\mathcal{P} \rangle, \dots$)

Corollary 3.18. The set of all interpretations $\mathcal{I}$ is a complete partial order with respect to the set inclusion $\subseteq$.

Proof. The reflexivity, antisymmetry and transitivity of set inclusion follow from the basic properties of sets. The least element of $\mathcal{I}$ is $\langle \emptyset, \emptyset \rangle$ and by the previous Proposition every directed subset of $\mathcal{I}$ has the least upper bound in $\mathcal{I}$. Hence, $\mathcal{I}$ is a complete partial order with respect to $\subseteq$. □

3.3 Fitting Operator

Let us first recall again the definition of Fitting immediate consequence operator:

Let I be an interpretation and $\mathcal{P}$ a program. The *Fitting immediate consequence operator* is defined as follows: $\Phi_{F,\mathcal{P}}(I) = \left\langle J^\top, J^\bot \right\rangle$, where

$$J^\top = \{\, A \mid \text{there exists } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ with } I(\mathit{Body}) = \top \,\} \text{ and}$$
$$J^\bot = \{\, A \mid \text{for all } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ we find } I(\mathit{Body}) = \bot \,\}.$$

In the previous section we proved that the set of interpretations is a complete partial order. In this section we will investigate whether the Fitting operator is in general monotonic or even continuous. We show in the following proposition that the Fitting operator is monotonic and so by Knaster-Tarski Fixed Point Theorem (Theorem 3.10), it has a least fixed point.

Proposition 3.19. Let $\mathcal{P}$ be a program. Then $\Phi_{F,\mathcal{P}}$ is a monotonic mapping.

Proof. Let I, J be interpretations such that $I \subseteq J$. We need to show that $\Phi_{F,\mathcal{P}}(I) \subseteq \Phi_{F,\mathcal{P}}(J)$.

Suppose $A \in (\Phi_{F,\mathcal{P}}(I))^\top$. Then $\Phi_{F,\mathcal{P}}(I)(A) = \top$, so $\mathcal{P}$ contains some clause

$$A \leftarrow B_1, \dots, B_k, \neg B_{k+1}, \dots, \neg B_m$$

such that $B_i \in I^\top$ and $B_j \in I^\bot$ for all i, j with $1 \leq i \leq k,\ k+1 \leq j \leq m$. But since $I^\top \subseteq J^\top$ and $I^\bot \subseteq J^\bot$, we obtain $B_i \in J^\top$ and $B_j \in J^\bot$ for all i, j. Hence, $\Phi_{F,\mathcal{P}}(J)(A) = \top$ or, equivalently, $A \in (\Phi_{F,\mathcal{P}}(J))^\top$.

Now suppose $A \in (\Phi_{F,\mathcal{P}}(I))^\bot$. Then $\Phi_{F,\mathcal{P}}(I)(A) = \bot$, so for all clauses in $\mathcal{P}$ of the form

$$A \leftarrow B_1, \dots, B_k, \neg B_{k+1}, \dots, \neg B_m$$

there is some i with $1 \leq i \leq k$ such that $B_i \in I^\bot$ or some j with $k+1 \leq j \leq m$ such that $B_j \in I^\top$. But since $I^\top \subseteq J^\top$ and $I^\bot \subseteq J^\bot$, we obtain $B_i \in J^\bot$ for

some i or $B_j \in J^\top$ for some j. So all clauses with head A have a false body in J and hence $\Phi_{F,\mathcal{P}}(J)(A) = \bot$ or, equivalently, $A \in (\Phi_{F,\mathcal{P}}(J))^\bot$. □

In [Fit85], it has been shown that for some programs, the Fitting operator doesn't achieve a fixed point in ω steps, where ω is the limit ordinal. For completeness, we reproduce the counterexample from [Fit85] here:

Example 3.20. Consider the program $\mathcal{P}_1$:

$$\begin{aligned} \mathcal{P}_1 : \qquad p(a) &\leftarrow p(X), q(X). \\ p(s(X)) &\leftarrow p(X). \\ q(b) &\leftarrow \top. \\ q(s(X)) &\leftarrow q(X). \end{aligned}$$

Iterations of $\Phi_{F,\mathcal{P}_1}$ starting from $\langle \emptyset, \emptyset \rangle$ are as follows:

$$\begin{aligned} &\Phi_{F,\mathcal{P}_1}(\langle \emptyset, \emptyset \rangle) = \langle \{\, q(b) \,\}, \{\, p(b), q(a) \,\} \rangle \\ &\Phi^2_{F,\mathcal{P}_1}(\langle \emptyset, \emptyset \rangle) = \langle \{\, q(b), q(s(b)) \,\}, \{\, p(b), q(a), p(s(b)), q(s(a)) \,\} \rangle \\ &\Phi^n_{F,\mathcal{P}_1}(\langle \emptyset, \emptyset \rangle) = \big\langle \{\, q(s^k(b)) \mid 0 \le k < n \,\}, \{\, p(s^k(b)), q(s^k(a)) \mid 0 \le k < n \,\} \big\rangle \\ &\Phi^\omega_{F,\mathcal{P}_1}(\langle \emptyset, \emptyset \rangle) = \big\langle \{\, q(s^k(b)) \mid k \in \mathbb{N} \,\}, \{\, p(s^k(b)), q(s^k(a)) \mid k \in \mathbb{N} \,\} \big\rangle \\ &\Phi^{\omega+1}_{F,\mathcal{P}_1}(\langle \emptyset, \emptyset \rangle) = \Phi^\omega_{F,\mathcal{P}_1}(\langle \emptyset, \emptyset \rangle) \cup \langle \emptyset, \{\, p(a) \,\} \rangle \\ &\Phi^{\omega+n}_{F,\mathcal{P}_1}(\langle \emptyset, \emptyset \rangle) = \Phi^\omega_{F,\mathcal{P}_1}(\langle \emptyset, \emptyset \rangle) \cup \big\langle \emptyset, \{\, p(s^k(a)) \mid 0 \le k < n \,\} \big\rangle \\ &\Phi^{\omega+\omega}_{F,\mathcal{P}_1}(\langle \emptyset, \emptyset \rangle) = \Phi^\omega_{F,\mathcal{P}_1}(\langle \emptyset, \emptyset \rangle) \cup \big\langle \emptyset, \{\, p(s^k(a)) \mid k \in \mathbb{N} \,\} \big\rangle = \Phi^{\omega+\omega+1}_{F,\mathcal{P}_1}(\langle \emptyset, \emptyset \rangle) \end{aligned}$$

For the program above, $\Phi_{F,\mathcal{P}_1}$ infers after the first application to the empty interpretation the interpretation $\langle \{\, q(b) \,\}, \{\, p(b), q(a) \,\} \rangle$. Here, $q(b)$ is true because it is a fact, $p(b)$ and $q(a)$ are false because there is no clause where the head is either $p(b)$ or $q(a)$. After the first ω steps, $\Phi_{F,\mathcal{P}_1}$ infers that $q(s^k(b))$ is true and $p(s^k(b))$ and $q(s^k(a))$ are false for all $k \in \mathbb{N}$. Furthermore, in the next step, $\Phi_{F,\mathcal{P}_1}$ infers that $p(a)$ is false because all of the clauses where $p(a)$ is the head have a false body. Only after $\omega + \omega$ steps $\Phi_{F,\mathcal{P}_1}$ reaches a fixed point where the fixed point is $\big\langle \{\, q(s^k(b)) \mid k \in \mathbb{N} \,\}, \{\, p(s^k(b)), q(s^k(a)), p(s^k(a)) \mid k \in \mathbb{N} \,\} \big\rangle$. Consequently, $\Phi_{F,\mathcal{P}_1}$ is not continuous.

3.4 Stenning and van Lambalgen Operator

Let us recall again the definition of Stenning and van Lambalgen immediate consequence operator [SvL08]:

Let I be an interpretation and $\mathcal{P}$ be an extended program. The *Stenning and van Lambalgen immediate consequence operator* is defined as follows: $\Phi_{SvL,\mathcal{P}}(I) = \big\langle J^\top, J^\bot \big\rangle$, where

$$\begin{aligned} J^\top &= \{\, A \mid \text{there exists } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ with } I(\mathit{Body}) = \top \,\} \text{ and} \\ J^\bot &= \{\, A \mid \text{there exists } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ and} \\ &\qquad \text{for all } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ we find } I(\mathit{Body}) = \bot \,\} \end{aligned}$$

This section is devoted to exploring the same properties of the Stenning and van Lambalgen operator as were investigated for the Fitting operator in the previous section. The results turn out to be the same: the Stenning and van Lambalgen operator is in general monotonic, but not continuous. The following proposition shows the former result:

Proposition 3.21. Let $\mathcal{P}$ be a program. Then $\Phi_{SvL,\mathcal{P}}$ is a monotonic mapping.

Proof. Let I, J be interpretations such that $I \subseteq J$. We need to show that $\Phi_{SvL,\mathcal{P}}(I) \subseteq \Phi_{SvL,\mathcal{P}}(J)$.

Suppose $A \in (\Phi_{SvL,\mathcal{P}}(I))^\top$. Then $\Phi_{SvL,\mathcal{P}}(I)(A) = \top$, so $\mathcal{P}$ contains some clause

$$A \leftarrow B_1, \ldots, B_k, \neg B_{k+1}, \ldots, \neg B_m$$

such that $B_i \in I^\top$ and $B_j \in I^\bot$ for all i, j with $1 \leq i \leq k,\ k \leq j \leq k+1$. But since $I^\top \subseteq J^\top$ and $I^\bot \subseteq J^\bot$, we obtain $B_i \in J^\top$ and $B_j \in J^\bot$ for all i, j. Hence, $\Phi_{SvL,\mathcal{P}}(J)(A) = \top$ or, equivalently, $A \in (\Phi_{SvL,\mathcal{P}}(J))^\top$.

Now suppose $A \in (\Phi_{SvL,\mathcal{P}}(I))^\bot$. Then $\Phi_{SvL,\mathcal{P}}(I)(A) = \bot$, so $\mathcal{P}$ contains a clause of the form

$$A \leftarrow B_1, \ldots, B_k, \neg B_{k+1}, \ldots, \neg B_m$$

and for all such clauses there is some i with $1 \leq i \leq k$ such that $B_i \in I^\bot$ or some j with $k+1 \leq j \leq m$ such that $B_j \in I^\top$. But since $I^\top \subseteq J^\top$ and $I^\bot \subseteq J^\bot$, we obtain $A_i \in J^\bot$ for some i or $B_j \in J^\top$ for some j. So all clauses with head A also have a false body in J and hence $\Phi_{SvL,\mathcal{P}}(J)(A) = \bot$ or, equivalently, $A \in (\Phi_{SvL,\mathcal{P}}(J))^\bot$. □

As for continuity, we cannot use the same counterexample as for the Fitting operator. This is because in case of the program $\mathcal{P}_1$, the Stenning and van Lambalgen operator does reach a fixed point after ω steps:

Example 3.22. Consider the program $\mathcal{P}_1$:

$$\begin{aligned} \mathcal{P}_1 : \qquad p(a) &\leftarrow p(X), q(X). \\ p(s(X)) &\leftarrow p(X). \\ q(b) &\leftarrow \top. \\ q(s(X)) &\leftarrow q(X). \end{aligned}$$

Iterations of $\Phi_{SvL,\mathcal{P}_1}$ starting from $\langle \emptyset, \emptyset \rangle$ are as follows:

$$\begin{aligned} &\Phi_{SvL,\mathcal{P}_1}(\langle \emptyset, \emptyset \rangle) = \langle \{\, q(b) \,\}, \emptyset \rangle \\ &\Phi^n_{SvL,\mathcal{P}_1}(\langle \emptyset, \emptyset \rangle) = \left\langle \{\, q(s^k(b)) \mid 0 \leq k < n \,\}, \emptyset \right\rangle \\ &\Phi^\omega_{SvL,\mathcal{P}_1}(\langle \emptyset, \emptyset \rangle) = \left\langle \{\, q(s^k(b)) \mid k \in \mathbb{N} \,\}, \emptyset \right\rangle = \Phi^{\omega+1}_{SvL,\mathcal{P}_1}(\langle \emptyset, \emptyset \rangle) \end{aligned}$$

Because there is no clause where $p(b)$ or $q(a)$ is the head, $p(b)$ and $q(a)$ are still undefined after the first iteration. Consequently, $p(s^k(b))$ and $q(s^k(a))$ also stay undefined in all the next iterations. Moreover, the bodies of clauses for $p(a)$ are all undefined and hence $p(s^k(a))$ stays undefined for every $k \in \mathbb{N}$ and all subsequent steps. In this case, the fixed point is reached in ω steps.

However, there are other programs for which Stenning and van Lambalgen doesn't reach a fixed point in ω steps. In the following example, $\omega + 1$ steps are necessary to reach a fixed point:

Example 3.23. Consider the following program:

$$\begin{aligned} \mathcal{P}_2 : \qquad q(a) &\leftarrow \top. \\ q(s(X)) &\leftarrow q(X). \\ p &\leftarrow \neg q(X). \end{aligned}$$

Iterations of $\Phi_{SvL,\mathcal{P}_2}$ starting from $\langle \emptyset, \emptyset \rangle$ are as follows:

$$\begin{aligned} \Phi^n_{SvL,\mathcal{P}_2}(\langle \emptyset, \emptyset \rangle) &= \left\langle \{\, q(s^k(a)) \mid k \leq n-1 \,\}, \emptyset \right\rangle \\ \Phi^\omega_{SvL,\mathcal{P}_2}(\langle \emptyset, \emptyset \rangle) &= \left\langle \{\, q(s^k(a)) \mid k \in \mathbb{N} \,\}, \emptyset \right\rangle \\ \Phi^{\omega+1}_{SvL,\mathcal{P}_2}(\langle \emptyset, \emptyset \rangle) &= \left\langle \{\, q(s^k(a)) \mid k \in \mathbb{N} \,\}, \{\, p \,\} \right\rangle \end{aligned}$$

Hence the Stenning and van Lambalgen operator cannot in general be continuous.

3.5 Ground Programs

As was shown in Corollary 3.15, in case the underlying complete partial order is finite, monotonicity implies continuity and so both conditions become equivalent (for the converse implication see Proposition 3.9). Both the Fitting and Stenning and van Lambalgen operators are monotonic (Propositions 3.19 and 3.21, respectively), so in case the space of interpretations is finite, they will also be continuous. One case in which the space of interpretations is finite is for propositional programs. We immediately obtain the following proposition:

Proposition 3.24. Let $\mathcal{P}$ be a propositional program. Then both $\Phi_{F,\mathcal{P}}$ and $\Phi_{SvL,\mathcal{P}}$ are continuous.

Proof. Follows from monotonicity of $\Phi_{F,\mathcal{P}}$ and $\Phi_{SvL,\mathcal{P}}$ (Propositions 3.19 and 3.21, respectively), Proposition 3.15 and the fact that the Herbrand base of $\mathcal{P}$ is finite and hence the complete partial order of interpretations is also finite. □

This result can be also be extended to ground programs.

Proposition 3.25. Let $\mathcal{P}$ be a ground program. Then both $\Phi_{F,\mathcal{P}}$ and $\Phi_{SvL,\mathcal{P}}$ are continuous.

However, the Herbrand base of a ground program can still be infinite (in case the program contains some function symbol) and consequently the set of all interpretations is also infinite. Hence we cannot use the previous argument to prove the above proposition. To overcome this problem, we can map the ground program to a propositional program and show that the behaviour of the operator is preserved.

The rest of this section is concerned only with the proof of Proposition 3.25. We will assume that a ground program $\mathcal{P}$ is given and that $\mathcal{A}_\mathcal{P}$ is the finite set of atoms occuring in $\mathcal{P}$. We also define a transformation that will replace each atom $A \in \mathcal{A}_\mathcal{P}$ by a new predicate symbol p_A of arity 0, resulting in a propositional program $\overline{\mathcal{P}}$. The continuity of $\Phi_{F,\overline{\mathcal{P}}}$ and $\Phi_{SvL,\overline{\mathcal{P}}}$ will then be exploited to show the continuity of $\Phi_{F,\mathcal{P}}$ and $\Phi_{SvL,\mathcal{P}}$.

Let us now start with a more formal definition of the above mentioned transformation:

Definition 3.26. By $\overline{\mathcal{P}}$ we denote the propositional program resulting from $\mathcal{P}$ by replacing every atom $A \in \mathcal{A}_\mathcal{P}$ by the propositional atom p_A. Moreover, for any interpretation I of $\mathcal{P}$, by $\overline{I}$ we denote the interpretation of $\overline{\mathcal{P}}$ such that $\overline{I}(p_A) = I(A)$ for every $A \in \mathcal{A}_\mathcal{P}$.

We will also need a number of auxiliary results that will help us in our final proof.

Lemma 3.27. Let X be a directed set of interpretations of $\mathcal{P}$. Then $\overline{X} = \{\, \overline{I} \mid I \in X \,\}$ is a directed set of interpretations of $\overline{\mathcal{P}}$.

Proof. Let $\overline{I_1}, \overline{I_2} \in \overline{X}$. Then $I_1, I_2 \in X$, so there is some $I \in X$ such that $I_1 \subseteq I$ and $I_2 \subseteq I$. We will show that $\overline{I_1} \subseteq \overline{I}$ and $\overline{I_2} \subseteq \overline{I}$. If $p_A \in \overline{I_1}^\top$, then $A \in I_1^\top \subseteq I^\top$ and hence $p_A \in \overline{I}^\top$. On the other hand, if $p_A \in \overline{I_1}^\bot$, then $A \in I_1^\bot \subseteq I^\bot$ and hence $p_A \in \overline{I}^\bot$. Consequently, $\overline{I_1} \subseteq \overline{I}$. Similarly it can be shown that $\overline{I_2} \subseteq \overline{I}$. □

Lemma 3.28. Let X be a directed set of interpretations of $\mathcal{P}$. Then $\overline{\mathit{lub}(X)} = \mathit{lub}(\overline{X})$.

Proof. Let $A \in \mathcal{A}_\mathcal{P}$. We need to show that $\overline{\mathit{lub}(X)}(p_A) = \mathit{lub}(\overline{X})(p_A)$. By definition we have $\overline{\mathit{lub}(X)}(p_A) = \mathit{lub}(X)(A)$, so it will suffice to show $\mathit{lub}(X)(A) = \mathit{lub}(\overline{X})(p_A)$.

By the previous Lemma and Proposition 3.17 we obtain $\mathit{lub}(X) = \left\langle \bigcup X^\top, \bigcup X^\bot \right\rangle$ and $\mathit{lub}(\overline{X}) = \left\langle \bigcup \overline{X}^\top, \bigcup \overline{X}^\bot \right\rangle$ where $X^\top$, $X^\bot$, $\overline{X}^\top$ and $\overline{X}^\bot$ are defined as in Section 3.2. Further, we have

$$\begin{aligned}
\mathit{lub}(X)(A) = \top &\iff A \in \bigcup X^\top \iff (\exists I \in X)\left(A \in I^\top\right) \\
&\iff (\exists I \in X)\left(p_A \in \overline{I}^\top\right) \iff (\exists I \in \overline{X})\left(p_A \in I^\top\right) \\
&\iff p_A \in \bigcup \overline{X}^\top \iff \mathit{lub}(\overline{X})(p_A) = \top \\
\mathit{lub}(X)(A) = \bot &\iff A \in \bigcup X^\bot \iff (\exists I \in X)\left(A \in I^\bot\right) \\
&\iff (\exists I \in X)\left(p_A \in \overline{I}^\bot\right) \iff (\exists I \in \overline{X})\left(p_A \in I^\bot\right) \\
&\iff p_A \in \bigcup \overline{X}^\bot \iff \mathit{lub}(\overline{X})(p_A) = \bot
\end{aligned}$$

From these two equivalences it follows that $\mathit{lub}(X)(A) = \mathit{lub}(\overline{X})(p_A)$. □

Lemma 3.29. For any interpretation I of $\mathcal{P}$ we have

$$\overline{\Phi_{F,\mathcal{P}}(I)} = \Phi_{F,\overline{\mathcal{P}}}(\overline{I})$$
$$\overline{\Phi_{SvL,\mathcal{P}}(I)} = \Phi_{SvL,\overline{\mathcal{P}}}(\overline{I})$$

Proof. We will show the proof only for the Fitting operator, the proof for the Stenning and van Lambalgen operator is analogous.

Let $A \in \mathcal{A}_\mathcal{P}$. We need to show that $\overline{\Phi_{F,\mathcal{P}}(I)}(p_A) = \Phi_{F,\overline{\mathcal{P}}}(\overline{I})(p_A)$. By definition we have $\overline{\Phi_{F,\mathcal{P}}(I)}(p_A) = \Phi_{F,\mathcal{P}}(I)(A)$, so it will suffice to show $\Phi_{F,\mathcal{P}}(I)(A) = \Phi_{F,\overline{\mathcal{P}}}(\overline{I})(p_A)$. We will prove this by proving two equivalences, one showing that

$\Phi_{F,\mathcal{P}}(I)(A) = \top$ holds if and only if $\Phi_{F,\overline{\mathcal{P}}}(\overline{I})(p_A) = \top$ and the other showing that $\Phi_{F,\mathcal{P}}(I)(A) = \bot$ holds if and only if $\Phi_{F,\overline{\mathcal{P}}}(\overline{I})(p_A) = \bot$:

1. $\Phi_{F,\mathcal{P}}(I)(A) = \top$ holds if and only if $\mathcal{P}$ contains some rule

$$A \leftarrow B_1, \ldots, B_k, \neg B_{k+1}, \ldots, \neg B_m$$

such that $\{ B_1, \ldots, B_k \} \subseteq I^\top$ and $\{ B_{k+1}, \ldots, B_m \} \subseteq I^\bot$. This in turn holds if and only if $\overline{\mathcal{P}}$ contains a rule

$$p_A \leftarrow p_{B_1}, \ldots, p_{B_k}, \neg p_{B_{k+1}}, \ldots, \neg p_{B_m}$$

such that $\{ p_{B_1}, \ldots, p_{B_k} \} \subseteq \overline{I}^\top$ and $\{ p_{B_{k+1}}, \ldots, p_{B_m} \} \subseteq \overline{I}^\bot$. In other words, when $\Phi_{F,\overline{\mathcal{P}}}(\overline{I})(p_A) = \top$.

2. $\Phi_{F,\mathcal{P}}(I)(A) = \bot$ holds if and only if for all clauses in $\mathcal{P}$ of the form

$$A \leftarrow B_1, \ldots, B_k, \neg B_{k+1}, \ldots, \neg B_m$$

there is either some $i \in \{ 1, \ldots, k \}$ such that $B_i \in I^\bot$ or some $j \in \{ k+1, \ldots, m \}$ such that $B_j \in I^\top$. This in turn holds if and only if for all clauses in $\overline{\mathcal{P}}$ of the form

$$p_A \leftarrow p_{B_1}, \ldots, p_{B_k}, \neg p_{B_{k+1}}, \ldots, \neg p_{B_m}$$

there is either some $i \in \{ 1, \ldots, k \}$ such that $p_{B_i} \in \overline{I}^\bot$ or some $j \in \{ k+1, \ldots, m \}$ such that $p_{B_j} \in \overline{I}^\top$. In other words, when $\Phi_{F,\overline{\mathcal{P}}}(\overline{I})(p_A) = \bot$. □

Lemma 3.30. Let X be a directed set of interpretations of $\mathcal{P}$. Then

$$\overline{lub(\Phi_{F,\mathcal{P}}(X))} = \overline{\Phi_{F,\mathcal{P}}(lub(X))}$$
$$\overline{lub(\Phi_{SvL,\mathcal{P}}(X))} = \overline{\Phi_{SvL,\mathcal{P}}(lub(X))}$$

Proof. By Lemmas 3.28 and 3.29 we have

$$\overline{lub(\Phi_{F,\mathcal{P}}(X))} = lub(\overline{\Phi_{F,\mathcal{P}}(X)}) = lub(\Phi_{F,\overline{\mathcal{P}}}(\overline{X}))$$

We know that $\overline{X}$ is a directed set by Lemma 3.27 and since $\overline{\mathcal{P}}$ is propositional, $\Phi_{F,\overline{\mathcal{P}}}$ is continuous by Proposition 3.24. Hence $lub(\Phi_{F,\overline{\mathcal{P}}}(\overline{X})) = \Phi_{F,\overline{\mathcal{P}}}(lub(\overline{X}))$ and by applying Lemmas 3.28 and 3.29 once again we obtain

$$\Phi_{F,\overline{\mathcal{P}}}(lub(\overline{X})) = \Phi_{F,\overline{\mathcal{P}}}(\overline{lub(X)}) = \overline{\Phi_{F,\mathcal{P}}(lub(X))}$$

Hence $\overline{lub(\Phi_{F,\mathcal{P}}(X))} = \overline{\Phi_{F,\mathcal{P}}(lub(X))}$. The claim for $\Phi_{SvL,\mathcal{P}}$ follows by analogous considerations. □

Proof of Proposition 3.25. Let X be a directed set of interpretations. We show $\Phi_{F,\mathcal{P}}(lub(X)) = lub(\Phi_{F,\mathcal{P}}(X))$ by proving that $\Phi_{F,\mathcal{P}}(lub(X))(A) = lub(\Phi_{F,\mathcal{P}}(X))(A)$ for every atom A. We consider two cases:

1. If $A \notin \mathcal{A}_\mathcal{P}$, then for every interpretation I of $\mathcal{P}$ we have $\Phi_{F,\mathcal{P}}(I)(A) = \bot$. It follows that $\Phi_{F,\mathcal{P}}(lub(X)) = \bot$ and also that $\Phi_{F,\mathcal{P}}(I)(A) = \bot$ for all

$I \in X$. Consequently, $lub(\Phi_{F,\mathcal{P}}(X))(A) = \bot = \Phi_{F,\mathcal{P}}(lub(X))(A)$ and our proof is finished.

