On the Relation between Weak Completion Semantics and Answer Set Semantics

Emmanuelle-Anna Dietz Saldanha[1] and Jorge Fandinno[2]

[1] Technische Universitat Dresden,
[2] Technische Universitat Potsdam,
Germany

Abstract. The Weak Completion Semantics (WCS) is a computational cognitive theory that has shown to be successful in modeling episodes of human reasoning. As the WCS is a recently developed logic programming approach, this paper investigates the correspondence of the WCS with respect to the well-established Answer Set Semantics (ASP). The underlying three-valued logic of both semantics is different and their models are evaluated with respect to different program transformations. We first illustrate these differences by the formal representation of some examples of a well-known psychological experiment, the suppression task. After that, we will provide a translation from logic programs understood under the WCS into logic programs understood under the ASP. In particular, we will show that logic programs under the WCS can be represented as logic programs under the ASP by means of a *definition completion*, where all *defined* atoms in a program must be false when their definitions are false.

Keywords: Answer Set Programming, Weak Completion Semantics, Strong Negation, Human Reasoning

1 Introduction

The Weak Completion Semantics (WCS), originally presented in [9], has been suggested as a computational cognitive theory, and demonstrated to be adequate for modeling various episodes of human reasoning summarized in [8]. Consider a well-known psychological experiment, the suppression task [2], which showed that participants' answer systematically diverged from classical logic correct answers. Participants were asked to derive conclusions given variations of a set of premises. The first group was given the following two premises:[1]

If she has an essay to finish, then she will study late in the library. $(e \rightsquigarrow \ell)$
She does not have an essay to finish. $(\mathsf{not}\ e)$

[1] The participants received only the natural language sentences, not the abbreviations.

Then, they were asked what necessarily follows assuming that the above premises were true and given three possible answer from where they could choose:

$$\textit{She will study late in the library.} \qquad (\ell)$$
$$\textit{She will not study late in the library.} \qquad (\mathsf{not}\,\ell)$$
$$\textit{She may or may not study late in the library.} \qquad (\ell \textit{ or } \mathsf{not}\,\ell)$$

54% of the participants answered that *She will not study late in the library.* The second group received, additionally to $(e \rightsquigarrow \ell)$ and $(\mathsf{not}\, e)$, the following premise:

$$\textit{If she has a textbook to read, then she will study late in the library.} \qquad (t \rightsquigarrow \ell)$$

Now, only 4% of the participants answered that *She will not study late in the library.* With these results, Byrne showed that humans seem to reason non-monotonically, i.e. they suppressed previously drawn conclusions. The above examples demonstrates that humans do not always apply the *close world assumption* in their inferences. In particular, if they are *made aware* of alternatives, they might rather apply the *open world assumption.* Stenning and van Lambalgen [15] suggested a formal representation of these premises by licenses for inferences. For the first group, the following logic program rules were suggested:

$$\ell \leftarrow e \wedge \mathsf{not}\, ab_1 \qquad e \leftarrow \bot \qquad ab_1 \leftarrow \bot \tag{1}$$

ab_1 is an abnormality predicate. $ab_1 \leftarrow \bot$ and $e \leftarrow \bot$ are (negative) *assumptions* which assume e and ab_1 being false.[2] In the designated model under the WCS [9], which is the least model of the weak completion of that program under the three-valued Łukasiewicz logic [11], e, ℓ and ab_1 are false. On the other hand, the logic program rules for the second group was suggested to be as follows:

$$\ell \leftarrow e \wedge \mathsf{not}\, ab_1 \qquad \ell \leftarrow t \wedge \mathsf{not}\, ab_2 \qquad e \leftarrow \bot \qquad ab_1 \leftarrow \bot \qquad ab_2 \leftarrow \bot \tag{2}$$

Here, in the designated model under the WCS, e, ab_1 and ab_2 are false, and ℓ and t *unknown.* The differences between both models, where in the first case ℓ is false, and in the second case ℓ is unknown, seems to represent well the *suppression effect* occurring in the second group: In the first group, 46% concluded that *She will not study late in the library,* whereas in the second group, only 4% concluded that *She will not study late in the library.* The overall results of all the twelve cases of the suppression tasks seem to be adequately modeled under the WCS [3].

In this paper we will investigate how the above two cases of the suppression task can be modeled under the Answer Set Semantics [7] (ASP), in particular how both semantics correspond to each other. For this purpose, we first introduce the notions and notation used throughout the paper and the underlying three-valued logics. Section 3 introduces ASP and WCS and shows some intermediate results. The main result is presented in Section 4, where the formal correspondence between both semantics is shown, and the above two cases will be discussed again.

[2] The rules will be understood under their *weak completion.*

2 Preliminaries

In this section, we present the general notation and terminology that will be used throughout the paper together with the semantics for *classical logic with strong negation* [16] and *three-valued Łukasiewicz logic* [11]. In the sequel, definitions are specified in the running text, except if we intend to emphasize them.