2. If $A \in \mathcal{A}_\mathcal{P}$, then by Lemma 3.30 we have

$$\begin{aligned}\Phi_{F,\mathcal{P}}(lub(X))(A) &= \overline{\Phi_{F,\mathcal{P}}(lub(X))}(p_A) = \overline{lub(\Phi_{F,\mathcal{P}}(X))}(p_A) \\ &= lub(\Phi_{F,\mathcal{P}}(X))(A) \ .\end{aligned}$$

We finish the proof by showing $\Phi_{SvL,\mathcal{P}}(lub(X))(A) = lub(\Phi_{SvL,\mathcal{P}}(X))(A)$ for every atom A. We consider two cases:

1. If $A \notin \mathcal{A}_\mathcal{P}$, then for every interpretation I of $\mathcal{P}$ we have $\Phi_{SvL,\mathcal{P}}(I)(A) = u$. It follows that $\Phi_{SvL,\mathcal{P}}(lub(X)) = u$ and also that $\Phi_{SvL,\mathcal{P}}(I)(A) = u$ for all $I \in X$. Consequently, $lub(\Phi_{SvL,\mathcal{P}}(X))(A) = u = \Phi_{SvL,\mathcal{P}}(lub(X))(A)$ and our proof is finished.

2. If $A \in \mathcal{A}_\mathcal{P}$, then by Lemma 3.30 we have

$$\begin{aligned}\Phi_{SvL,\mathcal{P}}(lub(X))(A) &= \overline{\Phi_{SvL,\mathcal{P}}(lub(X))}(p_A) = \overline{lub(\Phi_{SvL,\mathcal{P}}(X))}(p_A) \\ &= lub(\Phi_{SvL,\mathcal{P}}(X))(A) \ .\end{aligned}$$

□

Chapter 4

Contraction Property

Motivated by the Banach Contraction Theorem which states that every contraction mapping has a unique fixed point, we try to investigate whether the Fitting and the Stenning and van Lambalgen operators are contractions or not. Starting with the definition of acceptable programs from [AP90, AP93], Fitting showed that the usual two-valued immediate consequence operator is a contraction for acceptable logic programs [Fit94]. He claimed that using the same arguments, it is easy to show that the Fitting operator is also a contraction for any acceptable program. Further, it has been shown in [AP93] that the two-valued immediate consequence and the Fitting operator semantically coincide on the class of acceptable programs.

In this chapter, we first present the necessary preliminaries that we use. In particular, we define the notions of a metric and metric space and also the class of acceptable programs. Then we show that the Fitting operator is a contraction for any acceptable program using the same arguments as Fitting used in [Fit94] for the two-valued immediate consequence operator.

Turning to the Stenning and van Lambalgen operator, we show that in general it is not a contraction for the class of acceptable programs. Then we prove that at least for acyclic logic programs, the Stenning and van Lambalgen operator is guaranteed to be a contraction. However, even for this more restricted class of programs, it does not coincide with the two-valued immediate consequence operator.

The chapter ends with a summary of our results and of the relation between the Stenning and van Lambalgen operator, Fitting operator and two-valued immediate consequence operator.

4.1 Metric Spaces

In this section we introduce the notions from metric spaces that we need later on. For further information and references on this topic the readers can refer to [BS89a, KK01, KS01].

Definition 4.1 (Metric)**.** A *metric* or distance function on a space $\mathcal{M}$ is a mapping

$$d : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$$

where $\mathbb{R}$ is the set of real numbers, such that:

$$d(x, y) = 0 \text{ if and only if } x = y \quad (m_1)$$
$$d(x, y) = d(y, x) \quad (m_2)$$
$$d(x, y) \leq d(x, z) + d(z, y) \quad (m_3)$$

A *pseudo-metric* is a metric except that the first condition is as follows.

$$x = y \text{ implies } d(x, y) = 0 \quad (m_1')$$

Definition 4.2 (Complete Metric Space)**.** A metric space $\mathcal{M}$ is *complete* if every Cauchy sequence converges. A sequence $s_1, s_2, s_3, \ldots$ is *Cauchy* if, for every $\varepsilon > 0$ there is an integer N such that for all $n, m \geq N, d(s_n, s_m) \leq \varepsilon$. The sequence *converges* if there is an s such that, for every $\varepsilon > 0$, there is an integer N such that for all $n \geq N, d(s_n, s) \leq \varepsilon$.

Definition 4.3 (Contraction)**.** Let $\mathcal{M}$ be a metric space. A mapping

$$f : \mathcal{M} \to \mathcal{M}$$

is a *contraction* if for all $x, y \in \mathcal{M}$ there exists a $k \in \mathbb{R}$ with $0 < k < 1$ such that

$$d(f(x), f(y)) \leq k \cdot d(x, y)$$

Theorem 4.4 (Banach Contraction Theorem)**.** A contraction mapping f on a complete metric space has a unique fixed point. Further, the sequence $x, f(x), f(f(x)), \ldots$ converges to this fixed point for any x.

Proof. The proof can be found in [Wil04]. □

4.2 Acyclic and Acceptable Programs

Let us start with the relatively simple and strict notion of an acyclic program. The acyclicity of a program is guaranteed by the existence of a certain fixed assignment of natural numbers to atoms that is called a *level mapping*.

Definition 4.5 (Level Mapping)**.** A *level mapping* for a program $\mathcal{P}$ is a function

$$l : \mathcal{B}_\mathcal{P} \to \mathbb{N}$$

where $\mathbb{N}$ is the set of natural numbers and $\mathcal{B}_\mathcal{P}$ is the Herbrand base for $\mathcal{P}$. A *partial level mapping* is a level mapping that may be undefined for some ground atoms. We extend the definition of level mapping to a mapping from ground literals to natural numbers by setting $l(\neg A) = l(A)$.

Definition 4.6 (Acyclic Program)**.** Let $\mathcal{P}$ be a logic program and l a level mapping for $\mathcal{P}$. $\mathcal{P}$ is *acyclic with respect to* l if for every clause $A \leftarrow B_1, \ldots, B_m$ in *ground*($\mathcal{P}$) we find that

$$l(A) > l(B_i) \text{ for all } i \text{ with } 1 \leq i \leq m.$$

$\mathcal{P}$ is *acyclic* if it is acyclic with respect to some level mapping.

The problem of deciding whether a program is acyclic is, however, not decidable [AB90]. Further, the condition of acyclicity restricts the ways in which recursion can be used in the program. In some cases, these restrictions are too strict, for example when transitively closed predicates need to be defined. In order to overcome these limitations, the broader class of acceptable programs has been defined [AP90, AP93]. This definition uses two-valued interpretations, so before we start with the definition, let us take a look at the correlation between two-valued and three-valued interpretations.

A three-valued interpretation $I = \langle I^\top, I^\bot \rangle$ is called *total* if and only if $I^\top \cup I^\bot = \mathcal{B}_\mathcal{P}$. Total three-valued interpretations directly correspond to two-valued ones because they do not assign u to any atom. In particular, if we represent each two-valued interpretation by the subset of $\mathcal{B}_\mathcal{P}$ of atoms that it maps to true, then the three-valued interpretation I corresponds to the two-valued interpretation $I^\top$. To avoid confusion, we will distinguish between the positive component $I^\top$ of a three-valued interpretation I and a two-valued interpretation corresponding to I by denoting the latter by $I_2^\top$. This correspondence can also be extended to the case of non-total three-valued interpretations and amounts to assigning false to every atom that was undefined in I.

Similarly, a two-valued interpretation $I_2^\top$ also uniquely identifies the total three-valued interpretation $I = \langle I_2^\top, \mathcal{B}_\mathcal{P} \setminus I_2^\top \rangle$. To distinguish between the assignment of true by a two-valued and a three-valued interpretation, we will use the symbols $\top_2$ and $\top_3$, respectively. Similarly, to distinguish the two-valued and three-valued false we will use the symbols $\bot_2$ and $\bot_3$, respectively.

Let $I_2^\top$ be a two-valued interpretation. $I_2^\top$ is a *two-valued model* for a formula F if $I_2^\top(F) = \top_2$.

Remark. For the rest of this chapter, we use $I_2^\top$ to denote a two-valued interpretation and I to denote a three-valued interpretation.

Proposition 4.7. Let I be a three-valued interpretation and L a literal. Then

1. $I(L) = \top_3$ implies $I_2^\top(L) = \top_2$.
2. $I_2^\top(L) = \top_2$ implies $I(L) \neq \bot_3$, i.e. either $I(L) = \top_3$ or $I(L) = u$.

Proof. 1. Suppose $L = A$ and $I(L) = \top_3$. Then, $A \in I^\top$, so $I_2^\top(A) = \top_2$. Suppose $L = \neg A$ and $I(L) = \top_3$. Then, $A \in I^\bot$, so $A \notin I_2^\top$. Hence, $I_2^\top(L) = \top_2$.

2. Suppose $L = A$ and $I_2^\top(L) = \top_2$. Then, $A \in I_2^\top$, so $I(A) = \top_3$. Suppose $L = \neg A$ and $I_2^\top(L) = \top_2$. Then, $A \notin I_2^\top$, so $I(L) \neq \bot_3$. □

The notion of an acceptable program was first introduced by Apt and Pedreschi in [AP90] for definite logic programs and later extended in [AP93] for logic programs. Intuitively, a program $\mathcal{P}$ is acceptable if for some level mapping and a two-valued model $M_2^\top$ of $\mathcal{P}$, for all ground instances of the clauses of the program, the level of the head is higher than the level of atoms in a certain prefix of the body. Which prefix is considered is determined by the model $M_2^\top$.

Definition 4.8 (Acceptable Definite Program)**.** Let $\mathcal{P}$ be a definite program, l a level mapping for $\mathcal{P}$ and $M_2^\top$ a two-valued model of $\mathcal{P}$. $\mathcal{P}$ is called *acceptable*

with respect to l and $M_2^\top$ if for every clause $A \leftarrow B_1, \ldots, B_m$ in *ground*($\mathcal{P}$) the following implication holds for all i with $1 \leq i \leq m$:

$$M_2^\top(B_1 \wedge \cdots \wedge B_{i-1}) = \top_2 \text{ implies } l(A) > l(B_i).$$

$\mathcal{P}$ is called *acceptable* if it is acceptable with respect to some level mapping and some model of $\mathcal{P}$.

Thus, given a level mapping l and a model $M_2^\top$ of $\mathcal{P}$, in the definition of acceptability with respect to l and $M_2^\top$ for every clause $A \leftarrow B_1, \ldots, B_m$ in *ground*($\mathcal{P}$) we only require the level of A to be higher than the level of the B_is in a certain prefix of $B_1, \ldots, B_m$. Which B_is are taken into account is determined by the model $M_2^\top$. If $M_2^\top(B_1 \wedge \ldots \wedge B_m) = \top_2$, then all of them are considered. Otherwise we consider only those whose index is less or equal to k where k is the least index i for which $M_2^\top(B_i) = \bot_2$.

The following example gives an illustration of the idea underlying the above definition.

Example 4.9. Consider the definite programs $CalP_1$ and $\mathcal{P}_2$:

$$\begin{aligned}
\mathcal{P}_{even1}: \quad & even(0) \leftarrow \top. \\
& odd(s(0)) \leftarrow \top. \\
& even(s(X)) \leftarrow odd(X). \\
& odd(s(X)) \leftarrow even(X). \\
\mathcal{P}_{even2}: \quad & even(0). \\
& even(s(s(X))) \leftarrow even(X). \\
& even(X) \leftarrow even(s(s(X))).
\end{aligned}$$

Both programs, $\mathcal{P}_{even1}$ and $\mathcal{P}_{even2}$, compute even numbers. Let $M_2^\top = \{ even(s^k(0)) \mid k \in \{0, 2, \ldots\} \} \cup \{ odd(s^k(0)) \mid k \in \{1, 3, \ldots\} \}$ be a model for $\mathcal{P}_{even1}$ and $l(even(s^k(0))) = k$ and $l(odd(s^k(0))) = k$. Please note that for the clause $even(s(X)) \leftarrow odd(X)$, the level of $even(s(X))$ in the head is always one greater than the level of $odd(X)$ in the body of the clause. Similarly also for the clause $odd(s(X)) \leftarrow even(X)$. Hence, $\mathcal{P}_{even1}$ is acceptable with respect to the model $M_2^\top$ and level mapping l. On the other hand, $\mathcal{P}_{even2}$ is not acceptable because there is a cycle in the program: $l(s(s(X))) > l(X)$ and $l(X) > l(s(s(X)))$ will lead to a contradiction for any model and level mapping.

Now we generalize the above concept to acceptability of logic programs. First, we define the notion of dependence on ground atoms in a program $\mathcal{P}$.

Definition 4.10. Let $\mathcal{P}$ be a program and A and B be ground atoms.

- *A refers to B* if there is a clause in *ground*($\mathcal{P}$) with A in its head and B or $\neg B$ in its body.
- *A depends on B* if $A = B$, or there is a sequence $A = B_1, B_2, \ldots, B_n = B$ where B_i refers to B_{i+1} for all i with $1 \leq i < n$.
- $Neg_\mathcal{P}$ is the set of ground atoms A such that *ground*($\mathcal{P}$) contains a clause with $\neg A$ in its body.

- $Neg^*_{\mathcal{P}}$ is the set of ground atoms on which the atoms in $Neg_{\mathcal{P}}$ depend.
- $\mathcal{P}^-$ is the set of clauses in $\mathcal{P}$ whose head contains an atom that has an instance in $Neg^*_{\mathcal{P}}$.

Example 4.11. Suppose $\mathcal{P}_1$ is the following program:

$$\begin{aligned} \mathcal{P}_1 : \quad & p \leftarrow q. \\ & p \leftarrow \neg r. \\ & r \leftarrow s. \end{aligned}$$

Then we have:

$$\begin{aligned} Neg_{\mathcal{P}_1} &= \{\, r \,\} \\ Neg^*_{\mathcal{P}_1} &= \{\, r, s \,\} \\ \mathcal{P}_1^- &= \{\, r \leftarrow s. \,\} \end{aligned}$$

Definition 4.12 (Acceptable Program)**.** Let $\mathcal{P}$ be a program, l be a level mapping for $\mathcal{P}$ and $M_2^\top$ be a two-valued model of $\mathcal{P}$ whose restriction to the atoms in $Neg^*_{\mathcal{P}}$ is a model for *comp*($\mathcal{P}^-$). $\mathcal{P}$ is *acceptable with respect to l and $M_2^\top$* if, for every clause $A \leftarrow B_1, \ldots, B_m$ in *ground*($\mathcal{P}$), and for every i with $1 \leq i \leq m$,

$$M_2^\top(B_1 \wedge \cdots \wedge B_{i-1}) = \top_2 \text{ implies } l(A) > l(B_i).$$

$\mathcal{P}$ is *acceptable* if it is acceptable with respect to some level mapping and some model of $\mathcal{P}$.

A program is acceptable if there is some two-valued model for it which is well behaved with respect to the uses of negation in the program, and with respect to which clause bodies are "simpler" than clause heads. The essential difference between this notion of simpler and that used in the definition of acyclicity is that we do not look at the whole of a clause body but only at enough of it, starting from the left, to know whether it is true or false in the model.

For a definite program $\mathcal{P}$ we have $Neg^*_{\mathcal{P}} = \emptyset$, so $\mathcal{P}^-$ is empty and the above definition coincides with the definition of acceptability for definite programs.

We will prove that for any acceptable program $\mathcal{P}$, the Fitting operator $\Phi_{F,\mathcal{P}}$ is a contraction and hence by the Banach Contraction Theorem has a unique fixed point. Further, we will provide a counterexample showing that a similar result does not hold for the Stenning and van Lambalgen operator. We will, however, give a proof of a weaker result, in particular that the Stenning and van Lambalgen operator is a contraction for any acyclic program $\mathcal{P}$.

The fundamental notions needed to prove both results are metrics and pseudo-metrics on the space of all three-valued interpretations. Such metrics can be defined based on level mappings as follows:

Proposition 4.13. Let l be a partial level mapping and I and J be three-valued

interpretations. The function $d_l : \mathcal{I} \times \mathcal{I} \to \mathbb{R}$ defined as

$$d_l(I,J) = \begin{cases} \left(\frac{1}{2}\right)^n & I \neq J \text{ and } I(A) = J(A) \neq u \text{ for all } A \text{ with } l(A) < n \text{ and} \\ & I(A) \neq J(A) \text{ or } I(A) = J(A) = u \text{ for some } A \\ & \text{with } l(A) = n \\ 0 & \text{otherwise} \end{cases}$$

is a pseudo-metric. Further, if l is total, then d_l is a metric.

Proof. To confirm that d_l is a pseudo-metric, let us check one by one the conditions from Def. 4.1. The weakened condition (m_1') is trivially fulfilled and condition (m_2) follows from the symmetry of the definition. The verification of the triangle inequality in condition (m_3) follows:

Let I, J and K be three-valued interpretations. If $d_l(I,K) = 0$, then I and K coincide on all atoms whose level is defined and are, hence, interchangeable as arguments of the metric d_l. So $d_l(I,J) = d_l(K,J) = 0 + d_l(K,J) = d_l(I,K) + d_l(K,J)$. Similarly if $d_l(K,J) = 0$, then K and J coincide on all atoms whose level is defined and are, hence, interchangeable as arguments of the metric d_l. So $d_l(I,J) = d_l(I,K) = d_l(I,K) + 0 = d_l(I,K) + d_l(K,J)$.

For the principal case let us assume $d_l(I,K) = \left(\frac{1}{2}\right)^m$ and $d_l(K,J) = \left(\frac{1}{2}\right)^k$ for some $m, k \in \mathbb{N}$. Without loss of generality we may assume that $m \leq k$. We will show that $d_l(I,J) \leq \left(\frac{1}{2}\right)^m$. Take some atom A with $l(A) < m$. Then we have $I(A) = K(A) \neq u$ and also $K(A) = J(A) \neq u$. Consequently, $I(A) = J(A) \neq u$ and we are done.

In case l is total, we need to prove that $d_l(I,J) = 0$ implies $I = J$. Suppose $I \neq J$. Then I and J do not coincide on some atoms. Take an atom A of a minimal level such that either $I(A) \neq J(A)$ or $I(A) = J(A) = u$. Then $d_l(I,J) = \left(\frac{1}{2}\right)^{l(A)} \neq 0$ and we are done. □

Example 4.14. Suppose we have a program as follows.

$$\begin{aligned} \mathcal{P}_{even3}: \qquad & even(0) \leftarrow \top. \\ & even(s(X)) \leftarrow \neg even(X). \end{aligned}$$

Let

$$\begin{aligned} I &= \left\langle \{\, even(s^k(0)) \mid k \in \{\, 0, 2, \dots \,\} \,\}, \{\, even(s^k(0)) \mid k \in \{\, 1, 3, \dots \,\} \,\} \right\rangle \ , \\ J &= \left\langle \{\, even(s^k(0)) \mid k \in \{\, 0, 2, \dots \,\} \,\}, \emptyset \right\rangle \end{aligned}$$

and $l(even(s^k(0))) = k$. Then, $d_l(I,J) = \frac{1}{2}$ because for $n = 1$

$$I(even(s^n(0))) = \bot \text{ but } J(even(s^n(0))) = u \ .$$

The previous result also gives rise to the following definition:

Definition 4.15 (Metric Induced by a Level Mapping)**.** For any (partial) level mapping l, the *(pseudo-)metric d_l induced by l* is defined as in the previous proposition.

4.3 Fitting Operator

This section is devoted to proving that the Fitting operator $\Phi_{F,\mathcal{P}}$ is a contraction for any acceptable program $\mathcal{P}$. Let us first recall the definition of the Fitting operator.

Let I be an interpretation and $\mathcal{P}$ a program. The *Fitting immediate consequence operator* is defined as follows: $\Phi_{F,\mathcal{P}}(I) = \langle J^\top, J^\bot \rangle$, where

$$J^\top = \{\, A \mid \text{there exists } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ with } I(\mathit{Body}) = \top \,\} \text{ and}$$
$$J^\bot = \{\, A \mid \text{for all } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ we find } I(\mathit{Body}) = \bot \,\}.$$

In order to proceed with the proof, we will make the following assumptions:

Remark. For the rest of this section we assume that $\mathcal{P}$ is a program that is acceptable with respect to the level mapping $l_\mathcal{P}$ and the two-valued model $M_2^\top$ of $\mathcal{P}$. We will denote by M the total three-valued interpretation corresponding to $M_2^\top$, i.e.

$$M = \langle M_2^\top, \mathcal{B}_\mathcal{P} \setminus M_2^\top \rangle$$

Further, let $\left(M_2^\top\right)^*$ be $M_2^\top$ restricted to the ground atoms in $Neg_\mathcal{P}^*$ and

$$M^* = \left\langle \left(M_2^\top\right)^*, \mathcal{B}_\mathcal{P} \setminus \left(M_2^\top\right)^* \right\rangle$$

We will be using the following general facts in our proofs:

(F_1) By the definition of acceptable program, $\left(M_2^\top\right)^*$ is a model of $\mathit{comp}(\mathcal{P}^-)$, so M^* is also a model of $\mathit{comp}(\mathcal{P}^-)$ and it follows from [Fit85] that $\Phi_{F,\mathcal{P}^-}(M^*) = M^*$.

(F_2) The behaviour of atoms in $Neg_\mathcal{P}^*$ only depends on clauses in $\mathcal{P}^-$. More precisely, suppose $A \in Neg_\mathcal{P}^*$ and I is any interpretation. Then $\Phi_{F,\mathcal{P}}(I)(A) = \Phi_{F,\mathcal{P}^-}(I)(A)$

An acceptable program is modular, in the sense that the behavior of atoms in $Neg_\mathcal{P}^*$ does not depend on the behavior of atoms which are not in $Neg_\mathcal{P}^*$. In the following, we will first work with the atoms in $Neg_\mathcal{P}^*$, and later on with the atoms which are not in $Neg_\mathcal{P}^*$.

We will now define a restriction l_1 of the level mapping $l_\mathcal{P}$ to atoms in $Neg_\mathcal{P}^*$ and prove that $\Phi_{F,\mathcal{P}}$ is a contraction with respect to the pseudo-metric induced by l_1.

Definition 4.16. We define the level mapping l_1 as follows: If $A \in Neg_\mathcal{P}^*$, then $l_1(A) = l_\mathcal{P}(A)$, and for all other ground atoms l_1 is undefined. Further, let d_1 be the pseudo-metric induced by the partial level mapping l_1.

We show a convergence result directly.

Lemma 4.17. Let J be any three-valued interpretation. Then

$$d_1(\Phi_{F,\mathcal{P}}(J), M) = d_1(\Phi_{F,\mathcal{P}^-}(J), M) \leq \frac{1}{2} \cdot d_1(J, M)$$

Proof. The pseudo-metric d_1 only depends on atoms in $Neg_\mathcal{P}^*$ and by general fact (F_2), $\Phi_{F,\mathcal{P}}(J)$ and $\Phi_{F,\mathcal{P}^-}(J)$ agree on such atoms, so we can easily see that the equality $d_1(\Phi_{F,\mathcal{P}}(J), M) = d_1(\Phi_{F,\mathcal{P}^-}(J), M)$ holds.

Suppose $d_1(J, M) = \left(\frac{1}{2}\right)^n$, so that J and M agree on all ground atoms A with $l_1(A) < n$ but differ on a ground atom A such that $l_1(A) = n$. Recall that M is total, so to prove $d_1(\Phi_{F,\mathcal{P}^-}(J), M) \leq \frac{1}{2} \cdot d_1(J, M)$ it is enough to show that $\Phi_{F,\mathcal{P}^-}(J)$ and M agree on all ground atoms A with $l_1(A) \leq n$.

Let A be a ground atom with $l_1(A) \leq n$. Since $l_1(A)$ is defined, $A \in Neg^*_{\mathcal{P}}$. Suppose $\Phi_{F,\mathcal{P}^-}(J)(A) \neq M(A)$. Because M is total, it cannot be the case that $M(A) = u$, so we have two cases to consider:

1. $M(A) = \top_3$ and $\Phi_{F,\mathcal{P}^-}(J)(A) \neq \top_3$. By general fact ($F_1$) we obtain $M(A) = M^*(A) = \Phi_{F,\mathcal{P}^-}(M^*)(A) = \top_3$. So there exists some clause $A \leftarrow B_1, \ldots, B_m$ in *ground*$(\mathcal{P}^-)$ such that $M^*(B_i) = M(B_i) = \top_3$ for all $1 \leq i \leq m$. Using Prop. 4.7(1) we obtain $M_2^\top(B_1 \wedge \cdots \wedge B_m) = \top_2$, so by the acceptability of $\mathcal{P}$ with respect to $M_2^\top$ we obtain for all $1 \leq i \leq m$

$$l(B_i) < l(A) \leq n$$

   Since $d_1(J, M) = \left(\frac{1}{2}\right)^n$, J and M must agree on all such B_i, so $\Phi_{F,\mathcal{P}^-}(J)(B_i) = \top_3$. But at the same time $\Phi_{F,\mathcal{P}^-}(J)(A) \neq \top_3$, so there also exists some k with $1 \leq k \leq m$ such that $\Phi_{F,\mathcal{P}^-}(J)(B_k) \neq \top_3$, which is a contradiction.

2. $M(A) = \bot_3$ and $\Phi_{F,\mathcal{P}^-}(J)(A) \neq \bot_3$. By the latter it follows that there must be a clause $A \leftarrow B_1, \ldots, B_m$ in *ground*$(\mathcal{P}^-)$ such that $J(B_i) \neq \bot_3$ for all $1 \leq i \leq m$. Because $M(A) = \bot_3$, we obtain from the general fact (F_1) that $M(A) = M^*(A) = \Phi_{F,\mathcal{P}^-}(M^*)(A) = \bot_3$. It follows that there exists a k with $M^*(B_k) = \bot_3$ and $M^*(B_i) = \top_3$ for $1 \leq i < k$. Hence $M_2^\top(B_1 \wedge \cdots \wedge B_{k-1}) = \top_2$ by Proposition 4.7(1) and by the acceptability of $\mathcal{P}$ with respect to $M_2^\top$ we have that for all $1 \leq i \leq k$

$$l(B_i) < l(A) \leq n$$

   Since $d_1(J, M) = \left(\frac{1}{2}\right)^n$, J and M must agree on B_k, that is J and M^* must agree on B_k and they do not, which is a contradiction. □

It follows immediately from this Lemma that any sequence of interpretations $J, \Phi_{F,\mathcal{P}}(J), \Phi_{F,\mathcal{P}}(\Phi_{F,\mathcal{P}}(J)), \ldots$ converges to M given the pseudo-metric d_1. That is, the behavior of atoms in $Neg^*_{\mathcal{P}}$ is completely determined by the program $\mathcal{P}$ and agrees on them with the model M.

Now we turn to the atoms which are not in $Neg^*_{\mathcal{P}}$. This time we will find their behavior is also uniquely characterized, but need not agree with M. We will use a metric essentially consisting of two parts. One part measures how close the interpretation of atoms in $Neg^*_{\mathcal{P}}$ is to their meaning in M. This is much like what we did with d_1. The other part concerns itself with atoms which are not in $Neg^*_{\mathcal{P}}$. We will focus on the existence of contradictions to the "assertions" of M and how significant they are.

Definition 4.18. We define the level mapping l_2 as follows: If $A \notin Neg^*_{\mathcal{P}}$, then $l_2(A) = l_{\mathcal{P}}(A)$, and for all other ground atoms l_2 is undefined. Further, let d_2 be the pseudo-metric induced by the partial level mapping l_2.

Definition 4.19. A three-valued interpretation J *correctly asserts a ground atom* A if $J(A) = \top_3$ implies $M_2^\top(A) = \top_2$, that is either $J(A) = \bot_3$ or $J(A) = u$ or $M_2^\top(A) = \top_2$.

Definition 4.20. The mapping ρ on ground atoms which are not in $Neg^*_{\mathcal{P}}$ is defined as follows:

- set $\rho(J) = 0$ if J correctly asserts all ground atoms $A \notin Neg^*_{\mathcal{P}}$,
- otherwise, set $\rho(J) = \left(\frac{1}{2}\right)^n$ when n is the smallest integer such that J does not correctly assert a ground atom $A \notin Neg^*_{\mathcal{P}}$ with $l_2(A) = n$.

Definition 4.21. We define the mapping $d_3 : \mathcal{I} \times \mathcal{I} \to \mathbb{R}$ for any three-valued interpretations I, J as:

$$d_3(I,J) = \begin{cases} 0 & \text{if } I = J \\ \max\{\, d_1(I,M), d_1(J,M), d_2(I,J), \rho(I), \rho(J) \,\} & \text{if } I \neq J \end{cases}$$

In the following, we will show that the mapping d_3 we just defined is a metric with respect to which the Fitting operator is a contraction. A few preparatory claims are in place here. First of all, we need to show that d_3 is a metric. Then we need to prove that the space of all interpretations is a complete metric space with respect to the metric d_3. Finally, we can turn to showing that the Fitting operator is a contraction in this metric space and hence by the Banach Contraction Theorem has a unique fixed point that can be reached by iterating it from any interpretation.

Let's get started by showing that d_3 satisfies all the conditions for being a metric:

Lemma 4.22. Let $S, T, U \subseteq \mathbb{R}$ be finite sets of real numbers such that for every $s \in S$ there are some $t \in T$ and $u \in U$ such that $s \leq t + u$. Then

$$\max(S) \leq \max(T) + \max(U)$$

Proof. Since S is finite, for some $s_{\max} \in S$ is holds that $s_{\max} = \max(S)$. Let $t \in T$ and $u \in U$ be such that $s_{\max} \leq t + u$. We immediately obtain

$$\max(S) = s_{\max} \leq t + u \leq \max(T) + \max(U) \ . \qquad \square$$

Proposition 4.23. The mapping d_3 is a metric.

Proof. We will verify that the conditions (m_1), (m_2) and (m_3) hold for d_3.

For condition (m_1), suppose first that $d_3(I,J) = 0$ and $I \neq J$ for some three-valued interpretations I, J. We will derive a conflict. We have $d_1(I,M) = d_1(J,M) = d_2(I,J) = \rho(I) = \rho(J) = 0$. From $I \neq J$ we obtain that I and J do not agree on some atoms. Let A be some atom with minimal $l_{\mathcal{P}}(A)$ such that either $I(A) \neq J(A)$ or $I(A) = u$ or $J(A) = u$. We will consider two cases:

1. If $A \in Neg^*_{\mathcal{P}}$, then from $d_1(I,M) = d_1(J,M) = 0$ we obtain $I(A) = J(A) = M(A) \neq u$ which contradicts the fact that either $I(A) \neq J(A)$ or $I(A) = J(A) = u$.
2. If $A \notin Neg^*_{\mathcal{P}}$, then from $d_2(I,J) = 0$ we obtain $I(A) = J(A) \neq u$ which contradicts the fact that either $I(A) \neq J(A)$ or $I(A) = J(A) = u$.

For the converse implication assume $I = J$. Then $d_3(I,J) = 0$ by definition.

The condition (m_2) follows directly from the symmetry of d_3 and d_2.

Now let's verify the condition (m_3). Let I, J, K be three-valued interpretations. If $I = J$, then $d_3(I, J) = 0$, so the inequality is trivially satisfied. Assume $I \neq J$. If $I = K$, then $d_3(I, K) = 0$ and $d_3(I, J) = d_3(K, J) = d_3(J, K) = 0 + d_3(J, K) = d_3(I, K) + d_3(K, J)$. Similarly if $J = K$, then $d_3(J, K) = 0$ and $d_3(I, J) = d_3(I, K) = d_3(I, K) + 0 = d_3(I, K) + d_3(K, J)$.

So we can assume that I, J, K are pairwise distinct, i.e. $I \neq J \neq K \neq I$. Then,

$$\begin{aligned} d_3(I, J) &= \max\{\, d_1(I, M), d_1(J, M), d_2(I, J), \rho(I), \rho(J)\,\} = \max(S) \ , \\ d_3(I, K) &= \max\{\, d_1(I, M), d_1(K, M), d_2(I, K), \rho(I), \rho(K)\,\} = \max(T) \ , \\ d_3(J, K) &= \max\{\, d_1(K, M), d_1(J, M), d_2(K, J), \rho(K), \rho(J)\,\} = \max(U) \end{aligned}$$

where

$$\begin{aligned} S &= \{\, d_1(I, M), d_1(J, M), d_2(I, J), \rho(I), \rho(J)\,\} \ , \\ T &= \{\, d_1(I, M), d_1(K, M), d_2(I, K), \rho(I), \rho(K)\,\} \ , \\ U &= \{\, d_1(K, M), d_1(J, M), d_2(K, J), \rho(K), \rho(J)\,\} \ . \end{aligned}$$

Further, since d_2 is a pseudo-metric (Proposition 4.13), it satisfies condition (m_3), i.e.

$$d_2(I, J) \leq d_2(I, K) + d_2(K, J) \tag{4.1}$$

Our aim is to show $d_3(I, J) \leq d_3(I, K) + d_3(K, J)$, i.e. that

$$\max(S) \leq \max(T) + \max(U) \ .$$

It can be verified easily that this directly follows from Lemma 4.22 and (4.1). □

Now we will aim to show that the space of three-valued interpretations is a complete metric space. In order to prove this result, we will need the following lemma:

Lemma 4.24. Let I, J be three-valued interpretations such that $d_3(I, J) \leq \left(\frac{1}{2}\right)^{n+1}$. Then for every atom A with $l_{\mathcal{P}}(A) \leq n$ we have $I(A) = J(A)$.

Proof. If $I = J$, then the claim follows trivially. Otherwise $d_3(I, J) \leq \left(\frac{1}{2}\right)^{n+1}$ implies that each $d_1(I, M), d_1(J, M)$ and $d_2(I, J)$ is $\leq \left(\frac{1}{2}\right)^{n+1}$. Take some atom A with $l_{\mathcal{P}}(A) \leq n$. We will consider two cases:

a) If $A \in Neg^*_{\mathcal{P}}$, then from $d_1(I, M) \leq \left(\frac{1}{2}\right)^{n+1}$ and $d_1(J, M) \leq \left(\frac{1}{2}\right)^{n+1}$ we obtain that $I(A) = M(A) = J(A) \neq u$ and we are done.

b) If $A \notin Neg^*_{\mathcal{P}}$, then from $d_2(I, J) \leq \left(\frac{1}{2}\right)^{n+1}$ we directly obtain $I(A) = J(A) \neq u$ and we are done. □

We are now ready to show that the space of all three-valued interpretations with the metric d_3 is a complete metric space.

Proposition 4.25. The space of three-valued interpretations using the metric d_3 is a complete metric space.

Proof. Suppose $\{S_k\}_{k=1}^{\infty}$ is a Cauchy sequence of interpretations. Then for any $n \in \mathbb{N}$ there must be some $K \in \mathbb{N}$ such that for any $k_1, k_2 \geq K$ we have

$$d_3(S_{k_1}, S_{k_2}) \leq \left(\frac{1}{2}\right)^{n+1}$$

Let K_n be the least such K for every $n \in \mathbb{N}$. This definition implies

$$K_{n_1} \leq K_{n_2} \text{ for any } n_1, n_2 \in \mathbb{N} \text{ with } n_1 \leq n_2.$$

Let us define our limit interpretation S for any atom A as $S(A) = S_{K_l}(A)$ where $l = l_{\mathcal{P}}(A)$.

We now need to prove that for any $\varepsilon > 0$ there is some $K \in \mathbb{N}$ such that for any $k \geq K$ we have $d_3(S, S_k) \leq \varepsilon$. Choose some $\varepsilon > 0$ and let $n \in \mathbb{N}$ be such that $\left(\frac{1}{2}\right)^{n+1} \leq \varepsilon$. We will prove $d_3(S, S_k) \leq \left(\frac{1}{2}\right)^{n+1}$ for any $k \geq K_n$ from which the claim will follow.

We will first prove an auxiliary claim: For any atom A with $l_{\mathcal{P}}(A) \leq n$ and any $k \geq K_n$ it holds that

$$S(A) = S_{K_n}(A) = S_k(A) \tag{4.2}$$

Proof. Let $l_{\mathcal{P}}(A) = l \leq n$. Then $K_l \leq K_n \leq k$ and hence by the definition of K_l we have $d_3(S_{K_l}, S_{K_n}) \leq \left(\frac{1}{2}\right)^{l+1}$ and also $d_3(S_{K_l}, S_k) \leq \left(\frac{1}{2}\right)^{l+1}$. Consequently, by the previous Lemma we have $S(A) = S_{K_l}(A) = S_{K_n}(A) = S_k(A)$. □

Now we are ready to proceed with the main proof. Choose some $k_1 \geq K_n$. In case $S = S_{k_1}$ we have $d_3(S, S_{k_1}) = 0$ and we are done. Now let's consider a marginal case in which $S_{k_1} = S_k$ for all $k \geq K_n$. In that case we also have $S = S_{k_1}$ because

- For any atom A with $l_{\mathcal{P}}(A) \leq n$ we have by (4.2) $S(A) = S_{k_1}(A)$ and
- For any atom A with $l_{\mathcal{P}}(A) = l > n$ we have $S(A) = S_{K_l}(A)$ for some $K_l \geq K_n$. Therefore $S_{k_1} = S_{K_l}$ and hence $S(A) = S_{k_1}(A)$.

Let's move to the principal case in which $S \neq S_{k_1}$ and also $S_{k_1} \neq S_{k_2}$ for some $k_2 \geq K_n$. Then

$$d_3(S_{k_2}, S_{k_1}) = \max\{\, d_1(S_{k_2}, M), d_1(S_{k_1}, M), d_2(S_{k_2}, S_{k_1}), \rho(S_{k_2}), \rho(S_{k_1}) \,\}$$

and by the definition of K_n also $d_3(S_{k_2}, S_{k_1}) \leq \left(\frac{1}{2}\right)^{n+1}$ because both $k_2 \geq K_n$ and $k_1 \geq K_n$. Consequently also each of $d_1(S_{k_2}, M)$, $d_1(S_{k_1}, M)$, $d_2(S_{k_2}, S_{k_1})$, $\rho(S_{k_2})$ and $\rho(S_{k_1})$ is $\leq \left(\frac{1}{2}\right)^{n+1}$. Furthermore, by (4.2) we obtain $S(A) = S_{k_2}(A)$ for any atom A with $l_{\mathcal{P}}(A) \leq n$. But then by the definitions of d_1, d_2 and ρ we obtain that each of $d_1(S, M), d_2(S, S_{k_1})$ and $\rho(S)$ is also $\leq \left(\frac{1}{2}\right)^{n+1}$. Hence

$$d_3(S, S_{k_1}) = \max\{\, d_1(S, M), d_1(S_{k_1}, M), d_2(S, S_{k_1}), \rho(S), \rho(S_{k_1}) \,\} \leq \left(\frac{1}{2}\right)^{n+1}$$

as desired. □

Now we are finally ready to prove that the Fitting operator is a contraction in our metric space.

Theorem 4.26. $\Phi_{F,\mathcal{P}}$ is a contraction relative to the metric d_3.

Proof. We will show that $d_3(\Phi_{F,\mathcal{P}}(I), \Phi_{F,\mathcal{P}}(J)) \leq \frac{1}{2} \cdot d_3(I, J)$. The claim follows trivially in case $I = J$, so assume $I \neq J$ and $d_3(I, J) \leq \left(\frac{1}{2}\right)^n$ for some $n \geq 0$. Then all of $d_1(I, M), d_1(J, M), d_2(I, J), \rho(I)$ and $\rho(J)$ are $\leq \left(\frac{1}{2}\right)^n$. From Lemma 4.17 we get that $d_1(\Phi_{F,\mathcal{P}}(I), M)$ and $d_1(\Phi_{F,\mathcal{P}}(J), M)$ are $\leq \left(\frac{1}{2}\right)^{n+1}$, so it remains to show:

(i) $\rho(\Phi_{F,\mathcal{P}}(I)) \leq \left(\frac{1}{2}\right)^{n+1}$,

(ii) $\rho(\Phi_{F,\mathcal{P}}(J)) \leq \left(\frac{1}{2}\right)^{n+1}$,

(iii) $d_2(\Phi_{F,\mathcal{P}}(I), \Phi_{F,\mathcal{P}}(J)) \leq \left(\frac{1}{2}\right)^{n+1}$

The argument for (ii) will be the same as the argument for (i). Hence we will investigate the cases (i) and (iii):

(i) To prove $\rho(\Phi_{F,\mathcal{P}}(I)) \leq \left(\frac{1}{2}\right)^{n+1}$, it is necessary to show that $\Phi_{F,\mathcal{P}}(I)$ correctly asserts A for each ground atom $A \notin Neg^*_{\mathcal{P}}$ with $l_2(A) \leq n$. Suppose, to the contrary, that there is a ground atom A with $l_2(A) \leq n$ such that $\Phi_{F,\mathcal{P}}(I)(A) = \top_3$ but $M_2^\top(A) = \bot_2$. We will derive a contradiction. $\Phi_{F,\mathcal{P}}(I)(A) = \top_3$ if and only if there exists a clause $A \leftarrow B_1, \ldots, B_m \in \mathit{ground}(\mathcal{P})$ such that $I(B_i) = \top_3$ for all i with $1 \leq i \leq m$. Since $M_2^\top$ is a model of $\mathcal{P}$ and $M(A) = \bot_2$, there exists a k with $1 \leq k \leq m$ such that $M_2^\top(B_1 \wedge \ldots \wedge B_{k-1}) = \top_2$ but $M_2^\top(B_k) = \bot_2$. Note that $l(B_k) < l(A) \leq n$ by the acceptability of $\mathcal{P}$ w.r.t. $M_2^\top$. There are 2 cases to consider:

   (a) If $B_k \in Neg^*_{\mathcal{P}}$, then since $d_1(I, M) \leq \left(\frac{1}{2}\right)^n$, I and M agree on ground literals in $Neg^*_{\mathcal{P}}$ of level $< n$. Hence $I(B_k) = M(B_k) = \top_3$ and by Prop. 4.7(1) we obtain $M_2^\top(B_k) = \top_2$. But this contradicts the fact that $M_2^\top(B_k) = \bot_2$.

   (b) If $B_k \notin Neg^*_{\mathcal{P}}$, then B_k must be a positive literal. Since $\rho(I) \leq \left(\frac{1}{2}\right)^n$, I must correctly assert each ground atom not in $Neg^*_{\mathcal{P}}$ whose level is less than n; in particular, I correctly asserts B_k. But this is impossible since $I(B_k) = \top_3$ but $M_2^\top(B_k) = \bot_2$.

(iii) We must show for each ground atom $A \notin Neg^*_{\mathcal{P}}$ with $l_2(A) \leq n$ that $\Phi_{F,\mathcal{P}}(I)(A) = \Phi_{F,\mathcal{P}}(J)(A) \neq u$. If $\Phi_{F,\mathcal{P}}(I)(A) = \Phi_{F,\mathcal{P}}(J)(A) = \bot_3$, then the condition is satisfied. So let's assume, without loss of generality, that $\Phi_{F,\mathcal{P}}(I)(A) \neq \bot_3$. Then there is some clause $A \leftarrow B_1, \ldots, B_m$ in $\mathit{ground}(\mathcal{P})$ such that $I(B_i) \neq \bot_3$ for all i with $1 \leq i \leq m$. We will now consider two cases:

   (a) If $M_2^\top(B_1 \wedge \cdots \wedge B_m) = \top_2$, then since $\mathcal{P}$ is acceptable w.r.t. $M_2^\top$, we obtain
   $$l(B_i) < l(A) \leq n$$
   for every i with $1 \leq i \leq m$. Take some j with $1 \leq j \leq m$. Two cases can occur:

      i. If $B_j \in Neg^*_{\mathcal{P}}$, then from $d_1(I, M) \leq \left(\frac{1}{2}\right)^n$ and $d_1(J, M) \leq \left(\frac{1}{2}\right)^n$ and $l(B_j) < n$ we obtain $I(B_j) = J(B_j) = M(B_j) = \top_3$.

ii. If $B_j \notin Neg^*_{\mathcal{P}}$, then from $d_2(I,J) \leq \left(\frac{1}{2}\right)^n$ we obtain $I(B_j) = J(B_j) \neq u$. Further, since $I(B_j) \neq \bot_3$, it must be the case that $I(B_j) = J(B_j) = \top_3$.

In both cases we have $I(B_j) = J(B_j) = \top_3$ and since the choice of j was arbitrary, we can conclude that $I(B_1 \wedge \cdots \wedge B_m) = J(B_1 \wedge \cdots \wedge B_m) = \top_3$. Consequently,

$$\Phi_{F,\mathcal{P}}(I)(A) = \Phi_{F,\mathcal{P}}(J)(A) = \top_3$$

and we are done.

(b) If $M_2^\top(B_1 \wedge \cdots \wedge B_m) = \bot_2$, then we can find the least index k with $1 \leq k \leq m$ such that $M_2^\top(B_k) = \bot_2$. Hence also $M(B_k) = \bot_3$ and since $\mathcal{P}$ is acceptable w.r.t. $M_2^\top$, we obtain $l(B_k) < l(A)$. We will derive a conflict, considering two cases:

i. If $B_k \in Neg^*_{\mathcal{P}}$, then we arrive at a conflict with $d_1(I,M) \leq \left(\frac{1}{2}\right)^n$ because $I(B_k) \neq \bot_3$ while $M(B_k) = \bot_3$.

ii. If $B_k \notin Neg^*_{\mathcal{P}}$, then from $d_2(I,J) \leq \left(\frac{1}{2}\right)^n$ we obtain $I(B_k) \neq u$. From the assumption we further obtain $I(B_k) \neq \bot_3$, so the only possibility that remains is $I(B_k) = \top_3$. However, this is in conflict with $\rho(I) \leq \left(\frac{1}{2}\right)^n$ because $M(B_k) \neq \top_2$. □

To summarize, given an acceptable program $\mathcal{P}$, we found a mapping d_3 that is a metric on the space of all three-valued interpretations. Moreover, we showed that this space is a complete metric space. We also proved that the Fitting operator $\Phi_{F,\mathcal{P}}$ is a contraction in this space, so by applying the Banach Contraction Theorem we can conclude it has a unique fixed point that can be computed by iterating $\Phi_{F,\mathcal{P}}$ up to ω times starting from any three-valued interpretation.

Corollary 4.27. Let $\mathcal{P}$ be an acceptable logic program. Then $\Phi_{F,\mathcal{P}}$ has a unique fixed point that can be reached by iterating it up to ω times starting from any interpretation.

Proof. Follows by Propositions 4.23 and 4.25 and Theorems 4.26 and 4.4. □

Example 4.28. Consider the following program:

$$\begin{aligned} \mathcal{P}_2: \quad & p \leftarrow r, q. \\ & q \leftarrow r, p. \end{aligned}$$

Let $l(p) = l(q) = 1$, $l(r) = 0$ and $M_2^\top = \emptyset$. We can see that $l(p) > l(r)$ and $l(q) > l(r)$. Hence, $\mathcal{P}_2$ is acceptable with respect to the level mapping l and the model $M_2^\top$. In the following we demonstrate how iterating the operator from any interpretation always yields its unique fixed point.

First consider the interpretation $I_0 = \langle \{ q, r \}, \{ p \} \rangle$. By iterating the Fitting operator starting from I_0 we obtain the following results:

$$\begin{aligned} I_1 &= \Phi_{F,\mathcal{P}_2}(I_0) = \langle \{ p \}, \{ q, r \} \rangle \\ I_2 &= \Phi_{F,\mathcal{P}_2}(I_1) = \langle \emptyset, \{ p, q, r \} \rangle = \Phi_{F,\mathcal{P}_2}(I_2) \end{aligned}$$

If instead we choose the interpretation $I_0' = \langle \{ p \}, \emptyset \rangle$, then the results are as follows:

$$I_1' = \Phi_{F,\mathcal{P}_2}(I_0') = \langle \emptyset, \{ r \} \rangle$$
$$I_2' = \Phi_{F,\mathcal{P}_2}(I_1') = \langle \emptyset, \{ p, q, r \} \rangle = I_2 = \Phi_{F,\mathcal{P}_2}(I_2)$$

We can see that by the previous result, $I_2 = I_2'$ is the unique fixed point of $\Phi_{F,\mathcal{P}_2}$.

4.4 Stenning and van Lambalgen Operator

Let us first recall the definition of Stenning and van Lambalgen Operator.

Let I be an interpretation and $\mathcal{P}$ be an extended program. The *Stenning and van Lambalgen immediate consequence operator* is defined as follows: $\Phi_{SvL,\mathcal{P}}(I) = \left\langle J^{\top}, J^{\bot} \right\rangle$, where

$$J^{\top} = \{ A \mid \text{there exists } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ with } I(\mathit{Body}) = \top \} \text{ and}$$
$$J^{\bot} = \{ A \mid \text{there exists } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ and}$$
$$\text{for all } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ we find } I(\mathit{Body}) = \bot \}$$

Even though the difference between the Fitting operator and the Stenning and van Lambalgen operator is only subtle on the first glance, the Stenning and van Lambalgen operator does not always have a unique fixed point for an acceptable program. Here is a counterexample:

Example 4.29. Consider the acceptable program from Example 4.28:

$$\mathcal{P}_2 : \quad p \leftarrow r, q.$$
$$q \leftarrow r, p.$$

Now let $I_1 = \langle \emptyset, \emptyset \rangle$ and $I_2 = \langle \emptyset, \{ p, q \} \rangle$. We find $\Phi_{SvL,\mathcal{P}_2}(I_1) = I_1$ and $\Phi_{SvL,\mathcal{P}_2}(I_2) = I_2$, so I_1 and I_2 are two fixed points of $\Phi_{SvL,\mathcal{P}_2}$. Consequently, $\Phi_{SvL,\mathcal{P}_2}$ cannot be a contraction because from Banach Contraction Theorem it follows that a contraction always has a unique fixed point.

We can see that the problem in $\mathcal{P}_2$ is the cycle between p and q. Indeed, it turns out that for the smaller class of acyclic programs, the Stenning and van Lambalgen operator always has a unique fixed point. To prove this result, we first need to show that the space of three-valued interpretations with a metric induced by some (total) level mapping always forms a complete metric space. Then we will prove that for any program $\mathcal{P}$ that is acyclic with respect to some level mapping l, $\Phi_{SvL,\mathcal{P}}$ is a contraction with respect to the metric induced by l. This will allow us to use the Banach Contraction Theorem to conclude that $\Phi_{SvL,\mathcal{P}}$ has a unique fixed point that can be computed by iterating $\Phi_{SvL,\mathcal{P}}$ starting from any three-valued interpretation.

Let's start by showing that the space of three-valued interpretations using a metric induced by some level mapping is a complete metric space.

Proposition 4.30. The space of three-valued interpretations using a metric d_l induced by some level mapping l is a complete metric space.

Proof. Suppose $\{S_k\}_{k=1}^{\infty}$ is a Cauchy sequence of interpretations. Then for any $n \in \mathbb{N}$ there must be some $K \in \mathbb{N}$ such that for any $k_1, k_2 \geq K$ we have

$$d_l(S_{k_1}, S_{k_2}) \leq \left(\frac{1}{2}\right)^{n+1}$$

Let K_n be the least such K for every $n \in \mathbb{N}$. This definition implies

$$K_{n_1} \leq K_{n_2} \text{ for any } n_1, n_2 \in \mathbb{N} \text{ with } n_1 \leq n_2.$$

Let us define our limit interpretation S for any atom A as $S(A) = S_{K_l}(A)$ where $l(A) = l$.

We now need to prove that for any $\varepsilon > 0$ there is some $K \in \mathbb{N}$ such that for any $k \geq K$ we have $d_l(S, S_k) \leq \varepsilon$. Choose some $\varepsilon > 0$ and let $n \in \mathbb{N}$ be such that $\left(\frac{1}{2}\right)^{n+1} \leq \varepsilon$. We will prove $d_l(S, S_k) \leq \left(\frac{1}{2}\right)^{n+1}$ for any $k \geq K_n$ from which the claim will follow.