2.1 Syntax

We assume a fixed non-empty and (possibly infinite) set of ground atoms, denoted by At. The set of (strongly) negated atoms for the atoms in $S \subseteq \mathsf{At}$, is defined as $\neg S \stackrel{\text{def}}{=} \{\neg A \mid A \in S\}$. A *literal* L is either an atom or its (strong) negation, that is $L \in (\mathsf{At} \cup \neg\mathsf{At})$. Given a set of atoms At, a *formula* is defined according to the following grammar:

$$\varphi ::= A \mid \bot \mid \top \mid \mathsf{U} \mid \varphi \circ \psi \mid \neg\varphi \mid \mathsf{not}\,\varphi$$

$\top$, $\bot$ and U denote the truth constants *true*, *false* and *unknown*, respectively. The connective "not" stands for *weak* or *default negation*, whereas "$\neg$" stands for *strong negation*. The connectives "$\leftarrow_{\text{CL}}$", "$\leftarrow_{\text{Ł}}$" and "$\leftarrow$" stand for classical (or material) implication, Łukasiewicz implication and logic programming implication, respectively. The logic programming implication sign $\leftarrow$ is purely syntactic and, different to $\leftarrow_{\text{CL}}$ and $\leftarrow_{\text{Ł}}$, will not be assigned a fixed underlying semantics. We will study two different logic programming semantics and depending on the semantics in consideration, the meaning for $\leftarrow$ is then specified accordingly. We use $\leftrightarrow_{\text{X}}$ as an abbreviation defined by

$$\varphi \leftrightarrow_{\text{X}} \psi \quad \stackrel{\text{def}}{=} \quad (\varphi \leftarrow_{\text{X}} \psi) \wedge (\psi \leftarrow_{\text{X}} \varphi), \tag{3}$$

where $\text{X} \in \{\text{C}, \text{Ł}\}$. A formula φ is *regular* if its only occurrences of implications strong negation $\neg$ only occurs in front of atoms. A formula φ is called *implication-free* if there are no occurrences of the implication connectives $\leftarrow_{\text{CL}}$, $\leftarrow_{\text{Ł}}$ or $\leftarrow$, i.e. its set of connectives is $\{\wedge, \vee, \neg, \mathsf{not}\}$. A formula φ is called *basic* if it is implication-free, and in addiction, it has no occurrences of weak negation, i.e. its only connective are $\{\wedge, \vee, \neg\}$. Basic formulas are implication-free formulas, but not vice versa. Note that, in general, regular formulas that are implication-free need not to be basic nor basic formulas need to be regular.

Example 1. Consider the following three formulas:

$$\varphi_0 = (\neg(\neg p \wedge q) \vee \mathsf{not}\,r) \qquad \varphi_1 = (\neg(\neg p \wedge q)) \qquad \varphi_2 = (\mathsf{not}\,(\neg p \wedge q))$$

φ_0, φ_1 and φ_2 are implication-free formulas, but φ_0 is neither basic nor regular. φ_1 is basic, but not regular, whereas φ_2 is regular but not basic.

F	$\mathsf{not}\, F$
$\top$	$\bot$
U	$\top$
$\bot$	$\top$

$\wedge$	$\top$	U	$\bot$
$\top$	$\top$	U	$\bot$
U	U	U	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$

$\leftarrow_{\text{CL}}$	$\top$	U	$\bot$
$\top$	$\top$	$\top$	$\top$
U	U	$\top$	$\top$
$\bot$	$\bot$	$\top$	$\top$

$\leftrightarrow_{\text{CL}}$	$\top$	U	$\bot$
$\top$	$\top$	U	$\bot$
U	U	$\top$	$\top$
$\bot$	$\bot$	$\top$	$\top$

F	$\neg F$
$\top$	$\bot$
U	U
$\bot$	$\top$

$\vee$	$\top$	U	$\bot$
$\top$	$\top$	$\top$	$\top$
U	$\top$	U	U
$\bot$	$\top$	U	$\bot$

$\leftarrow_{\text{L}}$	$\top$	U	$\bot$
$\top$	$\top$	$\top$	$\top$
U	U	$\top$	$\top$
$\bot$	$\bot$	U	$\top$

$\leftrightarrow_{\text{L}}$	$\top$	U	$\bot$
$\top$	$\top$	U	$\bot$
U	U	$\top$	U
$\bot$	$\bot$	U	$\top$

Table 1. Truth tables for three-valued Łukasiewicz logic $\{\neg, \wedge, \vee, \leftarrow_{\text{L}}, \leftrightarrow_{\text{L}}\}$, and Classical Logic extended with strong negation $\{\neg, \mathsf{not}\,, \wedge, \vee, \leftarrow_{\text{CL}}, \leftrightarrow_{\text{CL}}\}$.

2.2 Three-valued Semantics

A (three-valued) interpretation $I : \mathsf{At} \longrightarrow \{\top, \bot, \mathsf{U}\}$ is a function mapping each atom to a truth constant. We introduce now two different alternative representations of interpretations that are equivalent and common in the literature: An interpretation I can be represented as a pair of set of atoms, $I = \langle I^\top, I^\bot \rangle$ such that $I^\top \cap I^\bot = \emptyset$ where the correspondence is given as follows: $I^\top = \{A \mid I(A) = \top\}$ and $I^\bot = \{A \mid I(A) = \bot\}$. Note that $A \notin (I^\top \cup I^\bot)$ holds iff $I(A) = \mathsf{U}$. Alternatively, I can be represented as a set of literals $I^\top \cup \neg I^\bot$. The first representation is usual in the context of the WCS while the second is usual in the context of Answer Set Programming (ASP) [1]. We will use them interchangeably.

Three-valued interpretations can be ordered either by *knowledge* or by *truth*: Given two interpretations $I = \langle I^\top, I^\bot \rangle$ and $J = \langle J^\top, J^\bot \rangle$, we say that I *contains less knowledge* than J, in symbols $I \subseteq_k J$, iff $I^\top \subseteq J^\top$ and $I^\bot \subseteq J^\bot$ iff $(I^\top \cup \neg I^\bot) \subseteq (J^\top \cup \neg J^\bot)$. In other words, I and J agree in all atoms which are known in I, but I can have more unknown atoms. On the other hand, when the *truth order* is applied, i.e. $\bot \leq \mathsf{U} \leq \top$, then, given two interpretations $I = \langle I^\top, I^\bot \rangle$ and $J = \langle J^\top, J^\bot \rangle$, we say that I *contains less truth* than J, in symbols $I \subseteq_t J$, iff $I^\top \subseteq J^\top$ and $J^\bot \subseteq I^\bot$ iff $(I^\top \cup \neg J^\bot) \subseteq (J^\top \cup \neg I^\bot)$. As we are only interested in the knowledge ordering, we will omit the subscript k in the following, and simply write $I \subseteq J$ when we refer to $I \subseteq_k J$.