Take some atom A with $l(A) = l \leq n$. Then $K_l \leq K_n$ and hence by the definition of K_l we have $d_l(S_{K_l}, S_{K_n}) \leq \left(\frac{1}{2}\right)^{l+1}$. Consequently, by the definition of d_l we have $S(A) = S_{K_l}(A) = S_{K_n}(A)$. Further, for any $k \geq K_n$ we have $d_l(S_{K_n}, S_k) \leq \left(\frac{1}{2}\right)^{n+1}$, so we obtain $S(A) = S_{K_n}(A) = S_k(A)$ and therefore also

$$d_l(S, S_k) \leq \left(\frac{1}{2}\right)^{n+1} \quad . \qquad \square$$

We can now turn to the main result, i.e. that for any acyclic program $\mathcal{P}$, $\Phi_{SvL,\mathcal{P}}$ is a contraction relative to the metric induced by the level mapping with respect to which $\mathcal{P}$ is acyclic.

Theorem 4.31. Let $\mathcal{P}$ be an acyclic logic program with respect to the level mapping l. Then $\Phi_{SvL,\mathcal{P}}$ is a contraction relative to the metric d_l induced by the level mapping l.

Proof. We will show

$$d_l(\Phi_{SvL,\mathcal{P}}(I), \Phi_{SvL,\mathcal{P}}(J)) \leq \frac{1}{2} \cdot d_l(I, J) \ .$$

If $I = J$, then $\Phi_{SvL,\mathcal{P}}(I) = \Phi_{SvL,\mathcal{P}}(J)$, so $d_l(\Phi_{SvL,\mathcal{P}}(I), \Phi_{SvL,\mathcal{P}}(J)) = d_l(I, J) = 0$ and we are finished.

If $I \neq J$, then since l is total, we obtain $d_l(I, J) = \left(\frac{1}{2}\right)^n$ for some $n \in \mathbb{N}$. We will show that $d_l(\Phi_{SvL,\mathcal{P}}(I), \Phi_{SvL,\mathcal{P}}(J)) \leq \left(\frac{1}{2}\right)^{n+1}$, i.e. that for all ground atoms $A \in \mathit{ground}(\mathcal{P})$ with $l(A) \leq n$ we have $\Phi_{SvL,\mathcal{P}}(I)(A) = \Phi_{SvL,\mathcal{P}}(J)(A)$.

Let's take some A with $l(A) \leq n$ and let $\mathcal{C}_A$ be the set of all clauses in $\mathit{ground}(\mathcal{P})$ with A in the head. Since $\mathcal{P}$ is acyclic, for any clause $A \leftarrow B_1, \ldots, B_m$ from $\mathcal{C}_A$ we obtain that for all $1 \leq i \leq m$

$$l(B_i) < l(A) \leq n \ .$$

We know that $d_l(I, J) \leq \left(\frac{1}{2}\right)^n$, so $I(B_i) = J(B_i)$ for all $1 \leq i \leq m$. Therefore, since the choice of the clause was arbitrary, I and J interpret identically all bodies of clauses with A in the head. Consequently, $\Phi_{SvL,\mathcal{P}}(I)(A) = \Phi_{SvL,\mathcal{P}}(J)(A)$ as desired. $\square$

To summarize, although the Stenning and van Lambalgen operator of an acceptable program may not be a contraction and may even have multiple fixed points, for the smaller class of acyclic programs it is guaranteed to be a contraction. Similarly as for the Fitting operator, by the Banach Contraction Theorem it then has a unique fixed point that can be computed by iterating the operator up to ω times starting from any three-valued interpretation.

Corollary 4.32. Let $\mathcal{P}$ be an acyclic logic program. Then $\Phi_{SvL,\mathcal{P}}$ has a unique fixed point that can be reached by iterating it up to ω times starting from any interpretation.

Proof. Follows by Propositions 4.13 and 4.30 and Theorems 4.31 and 4.4. □

Example 4.33. Consider the following program:

$$\begin{aligned} \mathcal{P}_3 : \quad & p \leftarrow q, r. \\ & q \leftarrow \neg r. \\ & r \leftarrow \top. \end{aligned}$$

Let $l(p) = 2$, $l(q) = 1$ and $l(r) = 0$. We can see that $l(p) > l(q)$, $l(p) > l(r)$, $l(q) > l(r)$. Hence, $\mathcal{P}_3$ is acyclic with respect to the level mapping l. In the following, we demonstrate how iterating the operator from any interpretation always yields its unique fixed point.

First consider the interpretation $I_0 = \langle \{ q, r \}, \{ p \} \rangle$. By iterating the Stenning and van Lambalgen operator starting from I_0 we obtain the following results:

$$\begin{aligned} I_1 &= \Phi_{SvL,\mathcal{P}_3}(I_0) = \langle \{ p, r \}, \{ q \} \rangle \\ I_2 &= \Phi_{SvL,\mathcal{P}_3}(I_1) = \langle \{ r \}, \{ p, q \} \rangle = \Phi_{SvL,\mathcal{P}_3}(I_2) \end{aligned}$$

If instead we choose the interpretation $I'_0 = \langle \{ p \}, \emptyset \rangle$, then the results are as follows:

$$\begin{aligned} I'_1 &= \Phi_{SvL,\mathcal{P}_3}(I'_0) = \langle \{ r \}, \emptyset \rangle \\ I'_2 &= \Phi_{SvL,\mathcal{P}_3}(I'_1) = \langle \{ r \}, \{ q \} \rangle \\ I'_3 &= \Phi_{SvL,\mathcal{P}_3}(I'_2) = \langle \{ r \}, \{ p, q \} \rangle = I_2 = \Phi_{SvL,\mathcal{P}_3}(I_2) \end{aligned}$$

We can see that by the previous result, $I_2 = I'_3$ is the unique fixed point of $\Phi_{SvL,\mathcal{P}_3}$.

4.5 Discussion

We showed that for any acceptable program the Fitting operator is a contraction. By the Banach Contraction Theorem this implies that it has a unique fixed point and this fixed point can be reached by iterating the operator at most ω times starting from any three-valued interpretation.

In [AP93], the authors also proved, without using metric methods, that for any acceptable program the Fitting operator has a unique fixed point. They also showed that this fixed point is a total interpretation whose positive part

coincides with the unique fixed point of the two-valued $T_{\mathcal{P}}$ operator (see Corollary 6.9 on p. 33 in [AP93]). Further, since every stable model [GL88] is a fixed point of $\Phi_{F,\mathcal{P}}$, the fixed point is also a unique stable (and well-founded [GRS91]) model.

By contrast, we showed that the Stenning and van Lambalgen operator need not be a contraction for any acceptable program and may also have multiple fixed points. Then we proved that for the smaller class of acyclic programs, the Stenning and van Lambalgen operator is guaranteed to be a contraction and hence by Banach Contraction Theorem has a unique fixed point. However, it should be noted that this fixed point still need not be total, as is shown in the following example:

Example 4.34. Consider the following program:

$$\mathcal{P}_3 : \quad p \leftarrow q.$$

$\mathcal{P}_3$ is acyclic with respect to the level mapping $l(p) = 1, l(q) = 0$ and its unique fixed point is $I = \langle \emptyset, \emptyset \rangle$.

Chapter 5

Three-Valued Semantics for Logic Programs

In recent years, the Fitting semantics for logic programs has not been used much. It has been overtaken in interest by the well-founded semantics [GRS91] and stable model semantics [GL88]. The latter extends the former in a well-understood manner, and provides a two-valued semantics for logic programs. Both capture transitive closure and other recursive rule behaviour and, thus, are useful for programming. However, there are trade-offs between the Fitting semantics and well-founded semantics. The ability of well-founded semantics to capture properties like graph reachability means that it cannot be modelled by a finite first-order theory such as completion. Well-founded semantics also has a higher complexity than the Fitting semantics. The relationship of the Fitting semantics and the well-founded semantics is brought forward in [HW02] using level mappings. Hence, in this chapter we show the relationship of Stenning and van Lambalgen semantics based on its immediate consequence operator with Fitting semantics and well-founded semantics.

We will be using a more general notion of a level mapping in this chapter:

Definition 5.1 (Level Mapping)**.** Let $I = \langle I^\top, I^\bot \rangle$ be an interpretation and $\mathcal{P}$ a logic program. An *I-partial level mapping* for $\mathcal{P}$ is a partial mapping $l : B_\mathcal{P} \to \alpha$ where the domain $dom(l) = I^\top \cup I^\bot$ and α is a countable ordinal. We extend the definition of level mapping to literals by setting $l(\neg A) = l(A)$. A *total level mapping* is a total mapping $l : B_\mathcal{P} \to \alpha$ for some countable ordinal α.

5.1 Stenning and van Lambalgen Semantics

Let us now recall the definition of the Stenning and van Lambalgen operator.

Let I be an interpretation and $\mathcal{P}$ be an extended program. The *Stenning and van Lambalgen immediate consequence operator* is defined as follows:

$\Phi_{SvL,\mathcal{P}}(I) = \langle J^\top, J^\bot \rangle$, where

$$J^\top = \{\, A \mid \text{there exists } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ with } I(\mathit{Body}) = \top \,\} \text{ and}$$
$$J^\bot = \{\, A \mid \text{there exists } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ and for all } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ we find } I(\mathit{Body}) = \bot \,\}$$

The operator $\Phi_{SvL,\mathcal{P}}$ is monotonic (Proposition 3.21) and so by Proposition 3.11 it has a least fixed point that is characterized by transfinite induction – iterating $\Phi_{SvL,\mathcal{P}}$ starting from the empty interpretation must eventually yield a fixed point.

More formally, let M be the *least fixed point of* $\Phi_{SvL,\mathcal{P}}$ and let

$$M_0 = \langle \emptyset, \emptyset \rangle$$
$$M_\alpha = \Phi_{SvL,\mathcal{P}}(M_{\alpha-1}) \text{ for every non-limit ordinal } \alpha > 0$$
$$M_\alpha = \bigcup_{\beta<\alpha} M_\beta \text{ for every limit ordinal } \alpha$$

Then for some ordinal γ it holds that $M = M_\gamma$. The Stenning and van Lambalgen (SvL) semantics for logic programs is given by this least fixed point:

Definition 5.2 (Stenning and van Lambalgen Model)**.** The least fixed point of $\Phi_{SvL,\mathcal{P}}$ is called the *Stenning and van Lambalgen (SvL) model* of $\mathcal{P}$.

Now we give a characterization of the Stenning and van Lambalgen semantics using level mappings.

Definition 5.3. Let $\mathcal{P}$ be a logic program, $I = \langle I^\top, I^\bot \rangle$ be a model of $\mathcal{P}$ and l be an I-partial level mapping for $\mathcal{P}$. *$\mathcal{P}$ satisfies (SvL) with respect to I and l* if for every $A \in \mathit{dom}(l)$ one of the following conditions is satisfied:

SvLi $A \in I^\top$ and there exists a clause $A \leftarrow B_1, \ldots, B_m$ in $\mathit{ground}(\mathcal{P})$ such that $I(B_i) = \top$ and $l(A) > l(B_i)$ for all i.

SvLii $A \in I^\bot$ and there exists a clause $A \leftarrow B_1, \ldots, B_m$ in $\mathit{ground}(\mathcal{P})$ and for each such clause there exists an i with $I(B_i) = \bot$ and $l(A) > l(B_i)$.

If $A \in \mathit{dom}(l)$ satisfies (SvLi), then we say that A satisfies (SvLi) with respect to I and l, and similarly if $A \in \mathit{dom}(l)$ satisfies (SvLii).

Theorem 5.4. Let $\mathcal{P}$ be a logic program with the SvL model M. Then M is the greatest model among all models I for which there exists an I-partial level mapping l for $\mathcal{P}$ such that $\mathcal{P}$ satisfies (SvL) with respect to I and l.

Proof. Let $M = \langle M^\top, M^\bot \rangle$ be the SvL model of $\mathcal{P}$ and let M_α be defined as above (where α is an ordinal).

We define the M-partial level mapping l_M as follows: $l_M(A) = \alpha$, where α is the least ordinal such that A is not undefined in M_α. The proof will be established by showing the following facts: (1) $\mathcal{P}$ satisfies (SvL) with respect to M and l_M. (2) If I is a model of $\mathcal{P}$ and l_I an I-partial level mapping such that $\mathcal{P}$ satisfies (SvL) with respect to I and l_I, then $I \subseteq M$.

1. Let $A \in \mathit{dom}(l_M)$ and $l_M(A) = \alpha$. Now we consider two cases:

(a) If $A \in M^\top$, then $A \in M^\top_\alpha$, hence there exists a clause $A \leftarrow B_1, \ldots, B_m$ in *ground*($\mathcal{P}$) such that $M_{\alpha-1}(B_1, \ldots, B_m) = \top$. Thus, for all B_i we have that $M_{\alpha-1}(B_i) = \top$ and hence $l_M(B_i) < \alpha$ and $M(B_i) = \top$ for all i. Consequently, A satisfies (SvLi) with respect to M and l_M.

(b) If $A \in M^\bot$, then $A \in M^\bot_\alpha$, hence there exists a clause $A \leftarrow B_1, \ldots, B_m$ in *ground*($\mathcal{P}$) and for all such clauses there exists a B_i with $M_{\alpha-1}(B_i) = \bot$. So $l_M(B_i) < \alpha$ and $M(B_i) = \bot$. Consequently, A satisfies (SvLii) with respect to M and l_M, and we have established that fact (1) holds.

2. We show via transfinite induction on $\alpha = l_I(A)$ that whenever $A \in I^\top$ (respectively, $A \in I^\bot$), then $A \in M^\top$ (respectively, $A \in M^\bot$).

   1° If $l_I(A) = 0$, then $A \in I^\top$ implies that A occurs as the head of a fact in *ground*($\mathcal{P}$), hence $A \in M^\top_1 \subseteq M^\top$. There cannot be an atom $A \in I^\bot$ with $l_I(A) = 0$ because then for some literal B we would have $l_I(B) < 0$ which is a contradiction.

   2° Assume now that the induction hypothesis holds for all $B \in \mathcal{B}_\mathcal{P}$ with $l_I(B) < \alpha$. We consider two cases:

      i. If $A \in I^\top$, then it satisfies (SvLi) with respect to I and l_I. Hence there is a clause $A \leftarrow B_1, \ldots, B_m$ in *ground*($\mathcal{P}$) such that $I(B_1, \ldots, B_m) = \top$ and $l_I(B_i) < \alpha$ for all i. Hence $M(B_1, \ldots, B_m) = \top$ by inductive hypothesis, and since M is a model of $\mathcal{P}$, we obtain $A \in M^\top$.

      ii. If $A \in I^\bot$, then it satisfies (SvLii) with respect to I and l_I. Hence there is a clause $A \leftarrow B_1, \ldots, B_m$ in *ground*($\mathcal{P}$) and for all such clauses there exists an i with $I(B_i) = \bot$ and $l_I(A) > l_I(B_i)$. Hence for all these B_i, $M(B_i) = \bot$ by induction hypothesis and consequently for all clauses $A \leftarrow B_1, \ldots, B_m$ in *ground*($\mathcal{P}$) we get $M(B_1, \ldots, B_m) = \bot$. Because M is a fixed point of $\Phi_{SvL,\mathcal{P}}$, we find that $A \in M^\bot$. This establishes fact (2) and concludes the proof. □

Corollary 5.5. A logic program $\mathcal{P}$ has a total SvL model if and only if there is a total model I of $\mathcal{P}$ and a (total) level mapping l for $\mathcal{P}$ such that $\mathcal{P}$ satisfies (SvL) with respect to I and l.

5.2 Fitting and Well-Founded Semantics

Let us recall back the definition of Fitting operator from [Fit85].

Let I be an interpretation and $\mathcal{P}$ a program. The *Fitting immediate consequence operator* is defined as follows: $\Phi_{F,\mathcal{P}}(I) = \langle J^\top, J^\bot \rangle$, where

$$J^\top = \{\, A \mid \text{there exists } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ with } I(\mathit{Body}) = \top \,\} \text{ and}$$
$$J^\bot = \{\, A \mid \text{for all } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ we find } I(\mathit{Body}) = \bot \,\}.$$

The operator $\Phi_{F,\mathcal{P}}$ is monotonic [Fit85] and so we can compute its least fixed point by iterating $\Phi_{F,\mathcal{P}}$ starting from the empty interpretation. Similarly as in

the case of the Stenning and van Lambalgen semantics, the Fitting semantics for logic programs is given by this least fixed point:

Definition 5.6 (Fitting Model)**.** The least fixed point of $\Phi_{F,\mathcal{P}}$ is called *Fitting model* of $\mathcal{P}$.

A level mapping characterization of the Fitting semantics, similar to the one for SvL semantics above, has been given in [HW02]:

Definition 5.7. Let $\mathcal{P}$ be a logic program, $I = \langle I^\top, I^\bot \rangle$ be a model of $\mathcal{P}$ and l be an I-partial level mapping for $\mathcal{P}$. *$\mathcal{P}$ satisfies Fitting (F) with respect to I and l* if for every $A \in \mathit{dom}(l)$ one of the following conditions is satisfied:

Fi $A \in I^\top$ and there exists a clause $A \leftarrow B_1, \ldots, B_m$ in *ground*($\mathcal{P}$) such that $I(B_i) = \top$ and $l(A) > l(B_i)$ for all i.

Fii $A \in I^\bot$ and for each clause $A \leftarrow B_1, \ldots, B_m$ in *ground*($\mathcal{P}$) there exists an i with $I(B_i) = \bot$ and $l(A) > l(B_i)$.

Notice that the condition (Fi) is exactly same as the condition (SvLi) and the condition (Fii) is strictly weaker than the condition (SvLii).

Theorem 5.8. Let $\mathcal{P}$ be a logic program with the Fitting model M. Then M is the greatest model among all models I for which there exists an I-partial level mapping l for $\mathcal{P}$ such that $\mathcal{P}$ satisfies (F) with respect to I and l.

Proof. See [HW02]. □

Corollary 5.9. A logic program $\mathcal{P}$ has a total Fitting model if and only if there is a total model I of $\mathcal{P}$ and a (total) level mapping l for $\mathcal{P}$ such that $\mathcal{P}$ satisfies (Fitting) with respect to I and l.

Corollary 5.10. Let $\mathcal{P}$ be a logic program with SvL model M_{SvL} and Fitting model M_F. Then $M_{SvL} \subseteq M_F$.

Proof. Follows from Theorems 5.4 and 5.8 and the fact that the condition (Fi) is exactly same as the condition (SvLi) and the condition (Fii) is strictly weaker than the condition (SvLii). □

Hitzler and Wendt in [HW02] also gave a level mapping characterization of the well-founded semantics as follows:

Definition 5.11. Let $\mathcal{P}$ be a logic program, $I = \langle I^\top, I^\bot \rangle$ be a model of $\mathcal{P}$ and l be an I-partial level mapping for $\mathcal{P}$. *$\mathcal{P}$ satisfies (WF) with respect to I and l* if for every $A \in \mathit{dom}(l)$ one of the following conditions is satisfied:

WFi $A \in I^\top$ and there exists a clause $A \leftarrow B_1, \ldots, B_m$ in *ground*($\mathcal{P}$) such that $I(B_i) = \top$ and $l(A) > l(B_i)$ for all i.

WFii $A \in I^\bot$ and for each clause $A \leftarrow A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m$ in *ground*($\mathcal{P}$), at least one of the following conditions holds:

WFiia there exists i with $1 \leq i \leq n$, $A_i \in I^\bot$ and $l(A) \geq l(A_i)$,

WFiib there exists j with $1 \leq j \leq m$, $B_j \in I^\top$ and $l(A) > l(B_j)$.

Theorem 5.12. Let $\mathcal{P}$ be a logic program with the well-founded model M. Then M is the greatest model among all models I for which there exists an I-partial level mapping l for $\mathcal{P}$ such that $\mathcal{P}$ satisfies (WF) with respect to I and l.

Proof. See [HW02]. □

Corollary 5.13. Let $\mathcal{P}$ be a logic program with F model M_F, well-founded model M_{WF}. Then $M_F \subseteq M_{WF}$.

Proof. See [HW02]. □

Corollary 5.14. Let $\mathcal{P}$ be a logic program with SvL model M_{SvL}, F model M_F, well-founded model M_{WF}. Then $M_{SvL} \subseteq M_F \subseteq M_{WF}$.

Proof. Follow immediately from Corollary 5.10 and Corollary 5.13. □

5.3 Discussion

In [HW02], Hitlzler and Wendt gave a new characterizations of the Fitting semantics and the well-founded semantics using level mappings and we just gave a similar characterization for SvL semantics based on Stenning and van Lambalgen operator [SvL08]. The result shows that compared to the Fitting and well-founded model, the Stenning and van Lambalgen model is always the smallest. This means that the SvL semantics is more sceptical since the SvL model makes potentially more atoms undefined than the other semantics and hence, allows to conclude less from the same program.

Chapter 6

Consequence Operators under Łukasiewicz Semantics

In this chapter we discuss Fitting operator [Fit85] and Stenning van Lambalgen operator [SvL08] under the Łukasiewicz semantics [Łuk20]. First we introduce the Łukasiewicz, Kleene and Fitting three-valued logics and then we show that replacing the Fitting semantics by the Łukasiewicz semantics does not change the behavior of the operators. Moreover, in difference to the Fitting semantics, the model intersection property holds under the Łukasiewicz semantics and in addition the fixed points of the Fitting and the Stenning and van Lambalgen operators are models of the program itself.

6.1 Three-Valued Logics

In this section, we introduce three different three-valued logics, namely the Łukasiewicz three-valued logic [Łuk20], the Kleene strong three-valued logic [Kle52] and the Fitting three-valued logic [Fit85] which was defined by Fitting as a semantics for completed logic programs. We show the differences between these three-valued logics.

In 1920, the Polish philosopher Łukasiewicz introduced the first three-valued logic [Łuk20]. In his logic, the truth values are not only true and false, but there exists a third, intermediate value. A formula is allowed to be neither true nor false. We can interpret the intermediate truth value as possibility: the truth value is not decided yet but possibly decided at some later time. As before, we symbolize truth- and falsehood by $\top$ and $\bot$, respectively, and we call the third truth value *undecided* and use the symbol u to denote it.

Łukasiewicz used the following principles and definitions to assign values to formulas, where $\equiv$ denotes semantic equivalence:

1. The principles of identity and non-identity:
   $(\bot \leftrightarrow_{\text{Ł}} \bot) \equiv (\top \leftrightarrow_{\text{Ł}} \top) \equiv \top$, $(\top \leftrightarrow_{\text{Ł}} \bot) \equiv (\bot \leftrightarrow_{\text{Ł}} \top) \equiv \bot$,
   $(\bot \leftrightarrow_{\text{Ł}} u) \equiv (u \leftrightarrow_{\text{Ł}} \bot) \equiv (\top \leftrightarrow_{\text{Ł}} u) \equiv (u \leftrightarrow_{\text{Ł}} \top) \equiv u$, $(u \leftrightarrow_{\text{Ł}} u) \equiv \top$.

2. The principles of implication:
   $(\bot \leftarrow_{\text{Ł}} \bot) \equiv (\top \leftarrow_{\text{Ł}} \bot) \equiv (\top \leftarrow_{\text{Ł}} \top) \equiv \top$, $(\bot \leftarrow_{\text{Ł}} \top) \equiv \bot$,
   $(u \leftarrow_{\text{Ł}} \bot) \equiv (\top \leftarrow_{\text{Ł}} u) \equiv (u \leftarrow_{\text{Ł}} u) \equiv \top$, $(\bot \leftarrow_{\text{Ł}} u) \equiv (u \leftarrow_{\text{Ł}} \top) \equiv u$.

3. The definitions of negation, disjunction and conjunction:
   $\neg A \equiv (\bot \leftarrow_{\text{Ł}} A)$, $A \vee B \equiv (B \leftarrow_{\text{Ł}} (B \leftarrow_{\text{Ł}} A))$, $A \wedge B \equiv \neg(\neg A \vee \neg B)$.

Later, in 1952, Kleene proposed an alternative three-valued logic with the truth values true, false, and undefined [Kle52]. He distinguishes between weak and strong three-valued logics.

For weak three-valued logics, Kleene uses the propositional connectives in the weak sense. It means $F \vee G \equiv F \wedge G \equiv F \leftarrow G \equiv F \leftrightarrow G \equiv u$ when at least one of the formulae F, G has the truth value undefined. The Kleene weak three-valued is unsubtle and not interesting for us, so we will focus on Kleene strong three-valued logic.

The Kleene strong three-valued logic is similar to the Łukasiewicz logic, but differs in the semantics of implication and equivalence between two undefined formulae. More specifically, when the formulae F, G are both undefined, then both the implication $F \leftarrow G$ and the equivalence $F \leftrightarrow G$ under Łukasiewicz logic have the truth value true whereas under Kleene strong three-valued logic they are undefined.

In particular, Kleene's strong three-valued logic is based on the following principles and definitions, where we have underlined the differences to Łukasiewicz logic:

1. The principles of identity and non-identity:
   $(\bot \leftrightarrow_{\text{K}} \bot) \equiv (\top \leftrightarrow_{\text{K}} \top) \equiv \top$, $(\top \leftrightarrow_{\text{K}} \bot) \equiv (\bot \leftrightarrow_{\text{K}} \top) \equiv \bot$,
   $(\bot \leftrightarrow_{\text{K}} u) \equiv (u \leftrightarrow_{\text{K}} \bot) \equiv (\top \leftrightarrow_{\text{K}} u) \equiv (u \leftrightarrow_{\text{K}} \top) \underline{\equiv (u \leftrightarrow_{\text{K}} u)} \equiv u$

2. The principles of implication:
   $(\bot \leftarrow_{\text{K}} \bot) \equiv (\top \leftarrow_{\text{K}} \bot) \equiv (\top \leftarrow_{\text{K}} \top) \equiv \top$, $(\bot \leftarrow_{\text{K}} \top) \equiv \bot$,
   $(u \leftarrow_{\text{K}} \bot) \equiv (\top \leftarrow_{\text{K}} u) \equiv \top$, $(\bot \leftarrow_{\text{K}} u) \equiv (u \leftarrow_{\text{K}} \top) \ \underline{\equiv (u \leftarrow_{\text{K}} u)} \equiv u$

3. The definitions of negation, disjunction and conjunction:
   $\neg A \equiv (\bot \leftarrow_{\text{K}} A)$, $A \vee B \equiv (B \leftarrow_{\text{K}} (B \leftarrow_{\text{K}} A))$, $A \wedge B \equiv \neg(\neg A \vee \neg B)$.