In this paper, we will consider two different three-valued logics, so we introduce some general definitions parametrized by the logic. Given a (three-valued) logic $\mathcal{L}$, a three-valued interpretation I *satisfies* a formula φ, in symbols $I \models_{\mathcal{L}} \varphi$, iff I evaluates φ as true, that is $I(\varphi) = \top$. Furthermore, I is called a *(three-valued) model* of a theory Γ (where Γ is a set of formulas) under $\mathcal{L}$, denoted by $I \models_{\mathcal{L}} \Gamma$, iff $I \models_{\mathcal{L}} \varphi$ for all $\varphi \in \Gamma$. I is a $\subseteq$*-minimal model* of Γ iff for no other model J of Γ, $J \subset I$ (ordered according to the knowledge). I is the $\subseteq$*-least model* of Γ iff it is the unique minimal model of Γ. A formula φ is *valid* in $\mathcal{L}$, denoted by $\models_{\mathcal{L}} \varphi$, iff $I \models_{\mathcal{L}} \varphi$ for every interpretation I. Furthermore, we write $\Gamma \models_{\mathcal{L}} \varphi$ iff $I \models_{\mathcal{L}} \Gamma$ implies $I \models_{\mathcal{L}} \varphi$ for every interpretation I. For theories Γ and Γ', we write $\Gamma \equiv_{\mathcal{L}} \Gamma'$ iff every interpretation I satisfies: $I \models_{\mathcal{L}} \Gamma$

iff $I \models_{\mathcal{L}} \Gamma'$. We will omit the brackets { and }, in case $\equiv_{\mathcal{L}}$ is applied to formulas, i.e. we write $\varphi \equiv_{\mathcal{L}} \varphi'$ iff $\{\varphi\} \equiv_{\mathcal{L}} \{\varphi'\}$.

2.3 Classical Logic extended with strong Negation (N-logic)

The distinction between strong and weak negation was first noticed by Nelson [13] in the context of Intuitionistic Logic and later studied by Vakarelov [16] in the context of Classical Logic. The syntax is obtained from the syntax of Classical Logic extended with a the connective "$\neg$" standing for strong negation, that is, formulas are built from the set of connectives $\{\wedge, \vee, \leftarrow_{\text{CL}}, \mathsf{not}\,, \neg\,\}$. Note that here classical negation is dentoed by "not". We call them the **N**-formulas. Accordingly, a **N**-theory is a set of **N**-formulas. Evaluation of its connectives, $\wedge, \vee, \leftarrow_{\text{CL}}, \mathsf{not}$ and $\neg$, is given by the corresponding truth tables in Table 1. In the sequel, we refer to **N**-logic if **N**-formulas or **N**-theories are considered and evaluated with respect to these truth tables. Note that **N**-logic is a conservative extension of classical logic in the sense that, if we restrict ourselves to formulas without strong negation (formulas without the $\neg$ connective), the valid formulas in **N**-logic and Classical logic are the same. We use $\models_{\text{CL}}$ and $\equiv_{\text{CL}}$ to denote entailment and equivalence according to **N**-logic. Weak negation "not" can be defined in terms of "$\leftarrow_{\text{CL}}$" by the following equivalence:

$$\mathsf{not}\,\varphi \quad \equiv_{\text{CL}} \quad \bot \leftarrow_{\text{CL}} \varphi \tag{4}$$

We will consider weak negation here as connective in its own right because of its importance for logic programming.

It is interesting to note that, every (possibly non-regular) **N**-formula can be rewritten as as an equivalent regular **N**-formula applying the following equivalences taken from Vorob'ev calculus (see Section 2.1 in [14] for more details):

$$\neg\neg\varphi \equiv_{\text{CL}} \varphi \tag{5}$$

$$\neg\mathsf{not}\,\varphi \equiv_{\text{CL}} \varphi \tag{6}$$

$$\neg\,(\varphi \wedge \psi) \equiv_{\text{CL}} \neg\varphi \vee \neg\psi \tag{7}$$

$$\neg\,(\varphi \vee \psi) \equiv_{\text{CL}} \neg\varphi \wedge \neg\psi \tag{8}$$

$$\neg\,(\varphi \leftarrow_{\text{CL}} \psi) \equiv_{\text{CL}} \neg\varphi \wedge \psi \tag{9}$$

For any **N**-formula φ, we write $\text{reg}(\varphi)$ for the regular formula obtained from φ by applying the above equivalences. For a theory Γ, by $\text{reg}(\Gamma) \stackrel{\text{def}}{=} \{\, \text{reg}(\varphi) \mid \varphi \in \Gamma \,\}$ we denote the regular theory obtained in the same way.

2.4 Three-valued Łukasiewicz Logic (Ł-logic)

The syntax of the three-valued Łukasiewicz logic introduced in [11] is restricted to the set of connectives $\{\bot, \wedge, \vee, \leftarrow_{\text{L}}, \neg\,\}$, that is, by replacing in Classical Logic connectives "$\leftarrow_{\text{CL}}$" and "not" by "$\leftarrow_{\text{L}}$" and "$\neg$", respectively. We call formulas build from these connectives **Ł**-formulas. Accordingly, a **Ł**-theory is a set of

Ł-formulas. Evaluation of its connectives, $\bot, \wedge, \vee, \leftarrow_{\text{L}}$ and $\neg$, is given by the corresponding truth tables in Table 1. In the sequel, we refer to the Ł-logic, if Ł-formulas or a Ł-theory are considered and evaluated with respect to these truth tables. We use $\models_{\text{L}}$ and $\equiv_{\text{L}}$ to denote entailment and equivalence according to Ł-logic.

3 Logic Programming

A rule r is an expression of the form:

$$\varphi \leftarrow \psi \tag{10}$$

where φ and ψ are implication-free formulas respectively called the *head* the *body* of the rule and L is a literal. $Head(r)$ and $Body(r)$ denote the set of literals that occur in the head and the body of the rule r, respectively. A rule is *basic* (resp. *regular*) iff its body and head are basic (resp. regular). A *normal nested rule* iff is a rule whose head $\varphi \in (\mathsf{At} \cup \neg\mathsf{At})$ is a literal, an *extended rule* is a normal nested rules such that its body is either $\top$, (this rule is called a positive fact), or a conjunction of literals or literals preced by weak negation. A *normal rule* is an extended nested rules such that its head is a positive literal.

A *(logic) program* $\mathcal{P}$ is a set of rules and, for any program $\mathcal{P}$, by $\mathsf{at}(\mathcal{P})$, we denote the set of all atoms occurring in program $\mathcal{P}$. If it is clear from the context, then we assume that $\mathsf{At} = \mathsf{at}(\mathcal{P})$. We also denote by $Head(\mathcal{P}) = \{L \mid L \in Head(r) \text{ and } r \in \mathcal{P}\}$ and $Body(\mathcal{P}) = \{L \mid L \in Body(r) \text{ and } r \in \mathcal{P}\}$ the set of literals occurring in the head and the body of a program $\mathcal{P}$, respectively.

A program is called *basic* (resp. *regular*, *normal*, *extended* or *normal nested*) iff all its rules are basic (resp. regular, normal, extended or normal nested). In case that $\psi = \top$, we will usually write just φ instead of $\varphi \leftarrow \top$. We will use $\varphi \leftrightarrow \psi$ as an abbreviation for a pair of rules $\varphi \leftarrow \psi$ and $\psi \leftarrow \varphi$. For instance, the program $\{a \leftrightarrow b\}$ is an abbreviation for the program $\{a \leftarrow b, b \leftarrow a\}$.

3.1 Answer Set Semantics

We review now and extend the definition of Answer Set Semantics from [10] to possibly non-regular programs. The definition of the Answer Set Semantics to non-regular programs was first introduced by Pearce [14] using an equilibrium condition over the models of the logic of here-and-there with strong negation. The formulation we present here is equivalent, but using classical logic with strong negation and a reduct instead.

An interpretation I is said to be *closed* under some regular program $\mathcal{P}$ iff every $(\varphi \leftarrow \psi) \in \mathcal{P}$ satisfies that $I \models_{\text{CL}} \varphi$ whenever $I \models_{\text{CL}} \psi$ holds. The *reduct* of a regular formula φ with respect to an interpretation I, in symbols φ^I, is recursively defined as follows:

$$\varphi^I \stackrel{\text{def}}{=} \varphi \text{ if } \varphi \text{ is a literal} \tag{11}$$

$$(\varphi \wedge \psi)^I \stackrel{\text{def}}{=} \varphi^I \wedge \psi^I \tag{12}$$

$$(\varphi \vee \psi)^I \stackrel{\text{def}}{=} \varphi^I \vee \psi^I \tag{13}$$

$$(\mathsf{not}\,\varphi)^I \stackrel{\text{def}}{=} \begin{cases} \bot & \text{if } I \models_{\text{CL}} \varphi^I \\ \top & \text{otherwise} \end{cases} \tag{14}$$

The reduct of a rule $(\varphi \leftarrow \psi)^I \stackrel{\text{def}}{=} \varphi^I \leftarrow \psi^I$ is obtained by applying the reduct to its head and body. The reduct of a logic program is obtained by applying the reduct to all its rules, that is $\mathcal{P}^I \stackrel{\text{def}}{=} \{\ r^I \mid r \in \mathcal{P}\ \}$. Furthermore, as done for **N**-theories, we also assign to any program $\mathcal{P}$ an equivalent regular program $\text{reg}(\mathcal{P}) \stackrel{\text{def}}{=} \{\ \text{reg}(r) \mid r \in \mathcal{P}\ \}$ with $\text{reg}(\varphi \leftarrow \psi) \stackrel{\text{def}}{=} \text{reg}(\varphi) \leftarrow \text{reg}(\psi)$ for any rule of the form $r = (\varphi \leftarrow \psi)$. For any **N**-formula φ, we obtain the regular formula $\text{reg}(\varphi)$ by applying the equivalences (5-8) specified in Section 2.3. Then, answer sets are defined in terms of the regular counterpart of any program.

Definition 1. *Given an interpretation I and a program $\mathcal{P}$, I is an* answer set *of $\mathcal{P}$ iff I is an $\subseteq$-minimal closed interpretation under $\text{reg}(\mathcal{P})^I$.*

It is easy to see that, for regular programs, Definition 1 precisely coincide with the definition of answer set from [10].

It is well-known that every answer set of a logic program without strong negation is also a model in classical logic of the propositional theory obtained by replacing the logic programming implication $\leftarrow$ by classical implication $\leftarrow_{\text{CL}}$. We extend this result to the case of logic programs with strong negation by replacing classical logic by its extension with strong negation. Formally, given a logic program $\mathcal{P}$, by $\mathbf{N}(\mathcal{P})$ we denote the **N**-theory resulting of replacing in $\mathcal{P}$ each occurrence of $\leftarrow$ by $\leftarrow_{\text{CL}}$.