Kleene also introduced a *complete equivalence* where $(F \leftrightarrow_C G) \equiv \top$ if and only if both F and G have the same logical value, else $(F \leftrightarrow_C G) \equiv \bot$.

We use $I_{\text{Ł}}$, I_K and I_F to denote that an interpretation I uses the Łukasiewicz [Łuk20], Kleene [Kle52] or Fitting [Fit85] semantics, respectively.

The Deduction Theorem does not hold under Łukasiewicz and Kleene semantics. Before showing the counterexample, let us introduce the definitions that are used in the Deduction Theorem.

Definition 6.1 (Model Relation)**.** Let I be an interpretation and ψ be an arbitrary formula. We say I is a model of ψ, denoted by $I \models \psi$, if and only if $I(\psi) = \top$. We write $\models \psi$ if and only if $I \models \psi$ for all interpretations I.

Definition 6.2 (Semantic Consequence)**.** Let Φ be a set of formulae and ψ be a formula. We say ψ is a *semantic consequence* of Φ, denoted by $\Phi \models \psi$, if every model of Φ is also a model of ψ.

	$\neg$
$\top$	$\bot$
$\bot$	$\top$
u	u

		$\wedge$	$\vee$	$\leftarrow_{\mathrm{K}}$	$\leftarrow_{\mathrm{Ł}}$	$\leftrightarrow_{\mathrm{K}}$	$\leftrightarrow_{\mathrm{Ł}}$	$\leftrightarrow_{\mathrm{C}}$
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\top$	$\top$	$\top$	$\bot$	$\bot$	$\bot$
$\top$	u	u	$\top$	$\top$	$\top$	u	u	$\bot$
$\bot$	$\top$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\top$	$\top$	$\top$	$\top$
$\bot$	u	$\bot$	u	u	u	u	u	$\bot$
u	$\top$	u	$\top$	u	u	u	u	$\bot$
u	$\bot$	$\bot$	u	$\top$	$\top$	u	u	$\bot$
u	u	u	u	u	$\top$	u	$\top$	$\top$

Table 6.1: A truth table for three-valued logics. The indices K and Ł refer to Kleene and Łukasiewicz logic, respectively, and $\leftrightarrow_C$ denotes the complete equivalence used by Fitting.

Definition 6.3 (Deduction Theorem)**.** A logic satisfies the *Deduction Theorem* if for any finite set of formulae $\Phi = \{\, \phi_1, \phi_2, \ldots, \phi_n \,\}$ and any formula ψ the following holds:

$$\Phi \models \psi \text{ if and only if } \models (\phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n) \rightarrow \psi \ .$$

Proposition 6.4. The deduction theorem is not satisfied under the Łukasiewicz and Kleene strong three-valued logics.

Proof. Suppose $\Phi = \{\, a, a \rightarrow b \,\}$ and $\psi = b$. Then under Łukasiewicz and Kleene strong three-valued logics, ψ is a semantic consequence of Φ, i.e. $\Phi \models \psi$. However, it does not hold under any of these logics that

$$\models (a \wedge (a \rightarrow b)) \rightarrow b$$

because for the interpretation I defined as $I(a) = u, I(b) = \bot$ we have

$$I_{\mathrm{K}}((a \wedge (a \rightarrow b)) \rightarrow b) = I_{\mathrm{Ł}}((a \wedge (a \rightarrow b)) \rightarrow b) = u \ . \qquad \square$$

The semantics of the connectives are summarized in Table 6.1. In Łukasiewicz logic [Łuk20], the set of connectives is $\{\, \neg,\ \wedge,\ \vee,\ \leftarrow_{\mathrm{Ł}},\ \leftrightarrow_{\mathrm{Ł}} \,\}$, in Kleene three-valued logic [Kle52] the set of connectives is $\{\, \neg,\ \wedge,\ \vee,\ \leftarrow_K,\ \leftrightarrow_K \,\}$. Fitting in [Fit85] proposed Kleene strong three-valued logic for logic programming and complete equivalence for the program completion, so in Fitting logic [Fit85] the set of connectives is $\{\, \neg,\ \wedge,\ \vee,\ \leftarrow_K,\ \leftrightarrow_C \,\}$. Later, Stenning and van Lambalgen used Fitting logic to model human reasoning [SvL08].

Table 6.2 gives an overview of validity of some common logical laws with respect to the Łukasiewicz, Kleene and Fitting logics.

Laws	Łukasiewicz	Kleene	Fitting
Idempotency $F \wedge F \equiv F$ $F \vee F \equiv F$	Yes	Yes	Yes
Commutativity $F \wedge G \equiv G \wedge F$ $F \vee G \equiv G \vee F$	Yes	Yes	Yes
Associativity $(F \wedge G) \wedge H \equiv F \wedge (G \wedge H)$ $(F \vee G) \vee H \equiv F \vee (G \vee H)$	Yes	Yes	Yes
Absorption $(F \wedge G) \vee F \equiv F$ $(F \vee G) \wedge F \equiv F$	Yes	Yes	Yes
Distributivity $F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$ $F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$	Yes	Yes	Yes
Double Negation $\neg\neg F \equiv F$	Yes	Yes	Yes
de Morgan $\neg(F \wedge G) \equiv (\neg F \vee \neg G)$ $\neg(F \vee G) \equiv (\neg F \wedge \neg G)$	Yes	Yes	Yes
Equivalence $F \leftrightarrow G \equiv (F \rightarrow G) \wedge (G \rightarrow F)$	Yes	Yes	No
Implication $F \leftarrow G \equiv F \vee \neg G$	No	Yes	Yes
Contraposition $F \leftarrow G \equiv \neg G \leftarrow \neg F$	Yes	Yes	Yes
Syllogism $(F \leftarrow G) \wedge (G \leftarrow H) \equiv F \leftarrow H$	No	Yes	Yes
Excluded Middle $F \vee \neg F \equiv \top$	No	No	No
Contradiction $F \wedge \neg F \equiv \bot$	No	No	No

Table 6.2: Some common logical laws under Łukasiewicz, Kleene and Fitting semantics.

6.2 Fitting Operator

Let us recall the definition of Fitting operator. Let I be an interpretation and $\mathcal{P}$ a program. The *Fitting immediate consequence operator* is defined as follows: $\Phi_{F,\mathcal{P}}(I) = \langle J^\top, J^\bot \rangle$, where

$$J^\top = \{\, A \mid \text{there exists } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ with } I(\mathit{Body}) = \top \,\} \text{ and}$$
$$J^\bot = \{\, A \mid \text{for all } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ we find } I(\mathit{Body}) = \bot \,\}.$$

Please recall that the body of the program is a conjunction of literals and, hence, $I_{\text{Ł}}(\mathit{Body}) = I_K(\mathit{Body}) = I_F(\mathit{Body})$ according to Table 6.1.

Suppose we have two programs $\mathcal{P}_1$ and $\mathcal{P}_2$ and their completions:

$$\begin{aligned} \mathcal{P}_1: &\quad p \leftarrow q. \\ \mathit{comp}(\mathcal{P}_1): &\quad p \leftrightarrow q. \\ &\quad q \leftrightarrow \bot. \\ \mathcal{P}_2: &\quad p \leftarrow q. \\ &\quad q \leftarrow p. \\ \mathit{comp}(\mathcal{P}_2): &\quad p \leftrightarrow q. \\ &\quad q \leftrightarrow p. \end{aligned}$$

Since both programs are ground, from Proposition 3.25 we obtain that the Fitting operators $\Phi_{F,\mathcal{P}_1}$ and $\Phi_{F,\mathcal{P}_2}$ are continuous and we obtain the least fixed points $\mathit{lfp}(\Phi_{F,\mathcal{P}_1}) = \langle \emptyset, \{\, p, q \,\} \rangle$ and $\mathit{lfp}(\Phi_{F,\mathcal{P}_2}) = \langle \emptyset, \emptyset \rangle$. It is easy to verify that the least fixed points are models of the completions under the Fitting semantics, which is no coincidence as was formally proven in [Fit85]. This property holds also under the Łukasiewicz semantics.

Proposition 6.5. Let $\mathcal{P}$ be a program.

1. $I_{\text{Ł}}$ is a fixed point of $\Phi_{F,\mathcal{P}}$ iff $I_{\text{Ł}}$ a model of $\mathit{comp}(\mathcal{P})$.
2. If $I_{\text{Ł}} = \mathit{lfp}(\Phi_{F,\mathcal{P}})$, then $I_{\text{Ł}}$ is the least model of $\mathit{comp}(\mathcal{P})$.

Proof. To show the if-part of 1., suppose $I_{\text{Ł}}(\mathit{comp}(\mathcal{P})) = \top$. In this case we have to show that $I_{\text{Ł}} = \langle I^\top, I^\bot \rangle$ is a fixed point of $\Phi_{F,\mathcal{P}}$, i.e., $\Phi_{F,\mathcal{P}}(I_{\text{Ł}}) = I_{\text{Ł}}$. Let $\Phi_{F,\mathcal{P}}(I_{\text{Ł}}) = J = \langle J^\top, J^\bot \rangle$. Then $J = I_{\text{Ł}}$ if and only if $J^\top = I^\top$ and $J^\bot = I^\bot$. We distinguish four cases:

1. Suppose $A \in I^\top$, i.e., $I_{\text{Ł}}(A) = \top$. Because $I_{\text{Ł}}(\mathit{comp}(\mathcal{P})) = \top$ we find $A \leftrightarrow \mathit{Body}_1 \vee \mathit{Body}_2 \vee \cdots \in \mathit{comp}(\mathcal{P})$ such that $I_{\text{Ł}}(\mathit{Body}_1 \vee \mathit{Body}_2 \vee \cdots) = \top$. Hence, there exists $A \leftarrow \mathit{Body}_i \in \mathit{ground}(\mathcal{P})$, $i \geq 1$, such that $I_{\text{Ł}}(\mathit{Body}_i) = \top$. Therefore, $A \in J^\top$.
2. Suppose $A \in J^\top$. By the definition of $\Phi_{F,\mathcal{P}}$, we find $A \leftarrow \mathit{Body}_i \in \mathit{ground}(\mathcal{P})$, $i \geq 1$, such that $I_{\text{Ł}}(\mathit{Body}_i) = \top$. Hence, we find $A \leftrightarrow \mathit{Body}_1 \vee \mathit{Body}_2 \vee \cdots \in \mathit{comp}(\mathcal{P})$ and $I_{\text{Ł}}(\mathit{Body}_1 \vee \mathit{Body}_2 \vee \cdots) = \top$. Because $I_{\text{Ł}}(\mathit{comp}(\mathcal{P})) = \top$, we find $I_{\text{Ł}}(A) = \top$. Hence, $A \in I^\top$.
3. Suppose $A \in I^\bot$, i.e., $I_{\text{Ł}}(A) = \bot$. Because $I_{\text{Ł}}(\mathit{comp}(\mathcal{P})) = \top$ we find $A \leftrightarrow F \in \mathit{comp}(\mathcal{P})$ such that $I_{\text{Ł}}(F) = \bot$. In this case either $F = \bot$ or

$F = \mathit{Body}_1 \vee \mathit{Body}_2 \vee \cdots$ and for all $i \geq 1$ we find $I_{\text{Ł}}(\mathit{Body}_i) = \bot$. By definition of $\Phi_{F,\mathcal{P}}$ we find $A \in J^{\bot}$ in either case.

4. Suppose $A \in J^{\bot}$. By the definition of $\Phi_{F,\mathcal{P}}$ we find for all $A \leftarrow \mathit{Body}_i \in \mathit{ground}(\mathcal{P})$, $i \geq 1$, that $I_{\text{Ł}}(\mathit{Body}_i) = \bot$. Hence, with $F = \bot$ or $F = \mathit{Body}_1 \vee \mathit{Body}_2 \vee \cdots$ we find $I_{\text{Ł}}(F) = \bot$. Because $I_{\text{Ł}}(\mathit{comp}(\mathcal{P})) = \top$ and $A \leftrightarrow F \in \mathit{comp}(\mathcal{P})$, we conclude $I_{\text{Ł}}(A) = \bot$. Consequently, $A \in I^{\bot}$.

To show the only-if-part of 1., suppose $I_{\text{Ł}} = \langle I^{\top}, I^{\bot} \rangle$ is a fixed point of $\Phi_{F,\mathcal{P}}$. In this case we have to show that $I_{\text{Ł}}(\mathit{comp}(\mathcal{P})) = \top$, i.e., for all formulae $A \leftrightarrow F \in \mathit{comp}(\mathcal{P})$ we have to show that $I_{\text{Ł}}(A) = I_{\text{Ł}}(F)$. We distinguish three cases:

1. If $I_{\text{Ł}}(A) = \top$, then $A \in I^{\top}$. By the definition of $\Phi_{F,\mathcal{P}}$, we find $A \leftarrow \mathit{Body}_i \in \mathit{ground}(P)$, $i \geq 1$, such that $I_{\text{Ł}}(\mathit{Body}_i) = \top$. Hence, $F = (\mathit{Body}_1 \vee \mathit{Body}_2 \vee \cdots)$, $I_{\text{Ł}}(F) = \top$, and the claim holds in this case.

2. If $I_{\text{Ł}}(A) = \bot$, then $A \in I^{\bot}$. By the definition of $\Phi_{F,\mathcal{P}}$ we distinguish two cases:

   - If there is no clause $A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P})$. Then, $A \leftrightarrow \bot \in \mathit{comp}(\mathcal{P})$. Hence $F = \bot$, $I_{\text{Ł}}(F) = \bot$, and the claim holds in this subcase.
   - If for all clauses of the form $A \leftarrow \mathit{Body}_i \in \mathit{ground}(\mathcal{P})$, $i \geq 1$, we find $I_{\text{Ł}}(\mathit{Body}_i) = \bot$, then $F = \mathit{Body}_1 \vee \mathit{Body}_2 \vee \cdots$, $I_{\text{Ł}}(F) = \bot$, and the claim holds in this subcase.

3. If $I_{\text{Ł}}(A) = u$, then $A \notin I^{\top} \cup I^{\bot}$. By the definition of $\Phi_{F,\mathcal{P}}$, for all $A \leftarrow \mathit{Body}_i \in \mathit{ground}(\mathcal{P})$, $i \geq 1$, we find $I_{\text{Ł}}(\mathit{Body}_i) \neq \top$ and there exists $A \leftarrow \mathit{Body}_i \in \mathit{ground}(\mathcal{P})$, $i \geq 1$, such that $I_{\text{Ł}}(\mathit{Body}_i) \neq \bot$. Hence, $F = \mathit{Body}_1 \vee \mathit{Body}_2 \vee \cdots$, $I_{\text{Ł}}(F) = u$, and the claim holds in the final case as well.

To prove 2., suppose $I_{\text{Ł}} = \mathit{lfp}(\Phi_{F,\mathcal{P}})$ and $I_{\text{Ł}}$ is not the least model of $\mathit{comp}(\mathcal{P})$. Then we find an interpretation $J_{\text{Ł}}$ such that $J_{\text{Ł}}(\mathit{comp}(\mathcal{P})) = \top$ and $J_{\text{Ł}} \subset I_{\text{Ł}}$. By 1., $J_{\text{Ł}}$ will be a fixed point of $\Phi_{F,\mathcal{P}}$, which contradicts the assumption that $I_{\text{Ł}}$ is the least fixed point of $\Phi_{F,\mathcal{P}}$. □

A fixed point of the Fitting operator under the Fitting semantics is a model of the completion of the program, but it is not necessarily a model of the program itself. Consider again the program $\mathcal{P}_2$:

$$\begin{aligned} \mathcal{P}_2 : \quad & p \leftarrow q. \\ & q \leftarrow p. \end{aligned}$$

We have $\mathit{lfp}(\Phi_{F,\mathcal{P}_2}) = \langle \emptyset, \emptyset \rangle$ but it is not a model for $\mathcal{P}_2$. This is because under Fitting semantics, if p and q are mapped to u, then both implications are mapped to u as well. However, under the Łukasiewicz semantics, if p and q are mapped to u, then both implications are mapped to $\top$. Hence, $\mathit{lfp}(\Phi_{F,\mathcal{P}_2}) = \langle \emptyset, \emptyset \rangle$ is a model for $\mathcal{P}_2$ under the Łukasiewicz semantics.

Proposition 6.6. Let $\mathcal{P}$ be a program. If $I_{\text{Ł}}(\mathit{comp}(\mathcal{P})) = \top$, then $I_{\text{Ł}}(\mathcal{P}) = \top$.

Proof. If $I_{\text{Ł}}(\mathit{comp}(\mathcal{P})) = \top$, then for all $A \leftrightarrow F \in \mathit{comp}(\mathcal{P})$ we find $I_{\text{Ł}}(A \leftrightarrow F) = \top$. By the law of equivalence we conclude $I_{\text{Ł}}((A \leftarrow F) \wedge (F \leftarrow A)) = \top$ and, consequently, $I_{\text{Ł}}(A \leftarrow F) = \top$. If $F = \bot$ then $\mathit{ground}(\mathcal{P})$ does not contain a clause with head A. Otherwise, $F = \mathit{Body}_1 \vee \mathit{Body}_2 \vee \cdots$ and we distinguish three cases:

1. If $I_{\text{Ł}}(A) = \top$, then we find $I_{\text{Ł}}(A \leftarrow \mathit{Body}_i) = \top$ for all $A \leftarrow \mathit{Body}_i \in \mathit{ground}(P)$.
2. If $I_{\text{Ł}}(A) = \bot$, then for all $i \geq 1$ we find $I_{\text{Ł}}(\mathit{Body}_i) = \bot$ and, consequently, $I_{\text{Ł}}(A \leftarrow \mathit{Body}_i) = \top$ for all $A \leftarrow \mathit{Body}_i \in \mathit{ground}(P)$.
3. If $I_{\text{Ł}}(A) = u$ then either $I_{\text{Ł}}(F) = \bot$ or $I_{\text{Ł}}(F) = u$. The former possibility being similar to case 2. we concentrate on the latter. If $I_{\text{Ł}}(F) = u$ then for at least one i we find $I_{\text{Ł}}(\mathit{Body}_i) = u$ and for all $i \geq 1$ either $I_{\text{Ł}}(\mathit{Body}_i) = u$ or $I_{\text{Ł}}(\mathit{Body}_i) = \bot$. In any case, we find $I_{\text{Ł}}(A \leftarrow \mathit{Body}_i) = \top$ for all $A \leftarrow \mathit{Body}_i \in \mathit{ground}(\mathcal{P})$. □

Corollary 6.7. Let $\mathcal{P}$ be a program. If $I_{\text{Ł}}$ is a fixed point of $\Phi_{F,\mathcal{P}}$, then $I_{\text{Ł}}(\mathcal{P}) = \top$.

Proof. The corollary follows immediately from Propositions 6.5 and 6.6. □

Although a fixed point of the Fitting operator is not always a model of the given program under the Fitting semantics, the program itself may have models. Returning to the example

$$\begin{aligned} \mathcal{P}_2 : \quad & p \leftarrow q. \\ & q \leftarrow p. \end{aligned}$$

its minimal models under the Fitting semantics are $\langle \emptyset, \{ p, q \} \rangle$ and $\langle \{ p, q \}, \emptyset \rangle$. Their intersection $\langle \emptyset, \emptyset \rangle$ is, however, not a model of $\mathcal{P}_2$ under the Fitting semantics. In other words, the model intersection property does not hold under the Fitting semantics. On the other hand, under the Łukasiewicz semantics, $\langle \emptyset, \emptyset \rangle$ is a model for $\mathcal{P}_2$ and, as we will show in the following, the model intersection property does in general hold under the Łukasiewicz semantics.

Proposition 6.8. Let $\mathcal{P}$ be a program. If $I_{\text{Ł}} = \langle I^{\top}, I^{\bot} \rangle$ is a model of $\mathcal{P}$, then $I'_{\text{Ł}} = \langle I^{\top}, \emptyset \rangle$ is also a model of $\mathcal{P}$.

Proof. Let $\mathcal{P}$ be a program and suppose $I_{\text{Ł}} = \langle I^{\top}, I^{\bot} \rangle$ is a model of $\mathcal{P}$. Let $A \leftarrow \mathit{Body}$ be a clause in $\mathit{ground}(\mathcal{P})$. In order to show $I'_{\text{Ł}}(A \leftarrow \mathit{Body}) = \top$ we distinguish three cases:

1. If $A \in I^{\top}$, then $I'_{\text{Ł}}(A \leftarrow \mathit{Body}) = \top$.
2. If $A \in I^{\bot}$, then $I_{\text{Ł}}(A) = \bot$ and $I'_{\text{Ł}}(A) = u$. Because $I_{\text{Ł}}(A \leftarrow \mathit{Body}) = \top$ we conclude that $I_{\text{Ł}}(\mathit{Body}) = \bot$. Hence, we find a literal C in Body with $I_{\text{Ł}}(C) = \bot$. For each literal B occurring in Body we find:
   (a) if B is an atom and $B \in I^{\top}$, then $I_{\text{Ł}}(B) = \top$ and $I'_{\text{Ł}}(B) = \top$,
   (b) if B is an atom and $B \in I^{\bot}$, then $I_{\text{Ł}}(B) = \bot$ and $I'_{\text{Ł}}(B) = u$,
   (c) if B is an atom and $B \notin I^{\top} \cup I^{\bot}$, then $I'_{\text{Ł}}(B) = I_{\text{Ł}}(B) = u$,

(d) if B is of the form $\neg B'$ and $B' \in I^{\top}$, then $I_{\text{Ł}}(B) = \bot$ and $I'_{\text{Ł}}(B) = \bot$,

(e) if B is of the form $\neg B'$ and $B' \in I^{\bot}$, then $I_{\text{Ł}}(B) = \top$ and $I'_{\text{Ł}}(B) = u$,

(f) if B is of the from $\neg B'$ and $B' \notin I^{\top} \cup I^{\bot}$, then $I'_{\text{Ł}}(B) = I_{\text{Ł}}(B) = u$,

Because C must belong to either case (b) or (d) and, hence, $I'_{\text{Ł}}(C)$ is either u or $\bot$, we conclude that $I'_{\text{Ł}}(\mathit{Body})$ is either $\bot$ or u as well. Because $I'_{\text{Ł}}(A) = u$ we conclude that $I'_{\text{Ł}}(A \leftarrow \mathit{Body}) = \top$.

3. If $A \notin I^{\top} \cup I^{\bot}$, then $I_{\text{Ł}}(A) = I'_{\text{Ł}}(A) = u$. Because $I_{\text{Ł}}(A \leftarrow \mathit{Body}) = \top$ we distinguish two cases:

   (a) If $I_{\text{Ł}}(\mathit{Body}) = \bot$, then we conclude as in case 2. that $I'_{\text{Ł}}(\mathit{Body})$ is either $\bot$ or u and, consequently, $I'_{\text{Ł}}(A \leftarrow \mathit{Body}) = \top$.

   (b) If $I_{\text{Ł}}(\mathit{Body}) = u$, then Body must contain a literal B with $I_{\text{Ł}}(B) = u$. In this case, $I'_{\text{Ł}}(B) = u$ as well and, consequently, $I'_{\text{Ł}}(\mathit{Body})$ is either $\bot$ or u. As in the previous subcase we conclude that $I'_{\text{Ł}}(A \leftarrow \mathit{Body}) = \top$. □

As an example consider the program

$$\mathcal{P}_3 : \quad p \leftarrow q, \neg r.$$

In the remainder of this paragraph all models are considered under the Łukasiewicz semantics. The interpretation $\langle \{ p, q \}, \{ r \} \rangle$ is a model for $\mathcal{P}_3$, and so is $\langle \{ p, q \}, \emptyset \rangle$. Similarly, the interpretation $\langle \{ p, r \}, \{ q \} \rangle$ is a model for $\mathcal{P}_3$, and so is $\langle \{ p, r \}, \emptyset \rangle$. Also, the interpretation $\langle \{ r \}, \{ q \} \rangle$ is a model for $\mathcal{P}_3$, and so is $\langle \{ r \}, \emptyset \rangle$. The least model of $\mathcal{P}_3$ is $\langle \emptyset, \emptyset \rangle$.

Proposition 6.9. Let $I_{\text{Ł}1} = \langle I_1^{\top}, \emptyset \rangle$ and $I_{\text{Ł}2} = \langle I_2^{\top}, \emptyset \rangle$ be two models for a program $\mathcal{P}$. Then $I_{\text{Ł}3} = \langle I_1^{\top} \cap I_2^{\top}, \emptyset \rangle$ is a model for $\mathcal{P}$ as well.

Proof. Suppose $I_{\text{Ł}3} = \langle I_3^{\top}, I_3^{\bot} \rangle = \langle I_1^{\top} \cap I_2^{\top}, \emptyset \rangle$ is not a model for $\mathcal{P}$. Then we find $A \leftarrow \mathit{Body} \in \mathcal{P}$ such that $I_{\text{Ł}3}(A \leftarrow \mathit{Body}) \neq \top$. According to Table 6.1, one of the following cases must hold:

1. $I_{\text{Ł}3}(A) = \bot$ and $I_{\text{Ł}3}(\mathit{Body}) = \top$.
2. $I_{\text{Ł}3}(A) = \bot$ and $I_{\text{Ł}3}(\mathit{Body}) = u$.
3. $I_{\text{Ł}3}(A) = u$ and $I_{\text{Ł}3}(\mathit{Body}) = \top$.

Because $I_3^{\bot} = \emptyset$ we find $I_{\text{Ł}3}(A) \neq \bot$ and, consequently, cases 1. and 2. cannot apply. Therefore, we turn our attention to case 3. If $I_{\text{Ł}3}(A) = u$ then there must exist $j \in \{ 1, 2 \}$ such that $I_{\text{Ł}j}(A) = u$. Because $I_{\text{Ł}j}$ is a model for $\mathcal{P}$ we find $I_{\text{Ł}j}(A \leftarrow \mathit{Body}) = \top$ and, thus, $I_{\text{Ł}j}(\mathit{Body})$ is either u or $\bot$. In this case, $\mathit{Body} \neq \top$. Let $\mathit{Body} = B_1 \wedge \cdots \wedge B_m$ with $m \geq 1$.

Because $I_{\text{Ł}3}(\mathit{Body}) = \top$ and $I_3^{\bot} = \emptyset$, we find for all $1 \leq i \leq m$ that B_i is an atom with $I_{\text{Ł}3}(B_i) = \top$. Hence, $\{ B_1, \ldots, B_m \} \subseteq I_3^{\top}$ and, consequently, $\{ B_1, \ldots, B_m \} \subseteq I_j^{\top}$, which contradicts the assumption that $I_{\text{Ł}j}(\mathit{Body})$ is either u or $\bot$. □

Proposition 6.9 does not hold for arbitrary models of $\mathcal{P}$. For instance, let

$$\begin{aligned}\mathcal{P}_4: \quad & p \leftarrow q_1, r_1.\\ & p \leftarrow q_2, r_2.\end{aligned}$$

and $I_{\text{L}1} = \langle \emptyset, \{ p, q_1, r_2 \} \rangle$, $I_{\text{L}2} = \langle \emptyset, \{ p, q_2, r_1 \} \rangle$. We can easily show that $I_{\text{L}1}$ and $I_{\text{L}2}$ are models for $\mathcal{P}_4$. Their intersection $\langle \emptyset, \{ p \} \rangle$, however, is not a model for $\mathcal{P}_4$.

Proposition 6.10. Let $\mathcal{M}_{\text{L}}$ be the set of all models of a program $\mathcal{P}$ under the Łukasiewicz semantics. Then, $\bigcap \mathcal{M}_{\text{L}}$ is a model for $\mathcal{P}$ as well.

Proof. The result follows immediately from Propositions 6.8 and 6.9. □

The least model of $\mathcal{P}_4$ under the Łukasiewicz semantics is $\langle \emptyset, \emptyset \rangle$. Suppose we have

$$\begin{aligned}\mathcal{P}_5: \quad & p \leftarrow \top.\\ & q \leftarrow p.\\ & r \leftarrow q, \neg s.\end{aligned}$$

The least model of $\mathcal{P}_5$ under the Łukasiewicz semantics is $\langle \{ p, q \}, \emptyset \rangle$. This last example also exhibits that the least fixed point of the Fitting operator is not necessarily the least model of the underlying program because $\mathit{lfp}(\Phi_{F,\mathcal{P}_4}) = \langle \{ p, q, r \}, \{ s \} \rangle$.

6.3 Stenning and van Lambalgen Operator

Let us recall the definition of Stenning and van Lambalgen operator.

Let I be an interpretation and $\mathcal{P}$ be an extended program. The *Stenning and van Lambalgen immediate consequence operator* is defined as follows: $\Phi_{SvL,\mathcal{P}}(I) = \left\langle J^{\top}, J^{\bot} \right\rangle$, where

$$\begin{aligned}J^{\top} &= \{\, A \mid \text{there exists } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ with } I(\mathit{Body}) = \top \,\} \text{ and}\\ J^{\bot} &= \{\, A \mid \text{there exists } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ and}\\ &\qquad \text{for all } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ we find } I(\mathit{Body}) = \bot \,\}\end{aligned}$$

We proved in Proposition 3.21 that $\Phi_{SvL,\mathcal{P}}$ is monotonic for and so by Proposition 3.11 that the least fixed point of $\Phi_{SvL,\mathcal{P}}$ can be computed by iterating $\Phi_{SvL,\mathcal{P}}$ starting from empty interpretation. More formally, let I be the *least fixed point of* $\Phi_{SvL,\mathcal{P}}$ and let

$$I_0 = \langle \emptyset, \emptyset \rangle \tag{6.1}$$

$$I_\alpha = \Phi_{SvL,\mathcal{P}}(I_{\alpha-1}) \text{ for every non-limit ordinal } \alpha > 0 \tag{6.2}$$

$$I_\alpha = \bigcup_{\beta<\alpha} I_\beta \text{ for every limit ordinal } \alpha \tag{6.3}$$

Then for some ordinal γ it holds that $I = I_\gamma$.

Before discussing further properties of the new operator, we reconsider the program $\mathcal{P}_1$ from the previous section:

$$\begin{aligned} \mathcal{P}_1: &\quad p \leftarrow q. \\ \mathit{comp}(\mathcal{P}_1): &\quad p \leftrightarrow q. \\ &\quad q \leftrightarrow \bot. \end{aligned}$$

$\Phi_{SvL,\mathcal{P}}$ admits a least fixed point for $\mathcal{P}_1$ and we obtain $\mathit{lfp}(\Phi_{SvL,\mathcal{P}_1}) = \langle \emptyset, \emptyset \rangle$. One should note that this result differs from $\mathit{lfp}(\Phi_{F,\mathcal{P}_1}) = \langle \emptyset, \{\, p, q \,\} \rangle$.

Now consider

$$\begin{aligned} \mathcal{P}_1': &\quad p \leftarrow q. \\ &\quad q \leftarrow \bot. \\ \mathit{comp}(\mathcal{P}_1'): &\quad p \leftrightarrow q. \\ &\quad q \leftrightarrow \bot. \end{aligned}$$

Note that $\mathit{comp}(\mathcal{P}_1') = \mathit{comp}(\mathcal{P}_1)$. Further, we find $\mathit{lfp}(\Phi_{SvL,\mathcal{P}_1'}) = \mathit{lfp}(\Phi_{F,\mathcal{P}_1}) = \langle \emptyset, \{p, q\} \rangle$. Thus, by adding negative facts, Stenning and van Lambalgen operator can simulate Fitting operator. But it is more liberal in that if there is no clause with head A in the extended program, then its meaning remains undefined.

Obviously, completion as defined in Section 2.13 is unsuitable for extended programs $\mathcal{P}$. Let us define a weak completion of $\mathcal{P}$ by omitting the second step of program completion.

Definition 6.11 (Weak Program Completion)**.** Let $\mathcal{P}$ be a logic program. Consider the following transformation:

1. Replace all clauses in $\mathit{ground}(\mathcal{P})$ with the same head (ground atom) $A \leftarrow \mathit{Body}_1,\ A \leftarrow \mathit{Body}_2, \ldots$ by the single expression $A \leftarrow \mathit{Body}_1 \vee \mathit{Body}_2 \vee \cdots$.
2. Replace all occurrences of $\leftarrow$ by $\leftrightarrow$.

The resulting set of formulae is called *weak completion of* $\mathcal{P}$ and is denoted by $\mathit{wcomp}(\mathcal{P})$. One should observe that in step 1 there may be infinitely many clauses with the same head resulting in a countable disjunction. However, its semantic behavior is unproblematic.

Returning to the previous examples, we find

$$\begin{aligned} \mathit{wcomp}(\mathcal{P}_1): &\quad p \leftrightarrow q. \\ \mathit{wcomp}(\mathcal{P}_1'): &\quad p \leftrightarrow q. \\ &\quad q \leftrightarrow \bot. \end{aligned}$$

In the following we relate the Stenning and van Lambalgen operator and weak completion under the Łukasiewicz semantics.

Lemma 6.12. Let $I_{\mathrm{Ł}}$ be the least fixed point of $\Phi_{SvL,\mathcal{P}}$ and $J_{\mathrm{Ł}}$ be a model of $\mathit{wcomp}(\mathcal{P})$. Then for every atom A the following propositions hold:

$$\text{If } I_{\mathrm{Ł}}(A) = \top, \text{ then } J_{\mathrm{Ł}}(A) = \top \tag{6.4}$$

$$\text{If } I_{\mathrm{Ł}}(A) = \bot, \text{ then } J_{\mathrm{Ł}}(A) = \bot \tag{6.5}$$

Proof. Let $I_{\text{Ł}}$ be the least fixed point of $\Phi_{SvL,\mathcal{P}}$. First we will prove by transfinite induction that for every ordinal α and every atom A it holds that:

$$\text{If } I_\alpha(A) = \top, \text{ then } J_{\text{Ł}}(A) = \top$$
$$\text{If } I_\alpha(A) = \bot, \text{ then } J_{\text{Ł}}(A) = \bot$$

Once we show this, the claim will follow directly by Proposition 3.11 because it implies there is an ordinal $\alpha_\mathcal{P}$ such that $I_{\text{Ł}} = I_{\alpha_\mathcal{P}}$.

Back to the proof by induction. We will consider three cases, one base case when the ordinal $\alpha = 0$ and two inductive cases, one for non-limit ordinals and the other for limit ordinals:

1° Let $\alpha = 0$. Then by (6.1) we have $I_\alpha = \langle \emptyset, \emptyset \rangle$ and so there is no atom such that $I_\alpha(A) = \top$ or $I_\alpha(A) = \bot$ and the claim follows trivially.

2° Let $\alpha > 0$ be a non-limit ordinal. By inductive hypothesis we have for every atom B that:

$$\text{If } I_{\alpha-1}(B) = \top, \text{ then } J_{\text{Ł}}(B) = \top \tag{6.6}$$
$$\text{If } I_{\alpha-1}(B) = \bot, \text{ then } J_{\text{Ł}}(B) = \bot \tag{6.7}$$

Moreover, by (6.2) we have $I_\alpha = \Phi_{SvL,\mathcal{P}}(I_{\alpha-1})$. Let us consider the two claims separately:

1. If $I_\alpha(A) = \top$, then according to the definition of $\Phi_{SvL,\mathcal{P}}$ there must be some rule $A \leftarrow \mathit{Body}_i \in \mathit{ground}(\mathcal{P})$ such that $I_{\alpha-1}(\mathit{Body}_i) = \top$. Let

$$\mathit{Body}_i = B_1 \wedge B_2 \wedge \cdots \wedge B_k \wedge \neg B_{k+1} \wedge \neg B_{k+2} \wedge \cdots \wedge \neg B_m$$

Then for each s with $1 \leq s \leq k$ we have $I_{\alpha-1}(B_s) = \top$ and for each t with $k < t \leq m$ we have $I_{\alpha-1}(B_t) = \bot$. Using the inductive hypothesis (6.6) and (6.7) we get that for each s with $1 \leq s \leq k$ we have $J_{\text{Ł}}(B_s) = \top$ and for each t with $k < t \leq m$ we have $J_{\text{Ł}}(B_t) = \bot$. Hence $J_{\text{Ł}}(\mathit{Body}_i) = \top$. Furthermore, in $\mathit{wcomp}(\mathcal{P})$ there will be a formula of the form $A \leftrightarrow F$ where F is a disjunction with Body_i as one of the disjuncts. So we have $J_{\text{Ł}}(F) = \top$ and also $J_{\text{Ł}}(A \leftrightarrow F) = \top$ because $J_{\text{Ł}}$ is a model of $\mathit{wcomp}(\mathcal{P})$. This implies $J_{\text{Ł}}(A) = \top$.

2. If $I_\alpha(A) = \bot$, then according to the definition of $\Phi_{SvL,\mathcal{P}}$ all rules of the form $A \leftarrow \mathit{Body}_i \in \mathit{ground}(\mathcal{P})$ must have $I_{\alpha-1}(\mathit{Body}_i) = \bot$. Pick an arbitrary but fixed j and let

$$\mathit{Body}_j = B_1 \wedge B_2 \wedge \cdots \wedge B_k \wedge \neg B_{k+1} \wedge \neg B_{k+2} \wedge \cdots \wedge \neg B_m$$

Then there are two cases to consider:

   i. There is some s with $1 \leq s \leq k$ such that $I_{\alpha-1}(B_s) = \bot$. Then by (6.7) we get $J_{\text{Ł}}(B_s) = \bot$ and hence $J_{\text{Ł}}(\mathit{Body}_j) = \bot$.
   ii. There is some t with $k < t \leq m$ such that $I_{\alpha-1}(B_t) = \top$. Then by (6.6) we get $J_{\text{Ł}}(B_s) = \top$ and hence $J_{\text{Ł}}(\mathit{Body}_j) = \bot$.

In either case we have $J_{\text{Ł}}(\mathit{Body}_j) = \bot$ and since j was arbitrarily chosen, we can conclude that for every i we have $J_{\text{Ł}}(\mathit{Body}_i) = \bot$.

Furthermore, in $wcomp(\mathcal{P})$ there is a formula of the form $A \leftrightarrow F$ where F is the disjunction $Body_1 \vee Body_2 \vee \cdots$. So $J_{\text{Ł}}(F) = \bot$ and also $J_{\text{Ł}}(A \leftrightarrow F) = \top$ because $J_{\text{Ł}}$ is a model of $wcomp(\mathcal{P})$. This implies $J_{\text{Ł}}(A) = \bot$.

3° Let α be a limit ordinal. By inductive hypothesis we have for every atom B and every ordinal $\beta < \alpha$ that:

$$\text{If } I_\beta(B) = \top, \text{ then } J_{\text{Ł}}(B) = \top \tag{6.8}$$

$$\text{If } I_\beta(B) = \bot, \text{ then } J_{\text{Ł}}(B) = \bot \tag{6.9}$$

Moreover, by (6.3) we have $I_\alpha = \bigcup_{\beta<\alpha} I_\beta$. Let us consider the two claims separately:

1. If $I_\alpha(A) = \top$, then there is some ordinal $\beta < \alpha$ such that $I_\beta(A) = \top$ and by the inductive hypothesis (6.8) we have $J_{\text{Ł}}(A) = \top$.
2. If $I_\alpha(A) = \bot$, then there is some ordinal $\beta < \alpha$ such that $I_\beta(A) = \bot$ and by the inductive hypothesis (6.9) we have $J_{\text{Ł}}(A) = \bot$. □

Proposition 6.13. Let $\mathcal{P}$ be an extended program. If $I_{\text{Ł}}$ is the least fixed point of $\Phi_{SvL,\mathcal{P}}$, then $I_{\text{Ł}}$ is a minimal model of $wcomp(\mathcal{P})$.

Proof. First we will show that $I_{\text{Ł}}$ is a model of $wcomp(\mathcal{P})$. Let us pick an arbitrary formula $(A \leftrightarrow F) \in wcomp(\mathcal{P})$. We want to show that $I_{\text{Ł}}(A \leftrightarrow F) = \top$. We will consider three cases according to the truth value of A in $I_{\text{Ł}}$:

a) If $I_{\text{Ł}}(A) = \top$, then according to the definition of $\Phi_{SvL,\mathcal{P}}$, there exists a rule $A \leftarrow Body_i \in ground(\mathcal{P})$ such that $I_{\text{Ł}}(Body_i) = \top$. Since $Body_i$ is one of the disjuncts of F, this implies $I_{\text{Ł}}(F) = \top$ and hence $I_{\text{Ł}}(A \leftrightarrow F) = \top$.

b) If $I_{\text{Ł}}(A) = \bot$, then according to the definition of $\Phi_{SvL,\mathcal{P}}$, for every rule $A \leftarrow Body_i \in ground(\mathcal{P})$ we have $I_{\text{Ł}}(Body_i) = \bot$. So all disjuncts in F are false in $I_{\text{Ł}}$ and therefore also $I_{\text{Ł}}(F) = \bot$. Hence $I_{\text{Ł}}(A \leftrightarrow F) = \top$ as desired.

c) If $I_{\text{Ł}}(A) = u$, then according to the definition of $\Phi_{SvL,\mathcal{P}}$ there is no rule $A \leftarrow Body_i \in ground(\mathcal{P})$ with $I_{\text{Ł}}(Body_i) = \top$ and there is some rule $A \leftarrow Body_j \in ground(\mathcal{P})$ with $I_{\text{Ł}}(Body_j) \neq \bot$. So none of the disjuncts in F is true, but it is also not the case that all of them are false. Therefore $I_{\text{Ł}}(F) = u$ and $I_{\text{Ł}}(A \leftrightarrow F) = \top$.

To prove that $I_{\text{Ł}}$ is a minimal model of $wcomp(\mathcal{P})$, let $I_{\text{Ł}} = \langle I^\top, I^\bot \rangle$. By Lemma 6.12 we have that for any model $J_{\text{Ł}} = \langle J^\top, J^\bot \rangle$ of $wcomp(\mathcal{P})$ it holds that $I^\top \subseteq J^\top$ and $I^\bot \subseteq J^\bot$. Hence no proper subset of $I_{\text{Ł}}$ can be a model of $wcomp(\mathcal{P})$ and $I_{\text{Ł}}$ is a minimal model of $wcomp(\mathcal{P})$. □

Proposition 6.14. Let $\mathcal{P}$ be an extended program. If $I_{\text{Ł}}$ is a minimal model of $wcomp(\mathcal{P})$, then $I_{\text{Ł}}$ is the least fixed point of $\Phi_{SvL,\mathcal{P}}$.

Proof. Let $I_{\text{Ł}} = \langle I^\top, I^\bot \rangle$ be a minimal model $wcomp(\mathcal{P})$ and let $J_{\text{Ł}} = \langle J^\top, J^\bot \rangle$ be the least fixed point of $\Phi_{SvL,\mathcal{P}}$. By Lemma 6.12 we know that $J^\top \subseteq I^\top$ and $J^\bot \subseteq I^\bot$. Further, by Proposition 6.13 we have that $J_{\text{Ł}}$ is a minimal model of $wcomp(\mathcal{P})$. But then it must be the case that $I_{\text{Ł}} = J_{\text{Ł}}$ because otherwise we have a conflict with the minimality of $I_{\text{Ł}}$. □

Corollary 6.15. For any extended program $\mathcal{P}$, *wcomp*($\mathcal{P}$) has a least model.

Proof. Follows from Propositions 6.13 and 6.14 and the fact that the least fixed point of $\Phi_{SvL,\mathcal{P}}$ is unique. □

Corollary 6.16. Let $\mathcal{P}$ be an extended program. Then $I_{\text{Ł}}$ is the least fixed point of $\Phi_{SvL,\mathcal{P}}$ iff $I_{\text{Ł}}$ is the least model of *wcomp*($\mathcal{P}$).

Proof. Follows from Propositions 6.13 and 6.14 and Corollary 6.15. □

Proposition 6.17. Let $\mathcal{P}$ be an extended program. If $I_{\text{Ł}}$ is a model of *wcomp*($\mathcal{P}$), then $I_{\text{Ł}}$ is a model of $\mathcal{P}$.

Proof. Suppose $I_{\text{Ł}}$ is a model of *wcomp*($\mathcal{P}$). Then

$$I_{\text{Ł}}(\textit{wcomp}(\mathcal{P})) = \top \ .$$

Hence, for all $A \leftrightarrow F \in \textit{wcomp}(\mathcal{P})$ we find $I_{\text{Ł}}(A \leftrightarrow F) = \top$. By the law of equivalence we conclude $I_{\text{Ł}}((A \leftarrow F) \wedge (F \leftarrow A)) = \top$ and, consequently, $I_{\text{Ł}}(A \leftarrow F) = \top$. Let $F = \textit{Body}_1 \vee \textit{Body}_2 \vee \cdots$. We distinguish three cases:

1. If $I_{\text{Ł}}(A) = \top$, then we find $I_{\text{Ł}}(A \leftarrow \textit{Body}_i) = \top$ for all $A \leftarrow \textit{Body}_i \in \textit{ground}(P)$.
2. If $I_{\text{Ł}}(A) = \bot$, then for all $i \geq 1$ we find $I_{\text{Ł}}(\textit{Body}_i) = \bot$ and, consequently, $I_{\text{Ł}}(A \leftarrow \textit{Body}_i) = \top$ for all $A \leftarrow \textit{Body}_i \in \textit{ground}(P)$.
3. If $I_{\text{Ł}}(A) = u$, then either $I_{\text{Ł}}(F) = \bot$ or $I_{\text{Ł}}(F) = u$. The former possibility being similar to case 2. we concentrate on the latter. If $I_{\text{Ł}}(F) = u$ then we find a j with $I_{\text{Ł}}(\textit{Body}_j) = u$ and for all $i \geq 1$ either $I_{\text{Ł}}(\textit{Body}_i) = u$ or $I_{\text{Ł}}(\textit{Body}_i) = \bot$. In any case, we find $I_{\text{Ł}}(A \leftarrow \textit{Body}_i) = \top$ for all $A \leftarrow \textit{Body}_i \in \textit{ground}(\mathcal{P})$. □

From Proposition 6.13 and Proposition 6.17 we can derive Corollary 6.18 for the Stenning and Lambalgen operator.

Corollary 6.18. Let $\mathcal{P}$ be an extended program. If $I_{\text{Ł}}$ is the least fixed point of $\Phi_{SvL,\mathcal{P}}$, then $I_{\text{Ł}}(\mathcal{P}) = \top$.

Proof. The corollary follows immediately from Propositions 6.13 and 6.17. □

Stenning and van Lambalgen say that the least fixed point of $\Phi_{SvL,\mathcal{P}}$ can be shown to be the minimal model of $\Phi_{SvL,\mathcal{P}}$ in [SvL08] Lemma 4(1.). Contrary to Lemma 4(1.) of [SvL08], this corollary does not hold under the Fitting semantics. Reconsider

$$\begin{aligned} \mathcal{P}_1 : \quad & p \leftarrow q. \\ \textit{comp}(\mathcal{P}_1) : \quad & p \leftrightarrow q. \\ & q \leftrightarrow \bot. \end{aligned}$$

Then $\textit{lfp}(\Phi_{SvL,\mathcal{P}_1}) = \langle \emptyset, \emptyset \rangle$ and, thus, both p and q are mapped to u. Under this interpretation $\mathcal{P}_1$ is mapped to u as well. One should also note that the least

fixed point of the Stenning and van Lambalgen operator for a given program $\mathcal{P}$ is not necessarily the least model of $\mathcal{P}$ under the Fitting semantics. Reconsidering

$$\begin{aligned} \mathcal{P}_1' : \quad & p \leftarrow q. \\ & q \leftarrow \bot. \end{aligned}$$

we find $lfp(\Phi_{SvL,\mathcal{P}_1'}) = \langle \emptyset, \{p, q\} \rangle$ whereas the least model of $\mathcal{P}_1'$ under the Łukasiewicz semantics is $\langle \emptyset, \emptyset \rangle$.

Chapter 7

Connectionist System for the Stenning and van Lambalgen Operator

Stenning and van Lambalgen argue that there is a relationship between the study of neural networks and the study of brain functions. Basically the function of brain can be modelled using neural networks. In this chapter, we show how to model an algorithm for mapping the Stenning and van Lambalgen operator onto a recurrent neural network with a feed-forward core. We show that the stable state of the network is the least fixed point of the Stenning and van Lambalgen operator. Later, we show how some human reasoning tasks can be adequately modelled in the proposed logic and its connectionist realization.

7.1 The Core Method

In [HK94] a connectionist model generator for propositional logic programs using recurrent networks with a feed-forward core was presented. It was later called the *core method* [BH06]. The core method has been extended and applied to a variety of programs including modal (see e.g. [dGZ99]) and first-order logic programs [BHHW07]. It is based on the idea that feed-forward connectionist networks can approximate almost all functions arbitrarily well [HSW89, Fun89] and, hence, they can also approximate – and in some cases compute – the immediate consequence operators associated with logic programs. Moreover, if such an operator is a contraction mapping on a complete metric space, then the Banach Contraction Theorem (Theorem 4.4) ensures that a unique fixed point exists such that the sequence constructed from applying the operator iteratively to any element of the metric space converges to the fixed point (Chapter 3). Turning the feed-forward core into a recurrent network allows to compute or approximate the least model of a logic program [HS99].

Kalinke has applied the core method to logic programs under the Fitting semantics presented in Chapter 6 [Kal94]. In particular, her feed-forward cores compute $\Phi_{F,\mathcal{P}}$ for any given program $\mathcal{P}$. Seda and Lane showed that the core method can be extended to many-valued logic programs [SL06]. Restricted to

three-valued logic programs considered here, their cores also compute $\Phi_{F,\mathcal{P}}$. In the sequel, these approaches are modified in order to compute $\Phi_{SvL,\mathcal{P}}$.

Given a program $\mathcal{P}$, the following algorithm translates $\mathcal{P}$ into a feed-forward core. Let m be the number of propositional atoms occurring in $\mathcal{P}$. Without loss of generality, we may assume that the atoms are denoted by natural numbers from $[1, m]$. Let $\omega \in \mathbb{R}^+$.

1. The input and output layer is a vector of binary threshold units of length $2m$ representing interpretations. The $2i-1$-st unit in the layers, denoted by $i^\top$, is active iff the i-th atom is mapped to $\top$. The $2i$-th unit in the layers, denoted by $i^\bot$, is active iff the i-th atom is mapped to $\bot$. Both, the $2i-1$-st and the $2i$-th unit, are passive iff the i-th atom is mapped to u. The case where the $2i-1$-st and the $2i$-th unit are active is not allowed.

   The threshold of each unit occurring in the input layer is set to $\frac{1}{2}$. The threshold of each $2i-1$-st unit occurring in the output layer is set to $\frac{\omega}{2}$. The threshold of each $2i$-th unit occurring in the output layer is set to $max\{\frac{\omega}{2}, l\omega - \frac{\omega}{2}\}$, where l is the number of clauses with head i in $\mathcal{P}$.

   In addition, two units representing $\top$ and $\bot$ are added to the input layer. The threshold of these units is set to $-\frac{1}{2}$.

2. For each clause of the form $A \leftarrow B_1, \ldots, B_k$ occurring in $\mathcal{P}$, do the following.

   (a) Add two binary threshold units $h^\top$ and $h^\bot$ to the hidden layer.

   (b) Connect $h^\top$ to the unit $A^\top$ in the output layer. Connect $h^\bot$ to the unit $A^\bot$ in the output layer.