Proposition 1. *Given an interpretation I and a program $\mathcal{P}$, I is closed under $\mathcal{P}$ if and only if I is a model of $\mathbf{N}(\mathcal{P})$ under **N**-logic.*

Proof. Note that I is closed under $\mathcal{P}$ iff all (rules) $\varphi \leftarrow \psi \in \mathcal{P}$ satisfy $I \models_{\text{CL}} \varphi$ whenever $I \models_{\text{CL}} \psi$ iff all $\varphi \leftarrow \psi \in \mathcal{P}$ satisfy either $I(\varphi) = \top$ or $I(\psi) \neq \top$ iff all $\varphi \leftarrow \psi \in \mathcal{P}$ satisfy $I(\varphi \leftarrow_{\text{CL}} \psi) = \top$ iff all formulas of the form $\varphi \leftarrow_{\text{CL}} \psi \in \mathbf{N}(\mathcal{P})$ satisfy $I(\varphi \leftarrow_{\text{CL}} \psi) = \top$ iff I is a model of $\mathbf{N}(\mathcal{P})$ under **N**-logic. $\square$

Corollary 1. *Given an interpretation I and a regular program $\mathcal{P}$, I is an answer set of $\mathcal{P}$ if and only if I is a $\subseteq$-minimal model of $\mathbf{N}(\mathcal{P}^I)$ under **N**-logic.*

Proof. From Definition 1 and Proposition 1, it follows that I is an answer set of $\mathcal{P}$ if and only if I is a $\subseteq$-minimal model of $\mathbf{N}(\mathcal{P}^I)$ under **N**-logic. Note that, since $\mathcal{P}$ is regular $\text{reg}(\mathcal{P}) = \mathcal{P}$. Furthermore, we also have $\mathbf{N}(\text{reg}(\mathcal{P}')) = \text{reg}(\mathbf{N}(\mathcal{P}')) \equiv_{\text{CL}} \mathbf{N}(\mathcal{P}')$ for every program $\mathcal{P}'$. Hence, the statement holds. $\square$

Proposition 2. *Given an interpretation I and a regular program $\mathcal{P}$, if I is an answer set of $\mathcal{P}$, then I is a model of $\boldsymbol{N}(\mathcal{P})$ under $\boldsymbol{N}$-logic.*

Proof (sketch). The proof can be carried out by induction in the structure of the formula by noting that φ^I is just the result of partially evaluating subformulas for the form of $\mathsf{not}\,\psi$ with respect to I. Hence, from Proposition 1, it follows I being an answer set of $\mathcal{P}$ implies that I is a ($\subseteq$-minimal) model of $\mathbf{N}(\mathrm{reg}(\mathcal{P})^I)$. Take any formula of the form of $\varphi \leftarrow_{\text{CL}} \psi$ in $\mathcal{P}$. Then, the formula $\mathrm{reg}(\varphi)^I \leftarrow_{\text{CL}} \mathrm{reg}(\psi)^I$ belongs to $\mathbf{N}(\mathrm{reg}(\mathcal{P})^I)$ and, since I is a model of $\mathbf{N}(\mathrm{reg}(\mathcal{P})^I)$, we get that $I(\mathrm{reg}(\varphi)^I \leftarrow_{\text{CL}} \mathrm{reg}(\psi)^I) = \top$. This implies that either $I(\mathrm{reg}(\varphi)^I) = \top$ or $I(\mathrm{reg}(\psi)^I) \neq \top$, which in its turn implies that either $I(\varphi) = \top$ or $I(\psi) \neq \top$ holds. Consequently, $I(\varphi \leftarrow_{\text{CL}} \psi) = \top$ and I is a model of $\mathbf{N}(\mathcal{P})$. □

As may be expected and as the following example shows, the other direction of Proposition 2 does not hold.

Example 2. $\langle \emptyset, \emptyset \rangle = \emptyset$ is the unique answer set of $\mathcal{P} = \{\neg p \leftarrow q\}$ while its corresponding $\mathbf{N}$-theory $\mathbf{N}(\mathcal{P}) = \{\neg p \leftarrow_{\text{CL}} q\}$ has several other models, as for instance $\langle \{p\}, \{q\} \rangle = \{p, \neg q\}$.

It is also well known that normal nested programs without weak nor strong negation (usually called *positive*) have a unique answer set which coincides with the $\subseteq$-minimal classical model (usually called the $\subseteq$-least model) of the corresponding propositional theory (see Proposition 3 in [1]). Similarly, normal nested programs without weak negation (i.e. basic) have at most one $\subseteq$-minimal model (and possibly no model) in classical logic with strong negation. For instance, in the case of $\{a, \neg a\}$ is inconsistent, so it has no $\subseteq$-least model,

Proposition 3. *Given any basic normal nested program $\mathcal{P}$, one of the following two statements holds:*[3]

1. *$\mathcal{P}$ has a unique answer set which is also the $\subseteq$-least model of $\mathbf{N}(\mathcal{P})$, or*
2. *$\mathcal{P}$ has no answer set and $\mathbf{N}(\mathcal{P})$ has no model.*

Proof. First, note that if $\mathbf{N}(\mathcal{P})$ has no model, from Proposition 2, we immediately get that $\mathcal{P}$ has no answer set. Let us show now that if $\mathbf{N}(\mathcal{P})$ has a model, then it has a $\subseteq$-least model. Obviously, since At is finite, if $\mathbf{N}(\mathcal{P})$ has a model, then it has at least some $\subseteq$-minimal model. Suppose, for the sake of contradiction, that $\mathbf{N}(\mathcal{P})$ has two different $\subseteq$-minimal models I_1 and I_2. Then, there are literals L_1 and L_2 such that $I_1(L_1) = I_2(L_2) = \top$ and $I_1(L_2) = I_2(L_1) = \bot$. Let J be an interpretation such that $J(L) = I_1(L)$ if $I_1(L) = I_2(L)$, and $J(L) = \mathsf{U}$, otherwise. Then, we have that $J \subset I_1$ and $J \subset I_2$ so $J \not\models_{\text{CL}} \mathbf{N}(\mathcal{P})$. Hence, there is a formula in $\mathbf{N}(\mathcal{P})$ is of the form $L \leftarrow_{\text{CL}} \varphi$ with L a literal and φ a basic formula such that $J(L) \neq \top$ and $J(\varphi) = \top$. Furthermore, it can be shown by induction in the structure of the formula that, for every basic formula φ and pair of interpretations $J \subseteq I$, we have that $J(\varphi) = \top$ implies $I(\varphi) = \top$. Hence, we

[3] Recall, basic programs only consist of rules, whose head and body are implication-free and have no occurences of weak negation.

have that $I_1(\varphi) = \top$ and $I_2(\varphi) = \top$ which, since I_1 and I_2 are models of $\mathbf{N}(\mathcal{P})$, implies that $I_1(L) = I_2(L) = \top$. Hence, by construction we have that $J(L) = \top$, which is a contradiction with the fact that $J(L) \neq \top$. Consequently, there is a unique $\subseteq$-minimal model I. Finally, note that, since $\mathcal{P}$ has no weak negation, $\mathcal{P}^I = \mathcal{P}$ and, thus, I is also the unique answer set of $\mathcal{P}$. □

3.2 Weak Completion Semantics

Formally, given a program $\mathcal{P}$, by $\mathbf{Ł}(\mathcal{P})$ we denote the $\mathbf{Ł}$-theory resulting of replacing in $\mathcal{P}$ each occurrence of $\leftarrow$ by $\leftarrow_{\text{Ł}}$. Furthermore, for a normal nested program $\mathcal{P}$, we say that a literal L is *defined in* $\mathcal{P}$ iff $\mathcal{P}$ contains a rule whose head is L; otherwise we say that L is *undefined in* $\mathcal{P}$. The set of rules defining a literal L (those with L in the head) is denoted as $\mathsf{def}(\mathcal{P}, L)$. The set of all literals that are undefined in $\mathcal{P}$ is denoted by $\mathsf{undef}(\mathcal{P})$. We specify the set of defined literals in $\mathcal{P}$ as $\mathit{Head}(\mathcal{P}) = (\mathsf{At} \cup \neg\mathsf{At}) \setminus \mathsf{undef}(\mathcal{P})$. The set $\mathrm{srf}(\mathcal{P})$ of a normal nested program $\mathcal{P}$, is defined as follows:

$$\begin{aligned}\mathrm{srf}(\mathcal{P}) \quad \overset{\text{def}}{=} \quad \{L \leftarrow \varphi \vee \cdots \vee \varphi_n \mid {} & L \in (\mathsf{At} \cup \neg\mathsf{At}) \text{ and} \\ & \mathsf{def}(\mathcal{P}, L) = \{L \leftarrow \varphi, \ldots, L \leftarrow \varphi_n\} \not\emptyset\}.\end{aligned}$$

Below we straightforwardly extend the definition of the *weak completion* [9] to normal nested programs, that is, to programs that may contain rules where the head is a strongly negated literal. The *weak completion* of a normal nested program $\mathcal{P}$, denoted $\mathrm{wc}(\mathcal{P})$, is defined as follows:

$$\mathrm{wc}(\mathcal{P}) \quad \overset{\text{def}}{=} \quad \{L \leftrightarrow \varphi \mid (L \leftarrow \varphi) \in \mathrm{srf}(\mathcal{P})\}$$

Note that $\mathrm{wc}(\mathcal{P})$ is also a normal nested program because we consider that $\varphi \leftrightarrow \psi$ in a program is a shorthand for the two rules $\varphi \leftarrow \psi$ and $\psi \leftarrow \varphi$.

Definition 2. *An interpretation I is called* wc-model *of a normal nested program P iff I is a $\subseteq$-minimal model of $\mathbf{Ł}(wc(\mathcal{P}))$.*

Originally WCS was only defined for basic normal programs, extended with rules of the form $A \leftarrow \bot$, called (negative) *assumption* [9]. Here, we will call these programs, wc-normal programs. Hence, we are only considering programs with one type of negation, which we will show, corresponds to strong negation in the ASP. Note that, as opposed to the ASP, the WCS is defined in terms of the three-valued Łukasiewicz logic instead of classical logic with strong negation.

[9] showed that wc-normal programs always have a unique wc-model which can be computed by the following consequence operator [15]:[4] Given an interpretation I and a wc-normal program $\mathcal{P}$, the application of Φ to I and $\mathcal{P}$, denoted by $\Phi_{\mathcal{P}}(I)$, is an interpretation $J = \langle J^{\top}, J^{\bot}\rangle$ defined as follows:

$$\begin{aligned} J^{\top} &= \{A \mid \text{there is } A \leftarrow \mathit{Body} \in \mathcal{P} \text{ such that } I(\mathit{Body}) = \top\}, \\ J^{\bot} &= \{A \mid \text{there is } A \leftarrow \mathit{Body} \in \mathcal{P} \text{ and} \\ &\qquad\qquad \text{all } A \leftarrow \mathit{Body} \in \mathcal{P} \text{ satisfy } I(\mathit{Body}) = \bot\}. \end{aligned}$$

[4] The correspondence of this unique wc-model and the well-founded model [17] for wc-normal programs without positive cycles, has been shown in [4].

The following example illustrates the WCS by means of two cases of Byrne's suppression task from the introduction.