   (c) For each B_j, $1 \leq j \leq k$, do the following.

      i. If B_j is an atom, then connect the units $B_j^\top$ and $B_j^\bot$ in the input layer to $h^\top$ and $h^\bot$, respectively.

      ii. If B_j is the literal $\neg B$, then connect the units $B^\bot$ and $B^\top$ in the input layer to $h^\top$ and $h^\bot$, respectively.

      iii. If B_j is $\top$, then connect the unit $\top$ in the input layer to $h^\top$.

      iv. If B_j is $\bot$, then connect the unit $\bot$ in the input layer to $h^\bot$.

   (d) Set the threshold of $h^\top$ to $k\omega - \frac{\omega}{2}$, and the threshold of $h^\bot$ to $\frac{\omega}{2}$.

3. Set the weights associated with all connections to ω.

Proposition 7.1. For each program $\mathcal{P}$, there exists a core of binary threshold units computing $\Phi_{SvL,\mathcal{P}}$.

Proof. Assume that the input layer is activated at time t such that it represents an interpretation I. Then, at time $t+1$ an $h^\top$-unit representing $A \leftarrow B_1, \ldots, B_k$ in the hidden layer becomes active iff all units representing $B_1, \ldots, B_k$ in the input layer are active, i.e., if $I(B_1) = \ldots = I(B_k) = \top$. Likewise, at time $t+1$ an $h^\bot$-unit representing $A \leftarrow B_1, \ldots, B_k$ in the hidden layer becomes active iff one unit representing the negation of $B_1, \ldots, B_k$ in the input layer is active, i.e., if $I(\neg B_1) \vee \cdots \vee I(\neg B_k) = \top$. At time $t+2$ a unit representing A in the output later becomes active iff there is an active $h^\top$-unit representing $A \leftarrow B_1, \ldots, B_k$

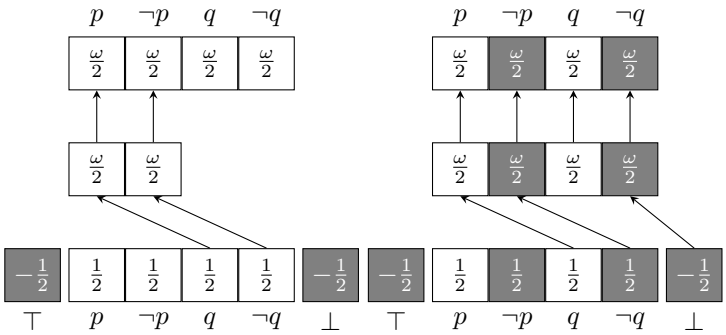

Figure 7.1: The stable states of the feed-forward cores for $\mathcal{P}_1$ (left) and $\mathcal{P}_2$ (right), where all connections have weight ω, active units are shown in grey and passive units in white. The recurrent connections between corresponding units in the output and input layer are not shown.

at time $t+1$. Likewise, at time $t+2$ a unit representing $\neg A$ in the output layer becomes active iff all $h^{\perp}$ units representing rules with head A are active at time $t+1$. Thus, the core is a direct encoding of $\Phi_{SvL,\mathcal{P}}$. □

Given a program $\mathcal{P}$ and its core, a recurrent network can be constructed by connecting each unit in the output layer to its corresponding unit in the input layer with weight 1. In Figure 7.1 the construction is illustrated for the program $\mathcal{P}_1$ and $\mathcal{P}_2$, where

$$\begin{aligned} \mathcal{P}_1: &\quad p \leftarrow q. \\ \mathcal{P}_2: &\quad p \leftarrow q. \\ &\quad q \leftarrow \bot. \end{aligned}$$

Proposition 7.2. For each program $\mathcal{P}$, the corresponding recurrent network initialized by the empty interpretation will converge to a stable state which corresponds to the least fixed point of $\Phi_{SvL,\mathcal{P}}$.

Proof. The result follows immediately from the construction of the recurrent network using Corrolary 6.16 and Proposition 7.1. □

7.2 Human Reasoning

In this section we will discuss some examples taken from [Byr89]. These examples were used by Byrne to show that classical logic cannot appropriately model human reasoning. Stenning and van Lambalgen argue that a three-valued logic programs under a completion semantics can model human reasoning well [SvL08]. Moreover, as we will see, the core method presented in Section 7.1 serves as a connectionist model generator in these cases.

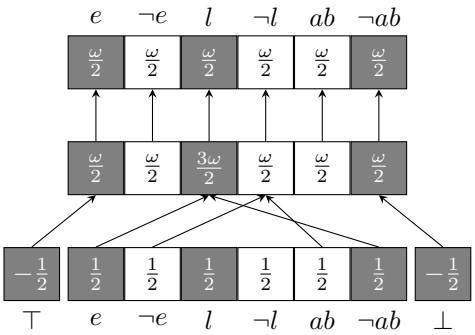

Figure 7.2: The stable state of the feed-forward core for $\mathcal{P}_{marian1}$.

7.2.1 Human Reasoning – Modus Ponens

Consider the following sentences:

If Marian has an essay to write, she will study late in the library. (M1)
She has an essay to write. (M2a)

In [Byr89] 96% of all subjects conclude that *Marian will study late in the library.* The two sentences can be represented by the program

$$\begin{aligned} \mathcal{P}_{marian1}: \quad & l \leftarrow e, \neg ab. \\ & e \leftarrow \top. \\ & ab \leftarrow \bot. \end{aligned}$$

The first sentence is interpreted as a licence for a conditional and the atom ab is used to cover all additional preconditions that we may be unaware of. As we know of no such preconditions, the rule $ab \leftarrow \bot$ is added. The corresponding network as well as its stable state are shown in Figure 7.2. From $lfp(\Phi_{SvL,\mathcal{P}_{marian1}}) = \langle\{l, e\}, \{ab\}\rangle$ follows that Marian will study late in the library.

7.2.2 Human Reasoning – Denial of Antecedent (DA)

Suppose now that the antecedent is denied:

If Marian has an essay to write, she will study late in the library. (M1)
She does not have an essay to write. (M2b)

In [Byr89] 46% of subjects conclude that Marian will not study late in the library. These subject err with respect to classical logic. But they do not err with respect to the non-classical logic considered here. The two sentences can

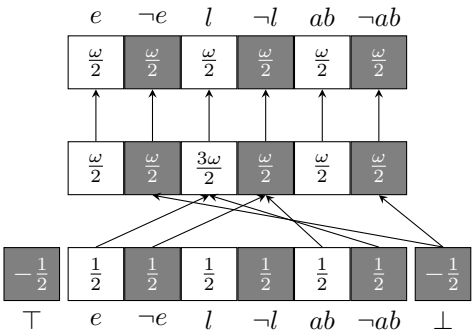

Figure 7.3: The stable state of feed-forward core for $\mathcal{P}_{marian2}$.

be represented by the program

$$\begin{aligned} \mathcal{P}_{marian2}: \quad l &\leftarrow e, \neg ab. \\ e &\leftarrow \bot. \\ ab &\leftarrow \bot. \end{aligned}$$

The corresponding network as well as its stable state are shown in Figure 7.3. From $lfp(\Phi_{SvL,\mathcal{P}_{marian2}}) = \langle \emptyset, \{ab, e, l\}\rangle$ follows that Marian will not study late in the library.

7.2.3 Human Reasoning – Alternative Argument

Suppose now that we have an alternative argument:

If Marian has an essay to write, she will study late in the library. (M1)
She has an essay to write. (M2a)
If she has some textbooks to read, she will study late in the library. (M3)

In [Byr89] 96% of subjects conclude that Marian will study late in the library. The sentences can be represented by the program

$$\begin{aligned} \mathcal{P}_{marian3}: \quad l &\leftarrow e, \neg ab_1. \\ e &\leftarrow \top. \\ ab_1 &\leftarrow \bot. \\ l &\leftarrow t, \neg ab_2. \\ ab_2 &\leftarrow \bot. \end{aligned}$$

The corresponding network as well as its stable state are shown in Figure 7.4. From $lfp(\Phi_{SvL,\mathcal{P}_{marian3}}) = \langle \{e, l\}, \{ab_1, ab_2\}\rangle$. From $\langle \{e, l\}, \{ab_1, ab_2\}\rangle$ follows that Marian will study late in the library. Thus, in this case, the alternative argument does not give any obstacle to the fact that Marian will not study in the library.

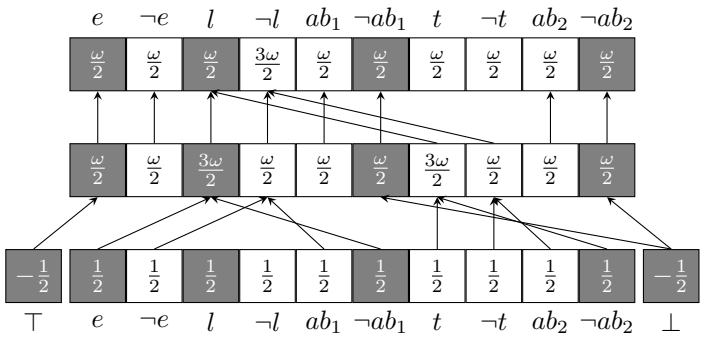

Figure 7.4: The stable state of feed-forward core for $\mathcal{P}_{marian3}$.

7.2.4 Human Reasoning – Alternative Argument and DA

Now consider an alternative argument and the antecedent is denied:

If Marian has an essay to write, she will study late in the library. (M1)
She does not have an essay to write. (M2b)
If she has textbooks to read, she will study late in the library. (M3)

In [Byr89] 4% of subjects conclude that Marian will not study late in the library. These sentences can be represented by

$$\begin{aligned} \mathcal{P}_{marian4}: \quad l &\leftarrow e, \neg ab_1. \\ e &\leftarrow \bot. \\ ab_1 &\leftarrow \bot. \\ l &\leftarrow t, \neg ab_2. \\ ab_2 &\leftarrow \bot. \end{aligned}$$

The corresponding network as well as its stable state are shown in Figure 7.5 From $lfp(\Phi_{SvL,\mathcal{P}_{marian4}}) = \langle \emptyset, \{ab_1, ab_2, e\}\rangle$ follows that it is unknown whether Marian will study late in the library. One should observe that $lfp(\Phi_{F,\mathcal{P}_{marian4}}) = \langle \emptyset, \{ab_1, ab_2, e, t, l\}\rangle$ and, consequently, one would conclude that Marian will not study late in the library. Thus, the Fitting operator leads to a wrong answer with respect to human reasoning, whereas the Stenning and van Lambalgen operator does not.

7.2.5 Human Reasoning – Additional Argument

Consider the presence of an additional argument:

If Marian has an essay to write, she will study late in the library. (M1)
She has an essay to write. (M2a)
If the library stays open, she will study late in the library. (M4)

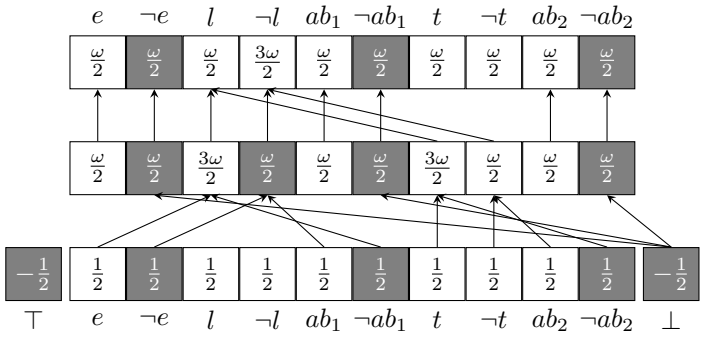

Figure 7.5: The stable state of feed-forward core for $\mathcal{P}_{marian4}$.

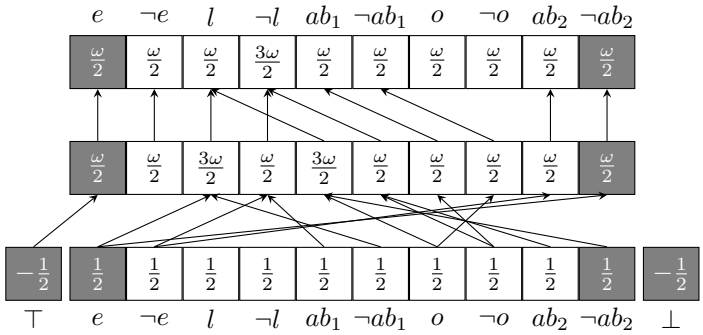

Figure 7.6: The stable state of feed-forward core for $\mathcal{P}_{marian5}$.

In [Byr89] 38% of subjects conclude that Marian will study late in the library. These sentences can be represented by

$$\begin{aligned} \mathcal{P}_{marian5}: \quad & l \leftarrow e, \neg ab_1. \\ & e \leftarrow \top. \\ & l \leftarrow o, \neg ab_2. \\ & ab_1 \leftarrow \neg o. \\ & ab_2 \leftarrow \neg e. \end{aligned}$$

As argued in [SvL08] the third sentence gives rise to an additional argument for studying in the library, viz. that the library is open. Likewise, there must be a reason for going to the library like, for example, writing an essay. The corresponding network as well as its stable state are shown in Figure 7.6. From $lfp(\Phi_{SvL,\mathcal{P}_{marian5}}) = \langle \{e\}, \{ab_2\} \rangle$ it follows that it is unknown whether Marian will study late in the library.

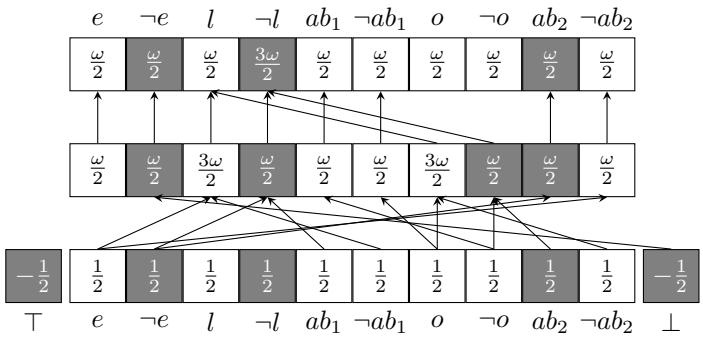

Figure 7.7: The stable state of feed-forward core for $\mathcal{P}_{marian6}$.

7.2.6 Human Reasoning – Additional Argument and DA

As final example suppose the antecedent is denied and there is the present of additional argument:

If Marian has an essay to write, she will study late in the library. (M1)
She does not have an essay to write. (M2b)
If the library is open, she will study late in the library. (M4)

In [Byr89] 63% of subjects conclude that Marian will not study late in the library. These sentences can be represented by

$$\begin{aligned} \mathcal{P}_{marian6}: \quad l &\leftarrow e, \neg ab_1. \\ e &\leftarrow \bot. \\ l &\leftarrow o, \neg ab_2. \\ ab_1 &\leftarrow \neg o. \\ ab_2 &\leftarrow \neg e. \end{aligned}$$

The corresponding network as well as its stable state are shown in Figure 7.7. From $lfp(\Phi_{SvL,\mathcal{P}_{marian6}}) = \langle \{ab_2\}, \{e, l\}\rangle$ follows that Marian will not study late in the library.

Chapter 8

Negative Facts

There is a difficulty to express negative information as facts in logic programs. In [SvL08] Stenning and van Lambalgen denote a negative fact as a clause of the form $A \leftarrow \bot$. However, their approach seems not well defined because the semantics of their syntax does not reflect the intended meaning of a negative fact. In this chapter we first discuss the problems with negative facts proposed by Stenning and van Lambalgen. Next, we give some different approaches from different points of view and we also discuss what problems would appear.

8.1 Stenning and van Lambalgen Negative Facts

Stenning and van Lambalgen in [SvL08] define negative information or a *negative fact* as a clause in the form

$$A \leftarrow \bot. \tag{8.1}$$

The name negative fact is considered only with respect to the (weak) completion of a program as, otherwise, a negative fact like $A \leftarrow \bot$ is also mapped to true by interpretations which map A to u or $\top$.

The first problem with this notation is that when a program contains a clause with the head A, then negative facts can be eliminated without changing the semantics of the program. This is demonstrated in the following program:

$$\begin{aligned} \mathcal{P}_1 : \quad & a \leftarrow \top. \\ & a \leftarrow \bot. \end{aligned}$$

The result of $lfp(\Phi_{SvL,\mathcal{P}_1})$ is $\langle \{ a \}, \emptyset \rangle$. From the previous example we get that a is true because there is a positive fact about a. Thus, in this program, the negative fact about a does not influence the least fixed point.

The second problem is that truth has higher priority than falsity. We can see in $\mathcal{P}_1$ that a is true, even though there is a negative fact about a. Moreover, the undefined atoms also have higher priority than false atoms. Here is an example

of this behaviour:

$$\begin{aligned} \mathcal{P}_2: \quad & a \leftarrow c. \\ & a \leftarrow \bot. \end{aligned}$$

We find that $lfp(\Phi_{SvL,\mathcal{P}_2}) = \langle \emptyset, \emptyset \rangle$. In this program, a is undefined because there is a clause $a \leftarrow c$ and c is undefined. Again, the negative fact about a is ignored in this case because of the presence of a clause with an undefined body.

The third problem is that the Stenning and van Lambalgen operator cannot detect inconsistency in programs. From the previous example, $\mathcal{P}_1$ is inconsistent because a is true from a positive fact but a is also false from the negative fact. Another example follows:

$$\begin{aligned} \mathcal{P}_3: \quad & a \leftarrow b. \\ & b \leftarrow \top. \\ & a \leftarrow \bot. \end{aligned}$$

It can be easily verified that $lfp(\Phi_{SvL,\mathcal{P}_3}) = \langle \{ a, b \}, \emptyset \rangle$. However, according to the information expressed in the program, a should be both true and false. It is true because there is a clause $a \leftarrow b$ and b is true. On the other hand, a should also be false because of the negative fact about a. Thus, this program is inconsistent.

To summarize, the notion of negative facts proposed by Stenning and van Lambalgen does not require any change in the definition of the operator, but it is not accommodated well by the semantics of program clauses. As a consequence, programs with negative information about A allow for models in which A is true or undefined. Moreover, for programs with multiple clauses with A in the head, the negative information about A is simply ignored when at least one of the clauses has a true or undefined body. This amounts to preferring truth over falsity and makes it impossible to detect inconsistencies in a program.

It is true that a human may gather some support for a fact as well as for its negation, but the final decision about the truth of the atom is context-dependent. Hence, we believe that the proposed notion of negative facts is not satisfactory. In the following sections we will see some alternative definitions for negative facts.

8.2 Negative Facts as Constraints

We can use constraints to simulate negative facts. We write a negative fact for atom A as

$$\bot \leftarrow A.$$

From the semantic point of view, this serves its purpose well – the clause $\bot \leftarrow A$ is true if and only if A is false. However, a problem occurs when considering the definition of $\Phi_{SvL,\mathcal{P}}$ because the operator does not accommodate $\bot$ in the head. We can modify the definition to accommodate it:

Definition 8.1. Let I be an interpretation of program $\mathcal{P}$. $\Phi_{\text{SvL-cons},\mathcal{P}}(I) =$

$\langle J^\top, J^\bot \rangle$ where

$$J^\top = \{\, A \mid \text{there exists } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ with } I(\mathit{Body}) = \top \,\} \text{ and}$$
$$J^\bot = \{\, A \mid \text{there exists } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ and}$$
$$\text{for all } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ we find } I(\mathit{Body}) = \bot \,\} \cup$$
$$\{\, A \mid \text{there exists a clause } \bot \leftarrow A \in \mathit{ground}(P) \,\}$$

Suppose we have the program $\mathcal{P}_1$:

$$\begin{aligned} \mathcal{P}_1 : \quad & a \leftarrow \top. \\ & \bot \leftarrow a. \end{aligned}$$

We find that $\mathit{lfp}(\Phi_{\text{SvL-cons},\mathcal{P}_1}) = \langle \{\, a \,\}, \{\, a \,\} \rangle = I$ where I is inconsistent because a is in $I^\top$ and also in $I^\bot$. Thus, this definition of negative facts allows us to detect inconsistent programs. To know whether a program $\mathcal{P}$ is inconsistent, we can see whether the least fixed point of $\Phi_{\text{SvL-cons},\mathcal{P}}(I)$ is consistent or not. If the least fixed point is inconsistent, then the program is inconsistent as well. The next example demonstrates a slightly more complicated case:

$$\begin{aligned} \mathcal{P}_3 : \quad & a \leftarrow b. \\ & b \leftarrow \top. \\ & \bot \leftarrow a. \end{aligned}$$

We have $\mathit{lfp}(\Phi_{\text{SvL-cons},\mathcal{P}_3}) = \langle \{\, a, b \,\}, \{\, a \,\} \rangle$ which also corresponds with the intuition: b is true because of the positive fact and a is both true and false and the program can be labeled as inconsistent.

Another problem that we run into is caused by the underlying semantics (be it Łukasiewicz, Kleene or Fitting semantics) that makes the implication $F \leftarrow G$ undefined when F is false and G is undefined (see Table 6.1). Consider the following program:

$$\begin{aligned} \mathcal{P}_2 : \quad & a \leftarrow c. \\ & \bot \leftarrow a. \end{aligned}$$

Then $\mathit{lfp}(\Phi_{\text{SvL-cons},\mathcal{P}_2}) = \langle \emptyset, \{\, a \,\} \rangle = I$. While we believe that this interpretation corresponds with the intuitions underlying the clauses of $\mathcal{P}_2$, it is not a model of $\mathcal{P}_2$ because $I(a) = \bot$, $I(c) = u$ and hence $I(a \leftarrow c) = u$. Based on Stenning and van Lambalgen's original intention, which is that the negative facts should be treated in the same way as positive facts, it turns out that I is indeed the intended meaning for the program $\mathcal{P}_2$. Moreover, the only model of $\mathcal{P}_2$ where a is false is the one where c is also false. This, however, seems to resemble abduction that is studied in abductive logic programs [KD01]. Further discussion and possible resolutions of this issue are left for future work.

Another problem with this definition is that we cannot handle constraints with more than one atom in the body. For example, we cannot conclude anything from the clause $\bot \leftarrow A, B$. Constraints are also usually not used for any kind of inference, so we seem to be abusing them. A proper extension by constraints would be desirable, but it should be independent of the introduction of negative facts.

To summarize, constraints can be used to semantically capture negative facts, in difference to the original proposal by Stenning and van Lambalgen. However, the operator needs to be modified to accommodate this definition and it may also happen that the least fixed point of the operator will no longer be a model of the program itself. Also, it is questionable whether constraints are a suitable tool for this purpose, since the modified operator makes inferences based on the constraints contained in the program.

8.3 Using Program Transformation to Compute Negative Facts

Given a program $\mathcal{P}$. If A is known to be false, i.e. $A \leftarrow \bot$ is in $\mathcal{P}$, then do the following:

1. remove all clauses from $\mathcal{P}$ where A is the head;
2. replace all occurencies of $\neg A$ by $\top$ and all occurencies of A by $\bot$.

We call the resulting program the *reduction of* $\mathcal{P}$ and denote it by $\mathit{reduct}(\mathcal{P})$. The next step is compute $\Phi_{SvL,\mathit{reduct}(\mathcal{P})}$ as usual. Let I be the resulting least fixed point of $\Phi_{SvL,\mathit{reduct}(\mathcal{P})}$, then add A to $I^{\bot}$. The result is the intended semantics of $\mathcal{P}$.

Let us see the result of $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ from the previous examples. The reduction of $\mathcal{P}_1$ is $\mathit{reduct}(\mathcal{P}_1) = \emptyset$ and $\mathit{lfp}(\Phi_{SvL,\mathit{reduct}(\mathcal{P}_1)}) = \langle \emptyset, \emptyset \rangle$. Based on the procedure, we get $I = \langle \emptyset, \{ a \} \rangle$ as the resulting intended semantics of $\mathcal{P}_1$. We can see that the inconsistency of $\mathcal{P}_1$ was not detected, but in this case, a was inferred to be false, as opposed to the original proposal where a is inferred to be true. Hence, it seems that in this proposal falsity is preferred to truth when both conclusions are required by the clauses of the program.

The reduction of $\mathcal{P}_2$ is also an empty program and hence $\mathit{lfp}(\Phi_{SvL,\mathit{reduct}(\mathcal{P}_2)})$ is $\langle \emptyset, \emptyset \rangle$. After adding a to the set of negative atoms, we obtain the interpretation $I = \langle \emptyset, \{ a \} \rangle$. We run into the same problem as with the previous example because I is neither a model of $\mathcal{P}_2$ nor a model of $\mathit{wcomp}(\mathcal{P}_2)$.

Turning to $\mathcal{P}_3$, the reduction of $\mathcal{P}_3$ is

$$\mathit{reduct}(\mathcal{P}_3): \quad b \leftarrow \top.$$

and $\mathit{lfp}(\Phi_{SvL,\mathit{reduct}(\mathcal{P}_3)}) = \langle \{ b \}, \emptyset \rangle$. After adding a to the set of false atoms we obtain the interpretation $I = \langle \{ b \}, \{ a \} \rangle$. In this example, we can observe the effect of preferring falsity over truth again.

Similarly to the original proposal, this notion of negative facts does not require any change in the definition of the operator. Further, it is based on a transformation, so there is no direct syntactic representation of the negative information that would have a corresponding semantics. Also, for programs with multiple clauses with A in the head, the negative information about A is preferred over any other information. This amounts to preferring falsity over truth and makes it impossible to detect inconsistencies in a program. Similarly as in the previous proposal, it may happen that the resulting interpretation is neither a model of the program nor a model of the program completion.

8.4 Negative Facts as Default Negation in the Head

The notion negation as failure in the head was introduced in [IS98] and used extensively in Dynamic Logic Programming [ALP+98]. Such programs allow negation as failure not only in the body of a rule as negative premises, but also in the head as negative conclusions. So the negative fact for A is written as follows:

$$\neg A \leftarrow \top.$$

Similarly as in the proposal using constraints from Section 8.2, this syntax is fully reflected in the semantics because a clause of the form $\neg A \leftarrow \top$ is true if and only if A is false. But in order to accommodate this syntax, we need to update the definition of the Stenning and van Lambalgen operator as follows:

Definition 8.2. Let $\mathcal{P}$ be a program which is allowed to have a negative fact in the head and let I be an interpretation of program $\mathcal{P}$. $\Phi_{\text{SvL-not},P}(I) = \langle J^{\top}, J^{\bot} \rangle$ where

$$\begin{aligned} J^{\top} &= \{\, A \mid \text{there exists } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ with } I(\mathit{Body}) = \top \,\} \text{ and} \\ J^{\bot} &= \{\, A \mid \text{there exists } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ and} \\ &\qquad \text{for all } A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ we find } I(\mathit{Body}) = \bot \,\} \cup \\ &\quad \{\, A \mid \text{there exists a clause } \neg A \leftarrow \mathit{Body} \in \mathit{ground}(\mathcal{P}) \text{ with } I(\mathit{Body}) = \top \,\} \end{aligned}$$

In this syntax, we write the program $\mathcal{P}_1$ as follows.

$$\begin{aligned} \mathcal{P}_1 : \quad a &\leftarrow \top. \\ \neg a &\leftarrow \top. \end{aligned}$$

The least fixed point of this program under the modified Stenning and van Lambalgen operator is $\langle \{\, a \,\}, \{\, a \,\} \rangle$ and it is inconsistent. Hence, consistency of the program can be determined based on the consistency of the least fixed point.

For the program $\mathcal{P}_2$, we arrive at the same problem as in the previous proposals:

$$\begin{aligned} \mathcal{P}_2 : \quad a &\leftarrow c. \\ \neg a &\leftarrow \top. \end{aligned}$$

The least fixed point of this program under the modified Stenning and van Lambalgen operator is $\langle \emptyset, \{\, a \,\} \rangle$ and it is neither a model of $\mathcal{P}_2$ nor a model of $\mathit{wcomp}(\mathcal{P}_2)$.

Finally, program $\mathcal{P}_3$ is represented as follows:

$$\begin{aligned} \mathcal{P}_3 : \quad a &\leftarrow b. \\ b &\leftarrow \top. \\ \neg a &\leftarrow \top. \end{aligned}$$

The least fixed point of $\Phi_{\text{SvL-not},\mathcal{P}_3}$ is $\langle \{\, a, b \,\}, \{\, a \,\} \rangle$ and it reflects the intended meaning of $\mathcal{P}_3$.

To summarize, default negation in heads can be used to semantically capture negative facts. The operator needs to be modified to accommodate this definition and it may also happen that the least fixed point of the operator will no longer be a model of the program itself.

8.5 Negative Facts as Explicit Negation

Pereira and Alferes in [AP92, AP96] suggest that there are two different negations in the logic program: default negation and explicit negation. Besides Pereira and Alferes, several authors have stressed and shown the importance of having a second kind of negation in a logic program for use in knowledge representation and non-monotonic reasoning (see e.g. [GL91, Wag91, KS90]).

The default negation is known as implicit negation, i.e., it is not possible to explicitly state falsity. Propositions are assumed false if there is no reason to believe they are true. This negation is often called negation-as-finite-failure and in this thesis it is denoted by $\neg$. This is what we want in some cases. For instance, in the classical example of a database that explicitly states flight connections, one wants to implicitly assume that the absence of a connection in the database means that no such connection exists.

However, we need to express explicit negative information like what is described by Stenning and van Lambalgen in [SvL08]. The negative information plays an important role in natural discourse and common sense reasoning. The representation of some problems in logic programming would be more natural if logic programs had some way of explicitly representing falsity. Consider for example the statement [GL90]:

> "A school bus may cross railway tracks under
> the condition that there is no approaching train"

It would be wrong to express this statement by the rule:

$$cross \leftarrow \neg train$$

By $\neg A$ we mean a notation for *default negation* of atom A. The problem in default negation is that the rule allows the bus to cross when there is no information about the presence or absence of a train.

Now, suppose the notation neg_A denotes the explicit negation of A. The situation is different if explicit negation is used:

$$cross \leftarrow neg\_train$$

Here, *neg_train* denotes the negative information about the absence of a train. From the previous clause, the bus is only allowed to cross if the driver is sure that there is no approaching train.

In order to extend our programs with the explicit negation, we additionally define a new terminology. We use neg_A to denote explicit negation of an atom A. Hence, the *extended clause* is a clause in the form

$$A \leftarrow B_1, \ldots, B_m. \quad (m \geq 1)$$

where A is an atom or an explicit negation of some atom and each B_i for $1 \leq i \leq m$ is either a literal (atom or default negation of an atom) or explicit negation of an atom or $\top$. For the negative information about A we add the clauses

$$\begin{aligned} neg\_A &\leftarrow \top. \\ A &\leftarrow \neg\ neg\_A. \end{aligned}$$

An example follows:

$$\begin{aligned} \mathcal{P}_4 : \qquad a &\leftarrow b. \\ neg\_b &\leftarrow \top. \end{aligned}$$

After adding the implicit clause to $\mathcal{P}_4$ we obtain

$$\begin{aligned} \mathcal{P}'_4 : \qquad a &\leftarrow b. \\ neg\_b &\leftarrow \top. \\ b &\leftarrow \neg neg\_b. \end{aligned}$$

The least fixed point of $\Phi_{SvL,\mathcal{P}'_4}$ is $\langle \{\, neg\_b \,\}, \{\, a, b \,\} \rangle$ and we can conclude that we know explicitly the negation of b is true, hence we conclude b is false and we infer that a is false as well.

In case of $\mathcal{P}_2$, we obtain the following program:

$$\begin{aligned} \mathcal{P}'_2 : \qquad a &\leftarrow c. \\ neg\_a &\leftarrow \top. \\ a &\leftarrow \neg neg\_a. \end{aligned}$$

Further, $lfp(\Phi_{SvL,\mathcal{P}'_2})$ is $\langle \{\, neg\_a \,\}, \emptyset \rangle$. We can see that using a different kind of negation encoded as new atoms, we were able to circumvent the problem of the fixed point not being a model of the program and its weak completion.

Turning to the discussion of $\mathcal{P}_3$, we obtain the following program after adding the implicit clauses:

$$\begin{aligned} \mathcal{P}'_3 : \qquad a &\leftarrow b. \\ b &\leftarrow \top. \\ neg\_a &\leftarrow \top. \\ a &\leftarrow \neg neg\_a. \end{aligned}$$

The least fixed point of $\Phi_{SvL,\mathcal{P}'_3}$ is $\langle \{\, a, b, neg\_a \,\}, \emptyset \rangle$. We can choose to treat this result as it is and perform some kind of paraconsistent reasoning with it, or conclude from it that $\mathcal{P}_3$ is inconsistent because both a and neg_a are true.

To summarize, the approach to negative facts using explicit negation does not require any change in the definition of the operator and allows for inconsistency detection. Although it looks promising, future investigation will be needed to examine it further.

8.6 Summary

Stenning and van Lambalgen [SvL08] argue that the negative facts should be processed in the same way as the positive facts. If $A \leftarrow \top$ is an expression for the positive fact, then Stenning and van Lambalgen proposed $A \leftarrow \bot$ to be the expression denoting a negative fact. However, several problems arise from their syntax like: (i) regardless of the truth value of A, the clause $A \leftarrow \bot$ is always true, (ii) if a program contains another clause with head A, then negative facts can be eliminated without changing the semantics of the program, (iii) the truth values $\top$ and u are preferred to $\bot$, (iv) the operator ignores any inconsistencies in a program.

We showed four different approaches to handle negative facts. However, we are not fully satisfied with any of them and each of them needs further investigation. One of the major issues is how to detect and handle inconsistencies in a program. Reasoning in the presence of inconsistency is desirable in many contexts and it is fundamental for understanding human cognitive processes. It has been studied in philosophical logic by several authors [dC74, Gra78, BR80]. Their intuitions and results have been brought to logic programming mainly by Blair, Pearce, Subrahmanian, Wagner, Damásio and Pereira [BS89b, Pea92, Wag93, DP95, DP97]. For a survey of paraconsistent semantics for logic programs, see [DP98].

Chapter 9

Conclusion and Future Work

It has been argued recently in [SvL08] that a completion-based approach captures many aspects of common sense reasoning. Unlike most approaches to logically modelling common sense reasoning which rely on introspection to characterize common sense, Stenning and van Lambalgen base their model on the large corpus of cognitive science. The result is already helping logic programming to be re-examined in fields such as medical decision-making.

In order to model human reasoning, Stenning and van Lambalgen define a new consequence operator for logic programs. Thus, the role of investigating human reasoning is as important as finding a model of the logic program by taking the least fixed point of the consequence operator as the final state of reasoning [SvL08]. In addition, their operator is defined similarly to the Fitting operator.

This thesis is devoted to studying the properties of the Fitting and the Stenning and van Lambalgen operators. We first studied the least fixed point of these operators and ways to characterize it.

The Knaster-Tarski Fixed Point Theorem ensures that every monotonic mapping on a complete partial order has a least fixed point. Moreover, this least fixed point can be found by repeatedly applying the mapping starting with the least element of the complete partial order. We have shown that the space of three-valued interpretations $\mathcal{I}$ is a complete partial order and that both the Fitting and the Stenning and van Lambalgen operators are monotonic mappings on $\mathcal{I}$. Thus, they have a least fixed point.

Then we turned to examining a stronger property of continuity. According to the Kleene Fixed Point Theorem, the least fixed point of a continuous mapping can be found by repeatedly applying the mapping starting with the least element of the complete partial order up to ω times. We found that neither of the operators is continuous in general, but for certain subclasses of logic programs, the operators are indeed continuous. In particular, the operators are continuous for the class of propositional programs because the underlying space of interpretations $\mathcal{I}$ is finite. Using a mapping to propositional programs, we then generalized this result to the class of ground programs for which the space of interpretations may be infinite.

Another important property in the semantics of logic programs is the contraction property. By the Banach Contraction Theorem, it is ensured that if a mapping is a contraction, then it has a unique fixed point that can be found by repeatedly applying the mapping starting with the any element of the metric space up to ω times. This is important because humans usually reason with some preliminary knowledge. Fitting in [Fit94] shows that using metric methods, the two-valued consequence operator is a contraction for the class of acceptable programs. He conjectures that the same method can be used to show the contraction property of the Fitting operator as well. In this thesis, we adopted his method and formally showed that the Fitting operator is also a contraction for acceptable programs. However, this is not the case with the Stenning and van Lambalgen operator. Their operator is not in general a contraction for acceptable programs, but we proved it is a contraction for the class of acyclic programs. It is shown in [AB90] and [AB94] that every acyclic program is also acceptable, hence for acyclic programs both the Fitting and the Stenning and van Lambalgen operator are contractions.

In recent years, the Fitting operator for logic programs has not been used much. It has been overtaken in interest by the well-founded semantics [GRS91] and stable model semantics [GL88]. The latter extends the former in a well-understood manner, and provides a two-valued semantics for logic programs. Both capture transitive closure and other recursive rule behavior and, thus, are useful for programming. However, there are trade-offs between the Fitting semantics and well-founded semantics. The ability of well-founded semantics to capture properties like graph reachability means that it cannot be modelled by a finite first-order theory such as completion. Well-founded semantics also has a higher complexity than the Fitting semantics. The relationship between the Fitting semantics and the well-founded semantics is brought forward in [HW02] using level mappings. The result is that the Fitting model is a subset of the well-founded model and we showed that the Stenning and van Lambalgen model is a subset of the Fitting model.

Both the Fitting and the Stenning and van Lambalgen operators use the Fitting semantics [Fit85] for three-valued logic to define a model of a logic program. However, the Fitting semantics does not admit the law of equivalence ($F \leftrightarrow G$ is semantically equivalent to $F \leftarrow G \wedge G \leftarrow F$). Consequently, the least fixed point of the operators is only guaranteed to be a model of the program completion but there is no guarantee that it is a model of the program itself. In order to overcome this, we proposed to replace the Fitting semantics by Łukasiewicz three-valued semantics [Łuk20] under which the law of equivalence holds. We showed that this replacement preserves the behavior of the operators and has additional properties that the Fitting semantics lacks. In particular, the model intersection property holds under the Łukasiewicz semantics and we also proved that the least fixed points of the operators are models of the program completion and also models of the program itself.

In [HK94] a connectionist model generator for propositional logic programs using recurrent networks with feed-forward core was presented. It was later called the core method. We showed that the core method can be adapted to implement the immediate consequence operator of Stenning and van Lambalgen. We presented an algorithm for constructing the networks and proved that the networks settle down in a state encoding the least fixed point of the operator. Although our networks consist of logical threshold units, they can be replaced

by bipolar sigmoidal ones while preserving the relationship to logic programs by applying the method first presented in [dGZdC97] (see also [dGGB02]). The modified networks can then be trained using backpropagation or related techniques. Rule extraction methods can be applied to the trained networks and the neural-symbolic cycle can be closed. On the other hand, to the best of our knowledge, there is no evidence that backpropagation is neurally plausible.

The main contribution of thesis is in the field of logic programs. There are, however, still many open issues to be addressed. In the following, we will mention some of them to lead the future work.

First, there is a difficulty to express negative information as facts in logic programs. In [SvL08] Stenning and van Lambalgen denote a negative fact as a clause of the form $A \leftarrow \bot$. However, their approach seems not well defined because the semantics of their syntax does not reflect the intended meaning of a negative fact. We proposed four different approaches to introduce negative facts, namely using constraints, program transformation, explicit negation and default negation in the head. We discussed the advantages and disadvantages of these proposals but we could not find a single best solution. Hence, these and possibly other approaches still needs to be discussed further.

Next, in [SvL08] integrity constraints and abduction are suggested to handle additional human reasoning tasks. As a part of future work we would like to investigate whether the techniques developed in [dGGRW07] can be applied to model these tasks in a connectionist setting. Another important question is that of how important the Stenning and van Lambalgen operator is from a logic programming perspective.

Last but not least, it remains to be seen which semantics is better suited for logic programming, common sense as well as human reasoning. It seems that the Łukasiewicz semantics has nicer theoretical properties, but we still have to investigated how this semantics relates to questions concerning computability and termination. It also appears that the Łukasiewicz semantics gives more flexibility than the Fitting semantics concerning common sense reasoning problems. As far as human reasoning is concerned we would like to find out how individuals treat implications where the premise as well as the conclusion are undefined as this is the distinctive feature between the Łukasiewicz and the Fitting semantics.

References

[AB90] Krzysztof R. Apt and Marc Bezem. Acyclic programs. In Péter Szeredi David H. D. Warren, editor, *Proceedings of the 7th International Conference on Logic Programming*, pages 617–633, Jerusalem, Israel, June 18-20 1990. MIT Press.

[AB94] Krzysztof R. Apt and Roland Bol. Logic programming and negation: A survey. *Journal of Logic Programming*, 19:9–71, 1994.

[ALP+98] José Júlio Alferes, João Alexandre Leite, Luís Moniz Pereira, Halina Przymusinska, and Teodor C. Przymusinski. Dynamic logic programming. In Lenhard K. Schubert Stuart C. Shapiro Anthony G. Cohn, editor, *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 98–111, Trento, Italy, June 2-5 1998. Morgan Kaufmann.

[AP90] Krzysztof R. Apt and Dino Pedreschi. Studies in pure Prolog: Termination. In *Proceedings of the Symposium on Computational Logic*, pages 150–176. Springer, Brussels, Belgium, November 13-14 1990.

[AP92] José Júlio Alferes and Luís Moniz Pereira. On logic program semantics with two kinds of negation. In Krzysztof R. Apt, editor, *Proceedings of the Joint International Conference and Symposium on Logic Programming (JICSLP'92)*, pages 574–588, Washington, DC, November 1992. MIT Press.

[AP93] Krzysztof R. Apt and Dino Pedreschi. Reasoning about termination of pure Prolog programs. *Information and Computation*, 106(1):109–157, 1993.

[AP96] José Júlio Alferes and Luís Moniz Pereira. *Reasoning with Logic Programming*, volume 1111 of *Lecture Notes in Computer Science*. Springer, 1996.

[BG94] Chitta Baral and Michael Gelfond. Logic programming and knowledge representation. *Journal of Logic Programming*, 19:73–148, 1994.

[BH06] Sebastian Bader and Steffen Hölldobler. The core method: Connectionist model generation. In Stefanos D. Kollias, Andreas Stafylopatis, Wlodzislaw Duch, and Erkki Oja, editors, *Proceedings of*

the 16th International Conference on Artificial Neural Networks (ICANN), volume 4132 of *Lecture Notes in Computer Science*, pages 1–13, Athens, Greece, September 10-14 2006. Springer.

[BHHW07] Sebastian Bader, Pascal Hitzler, Steffen Hölldobler, and Andreas Witzel. A fully connectionist model generator for covered first-order logic programs. In Manuela M. Veloso, editor, *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 666–671, Hyderabad, India, January 6-12 2007.

[BR80] Nicholas Brandon and Robert Rescher. *The Logic of Inconsistency.* Basil Blackwell, 1980.

[BS89a] Aida Batarekh and V. S. Subrahmanian. Topological model set deformations in logic programming. *Fundamenta Informaticae*, 12:357–400, 1989.

[BS89b] Howard A. Blair and V. S. Subrahmanian. Paraconsistent logic programming. *Theoretical Computer Science*, 68(2):135–154, 1989.

[Byr89] Ruth M. J. Byrne. Suppressing valid inferences with conditionals. *Cognition*, 31(1):61–83, 1989.

[Cla78] Keith L. Clark. Negation as failure. In Hervé Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum, New York, 1978.

[dC74] Newton C. A. da Costa. On the theory of inconsistent formal systems. *Notre Dame Journal of Formal Logic*, 15(4):479–510, 1974.

[dGGB02] Artur S. d'Avila Garcez, Dov M. Gabbay, and Krysia B. Broda. *Neural-Symbolic Learning Systems: Foundations and Applications.* Springer, Secaucus, NJ, USA, 2002.

[dGGRW07] Artur S. d'Avila Garcez, Dov M. Gabbay, Oliver Ray, and John Woods. Abductive reasoning in neural-symbolic systems. *Topoi*, 26(1):37–49, March 2007.

[dGZ99] Artur S. d'Avila Garcez and Gerson Zaverucha. The connectionist inductive learning and logic programming system. *Applied Intelligence*, 11(1):59–77, 1999.

[dGZdC97] Artur S. d'Avila Garcez, Gerson Zaverucha, and Luis Alfredo V. de Carvalho. *Logic Programming and Inductive Learning in Artificial Neural Networks*, pages 33–46. Logos-Verlag Berlin, 1997.

[DP95] Carlos Viegas Damásio and Luís Moniz Pereira. A model theory for paraconsistent logic programming. In Carlos A. Pinto-Ferreira and Nuno J. Mamede, editors, *Proceedings of the 7th Portuguese Conference on Artificial Intelligence (EPIA'95)*, volume 990 of *Lecture Notes in Computer Science*, pages 377–386, Funchal, Madeira Island, Portugal, October 3-6 1995. Springer.

[DP97] Carlos Viegas Damásio and Luís Moniz Pereira. A paraconsistent semantics with contradiction support detection. In Jürgen Dix, Ulrich Furbach, and Anil Nerode, editors, *Proceedings of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'97)*, volume 1265 of *Lecture Notes in Computer Science*, pages 224–243, Dagstuhl Castle, Germany, July 28-31 1997. Springer.

[DP98] Carlos Viegas Damásio and Luís Moniz Pereira. A survey of paraconsistent semantics for logic programs. In Dov M. Gabbay and Philippe Smets, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems, Volume 2*, pages 241–320. Kluwer Academic Publishers, Dordrecht, 1998.

[DP02] Brian A. Davey and Hilary A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2nd edition, 2002.

[Fit85] Melvin Fitting. A kripke-kleene semantics for logic programs. *Journal of Logic Programming*, 2(4):295–312, 1985.

[Fit94] Melvin Fitting. Metric methods three examples and a theorem. *Journal of Logic Programming*, 21(3):113–127, 1994.

[Fun89] Ken-ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2(3):183–192, 1989.

[GL88] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth Bowen, editors, *Proceedings of the 5th International Conference and Symposium on Logic Programming*, pages 1070–1080, Cambridge, Massachusetts, August 15-19 1988. The MIT Press.

[GL90] Michael Gelfond and Vladimir Lifschitz. Logic programs with classical negation. In Péter Szeredi David H. D. Warren, editor, *Proceedings of the 7th International Conference on Logic Programming*, pages 579–597, Jerusalem, Israel, June 18-20 1990. MIT Press.

[GL91] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[Gra78] John Grant. Classifications for inconsistent theories. *Notre Dame Journal Formal Logic*, 19(3):435–444, 1978.

[GRS91] Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *Journal ACM*, 38(3):620–650, 1991.

[HK94] Steffen Hölldobler and Yvonne Kalinke. Towards a new massively parallel computational model for logic programming. In *Proceedings of the ECAI94 Workshop on Combining Symbolic and Connectionist Processing (ECCAI)*, pages 68–77, 1994.

[HR09a] Steffen Hölldobler and Carroline Dewi Puspa Kencana Ramli. Logic programs under three-valued Łukasiewicz's semantics. In Patricia Hill and David H. D. Warren, editors, *Proceedings of the 25th International Conference on Logic Programming (ICLP 2009)*, volume 5649 of *Lecture Notes in Computer Science*, pages 464–478. Springer Berlin Heidelberg, 2009.

[HR09b] Steffen Hölldobler and Carroline Dewi Puspa Kencana Ramli. Logics and networks for human reasoning. In Cesare Alippi, editor, *Proceedings of the 19th International Conference on Artificial Neural Networks (ICANN 2009)*, volume 5769 of *Lecture Notes in Computer Science*, pages 85–94. Springer Berlin Heidelberg, 2009.

[HS99] Pascal Hitzler and Anthony Karel Seda. Characterizations of classes of programs by three-valued operators. In Michael Gelfond, Nicola Leone, and Gerald Pfeifer, editors, *Proceedings of the 5th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'99)*, volume 1730 of *Lecture Notes in Computer Science*, pages 357–371, El Paso, Texas, USA, December 2-4 1999. Springer.

[HSW89] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[HW02] Pascal Hitzler and Matthias Wendt. The well-founded semantics is a stratified Fitting semantics. In Matthias Jarke, Jana Koehler, and Gerhard Lakemeyer, editors, *Proceedings of the 25th Annual German Conference on Artificial Intelligence (KI 2002)*, volume 2479 of *Lecture Notes in Computer Science*, pages 205–221, Aachen, Germany, September 16-20 2002. Springer.

[IS98] Katsumi Inoue and Chiaki Sakama. Negation as failure in the head. *Journal Logic Programming*, 35(1):39–78, 1998.

[Kal94] Yvonne Kalinke. Ein massiv paralleles Berechnungsmodell für normale logische Programme. Master's thesis, TU Dresden, Fakultät Informatik, 1994. In German.

[KD01] Antonis Kakas and Marc Denecker. Abduction in logic programming. *Journal of Logic and Computation*, 2:719–770, 2001.

[KK01] Mohamed A. Khamsi and William A. Kirk. *An Introduction to Metric Spaces and Fixed Point Theory*. Wiley-Interscience, 2001.

[Kle52] Stephen C. Kleene. *Introduction to Metamathematics*. North Holland, 1952.

[KS90] Robert A. Kowalski and Fariba Sadri. Logic programs with exceptions. In Péter Szeredi David H. D. Warren, editor, *Proceedings of the 7th International Conference on Logic Programming (ICLP'90)*, pages 598–613, Jerusalem, Israel, June 18-20 1990. MIT Press.

[KS01] William Arthur Kirk and Brailey Sims, editors. *Handbook of Metric Fixed Point Theory*. Springer, 2001.

[Lif85] Vladimir Lifschitz. Closed-world databases and circumscription. *Artificial Intelligence*, 27(2):229–235, 1985.

[Llo84] John Wylie Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 1984.

[Łuk20] Jan Łukasiewicz. O logice trójwartościowej. *Ruch Filozoficzny*, 5:169–171, 1920. English translation: On Three-Valued Logic. In: *Jan Łukasiewicz Selected Works*. (L. Borkowski, ed.), North Holland, 87-88, 1990.

[NBR02] Rui Da Silva Neves, Jean-Francois Bonnefon, and Eric Raufaste. An empirical test of patterns for nonmonotonic inference. *Annals of Mathematics and Artificial Intelligence*, 34:107–130, 2002.

[Pea92] David Pearce. Reasoning with negative information, ii: Hard negation, strong negation and logic programs. In David Pearce and Heinrich Wansing, editors, *Proceedings of the International Workshop on Nonclassical Logics and Information Processing*, volume 619 of *Lecture Notes in Computer Science*, pages 63–79, Berlin, Germany, November 9-10 1992. Springer.

[Rei78] Raymond Reiter. On closed world data bases. In Hervé Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 55–76. Plenum Press, New York, 1978.

[SL06] Anthony Karel Seda and Máire Lane. Some aspects of the integration of connectionist and logic-based systems. *Information*, 9(4):551–562, 2006.

[SvL05] Keith Stenning and Michiel van Lambalgen. Semantic interpretation as computation in nonmonotonic logic: The real meaning of the suppression task. *Cognitive Science*, 29:919–960, 2005.

[SvL08] Keith Stenning and Michiel van Lambalgen. *Human Reasoning and Cognitive Science*. MIT Press, 2008.

[Wag91] Gerd Wagner. A database needs two kinds of negation. In Bernhard Thalheim, János Demetrovics, and Hans-Detlef Gerhardt, editors, *Proceedings of the 3rd Symposium on Mathematical Fundamentals of Database and Knowledge Bases Systems (MFDBS 91)*, volume 495 of *Lecture Notes in Computer Science*, pages 357–371, Rostock, Germany, May 6-9 1991. Springer.

[Wag93] Gerd Wagner. Reasoning with inconsistency in extended deductive databases. In Anil Nerode Luís Moniz Pereira, editor, *Proceedings of the 2nd International Workshop on Logic Programming and Non-monotonic Reasoning (LPNMR'93)*, pages 300–315, Lisbon, Portugal, June 1993. MIT Press.

[Wil04] Stephen Willard. *General Topology*. Dover Publications, 2004.