Example 3. Let $\mathcal{P}_1$ be the wc-normal program consisting of the rules in (1) in the introduction. $\mathbf{Ł}(\mathrm{wc}(\mathcal{P}_1))$ is the following $\mathbf{Ł}$-theory:

$$\ell \leftrightarrow_{\text{Ł}} e \wedge \neg ab_1 \qquad\qquad e \leftrightarrow_{\text{Ł}} \bot \qquad\qquad ab_1 \leftrightarrow_{\text{Ł}} \bot$$

whose unique wc-model is $\langle \emptyset, \{e, \ell, ab_1\}\rangle = \{\neg e, \neg \ell, \neg ab_1\}$. This program illustrates why assumptions such as $e \leftarrow \bot$, though being tautologies in Łukasiewicz logic, are not tautologies under the WCS: After the the weak completion transformation they become equivalences, $e \leftrightarrow_{\text{Ł}} \bot$, and, thus, e has to be false. Note that assumptions can also be overwritten by facts. Let for instance $\mathcal{P}_2 = \mathcal{P}_1 \cup \{e \leftarrow \top\}$ be the program obtained by adding the fact e to the above program. Then, its weak completion $\mathbf{Ł}(\mathrm{wc}(\mathcal{P}_2))$ is as follows:

$$\ell \leftrightarrow_{\text{Ł}} e \wedge \neg ab_1 \qquad\qquad e \leftrightarrow_{\text{Ł}} \bot \vee \top \qquad\qquad ab_1 \leftrightarrow_{\text{Ł}} \bot$$

As $e \leftrightarrow_{\text{Ł}} \bot \vee \top \equiv_{\text{Ł}} e \leftrightarrow_{\text{Ł}} \top$, the unique wc-model of $\mathcal{P}_2$ is $\langle \{e, \ell\}, \{ab_1\}\rangle = \{e, \ell, \neg ab_1\}$, where e and ℓ are true. Let $\mathcal{P}_3$ be the wc-normal program consisting of the rules in (2) in the introduction. $\mathbf{Ł}(\mathrm{wc}(\mathcal{P}_3))$ is the following $\mathbf{Ł}$-theory:

$$\ell \leftrightarrow_{\text{Ł}} (e \wedge \neg ab_1) \vee (t \wedge \neg ab_2) \qquad e \leftrightarrow_{\text{Ł}} \bot \qquad ab_1 \leftrightarrow_{\text{Ł}} \bot \qquad ab_2 \leftrightarrow_{\text{Ł}} \bot$$

whose unique wc-model is $\langle \emptyset, \{e, ab_1, ab_2\}\rangle = \{\neg e, \neg ab_1, \neg ab_2\}$. That is, e, ab_1 and ab_2 are false, while ℓ and t are unknown.

4 Correspondence between ASP and WCS

Let us first discuss the main differences between both semantics according to the two examples of the suppression task in the introduction.

Example 4 (Ex. 3 continued). Consider the wc-normal program $\mathcal{P}_1$: Its corresponding normal program can be obtained by replacing every assumption of the form $A \leftarrow \bot$ in $\mathcal{P}_1$ by a fact with strong negation $\neg A$. The resulting program, $\mathcal{P}_4$ consists of the following rules:

$$\ell \leftarrow e \wedge \neg ab_1 \qquad\qquad \neg e \qquad\qquad \neg ab_1 \tag{15}$$

Its unique answer set is $\langle \emptyset, \{e, ab_1\}\rangle = \{\neg e, \neg ab_1\}$, which does not coincide with the wc-model of $\mathcal{P}_1$, as ℓ is false under the WCS, but unknown under the ASP.

The above example illustrates that replacing assumptions by strong negation facts is not enough to obtain the same results between WCS and ASP.

As mentioned previously the ASP and WCS can be respectively defined in terms of classical logic with strong negation and three-valued Łukasiewicz logic. Interestingly, Vakarelov [16] showed that there is a correspondence between $\mathbf{Ł}$-logic and $\mathbf{N}$-logic, in the sense that all connectives of one logic are definable in terms of the other one. In particular, here we are interested in translating from the WCS to the ASP and, thus, that implies a translation from Ł-logic to **N**-logic. Formally, given a Ł-theory Γ, by $\tilde{\mathbf{N}}(\Gamma)$ we denote the result of replacing in Γ every occurrence of $\varphi \leftarrow_{\text{L}} \psi$ by $(\varphi \leftarrow_{\text{CL}} \psi) \wedge (\neg\psi \leftarrow_{\text{CL}} \neg\varphi)$.

Theorem 1 (Theorem 11 in [16]). *Given any* **Ł***-theory Γ, an interpretation I is a model of Γ under* **Ł***-logic iff I is a model of* $\mathbf{N}(\Gamma)$ *under* **N***-logic.*

Based on this result, we can establish the correspondence between the ASP and the WCS.

We need rules that *negatively complete* the information of the given program. Let us now formalize this idea by defining the definition completion of a program.

Definition 3. *Given a normal nested program $\mathcal{P}$, its* definition completion *is defined as follows:*

$$\tilde{\mathcal{P}} \stackrel{\text{def}}{=} \mathcal{P} \cup \{\neg L \leftarrow \neg\varphi \mid (L \leftarrow \varphi) \in \mathit{srf}(\mathcal{P})\} \tag{16}$$

Let us apply the suggested characterization for the programs in Example 4.

Example 5 (Ex. 4 continued). Given $\mathcal{P}_4$, its definition completion is as follows:

$$\tilde{\mathcal{P}_4} = \mathcal{P}_4 \cup \{ \quad \neg\ell \leftarrow \neg\,(e \wedge \neg ab_1), \quad \neg\neg e \leftarrow \neg\top, \quad \neg\neg ab_1 \leftarrow \neg\top \quad \}$$

Note that $\neg\ell \leftarrow (\neg\, e \wedge \neg ab_1)$ is equivalent to $\neg\ell \leftarrow \neg e \vee ab_1$, while the last two rules are tautologies under the ASP. The unique answer set of $\tilde{\mathcal{P}_4}$ corresponds to the unique wc-model of $\mathcal{P}_1$.

4.1 Characterization of WCS in Terms of ASP

We will now introduce some auxiliary results that will help us to show the correspondence between WCS and ASP. Let us start by showing that the answer sets of any program coincide with the answer sets of its weak completion. The proof of this statement relies on the the following lemma which is a straightforward lifting of the Completion Lemma from [6, p. 23] to the class of programs with strong negation.

Lemma 1. *Let $\mathcal{P}$ be any program, let At be any set of atoms (not necessarily equal to $\mathsf{at}(\mathcal{P})$) and let $Q \subseteq (\mathsf{At} \cup \neg\mathsf{At})$ be any set of literals such that $Q \cap \mathrm{Head}(\mathcal{P}) = \emptyset$. Let φ_L be some implication-free formula for each literal $L \in Q$ and I be an interpretation. Then, the following two statements are equivalent:*

1. *I is an answer set of $\mathcal{P} \cup \{\ L \leftarrow \varphi_L \ \big|\ L \in Q\ \}$.*
2. *I is an answer set of $\mathcal{P} \cup \{\ L \leftrightarrow \varphi_L \ \big|\ L \in Q\ \}$.*

Proposition 4. *Given any normal nested program $\mathcal{P}$, an interpretation I is an answer set of $\mathcal{P}$ if and only if I is an answer set of $\mathit{wc}(\mathcal{P})$.*

Proof. Assume that $\mathcal{P}$ is regular. From item (ii) of Proposition 6 in [10], $\mathcal{P}$ and $\mathrm{srf}(\mathcal{P})$ have the same answer sets. Note that there is a unique rule with head L for each literal $L \in (\mathsf{At} \cup \neg\mathsf{At})$ in $\mathrm{srf}(\mathcal{P})$. Hence, $\mathrm{wc}(\mathcal{P})$ is obtained by replacing all occurrences of $\leftarrow$ in $\mathrm{srf}(\mathcal{P})$ by $\leftrightarrow$ and, from Lemma 1 (by taking $\mathcal{P} = \emptyset$), $\mathrm{srf}(\mathcal{P})$ and $\mathrm{wc}(\mathcal{P})$ have the same answer sets. In case that $\mathcal{P}$ is not regular, we have that I is an answer set of $\mathcal{P}$ iff I is an answer set of $\mathrm{reg}(\mathcal{P})$ iff I is an answer set of $\mathrm{wc}(\mathrm{reg}(\mathcal{P})) = \mathrm{reg}(\mathrm{wc}(\mathcal{P}))$ iff I is an answer set of $\mathrm{wc}(\mathcal{P})$. □

Lemma 2. *Given any normal nested logic program $\mathcal{P}$, an interpretation I is a model of $\mathbf{N}(wc(\tilde{\mathcal{P}}))$ under $\mathbf{N}$-logic if and only if I is a model of $\mathbf{Ł}(wc(\mathcal{P}))$.*

Proof. Note that $\mathbf{N}(\mathrm{wc}(\tilde{\mathcal{P}}))$ has a pair of equivalences of the form

$$A \leftrightarrow_{\mathrm{CL}} \varphi_1 \vee \cdots \vee \varphi_n \tag{17}$$

$$\neg A \leftrightarrow_{\mathrm{CL}} \neg\varphi_1 \wedge \cdots \wedge \neg\varphi_n \tag{18}$$

for each $A \in \mathsf{At}$ with $\mathsf{def}(\mathcal{P}, L) = \{A \leftarrow \varphi_1, \ldots, A \leftarrow \varphi_n\} \neq \emptyset$. On the other hand, we have that $\mathbf{Ł}(\mathrm{wc}(\mathcal{P}))$ has an equivalences of the form

$$A \leftrightarrow_{\mathrm{L}} \varphi_1 \vee \cdots \vee \varphi_n \tag{19}$$

for each atom $A \in \mathsf{At}$ with $\mathsf{def}(\mathcal{P}, L) = \{A \leftarrow \varphi_1, \ldots, A \leftarrow \varphi_n\} \neq \emptyset$. By Theorem 1, the models of $\mathbf{Ł}(\mathrm{wc}(\mathcal{P}))$ under $\mathbf{Ł}$-logic and the models of $\mathbf{N}(\mathbf{Ł}(\mathrm{wc}(\mathcal{P})))$ under $\mathbf{N}$-logic coincide. Note now that, by definition, we have that (19) is equivalent to the following formula

$$(A \leftarrow_{\mathrm{L}} \varphi_1 \vee \ldots \vee \varphi_n) \wedge (\varphi_1 \vee \ldots \vee \varphi_n \leftarrow_{\mathrm{L}} A) \tag{20}$$

Hence, $\mathbf{N}(\mathbf{Ł}(\mathrm{wc}(\mathcal{P})))$ contains a formula of the form $\psi_1^A \wedge \psi_2^A \wedge \psi_3^A \wedge \psi_4^A$ for each atom $A \in \mathsf{At}$ with $\mathsf{def}(\mathcal{P}, A) = \{A \leftarrow \varphi_1, \ldots, A \leftarrow \varphi_n\} \neq \emptyset$ where

$$\psi_1^A \stackrel{\mathrm{def}}{=} A \leftarrow_{\mathrm{CL}} \varphi_1 \vee \cdots \vee \varphi_n \tag{21}$$

$$\psi_2^A \stackrel{\mathrm{def}}{=} \neg A \leftarrow_{\mathrm{CL}} \neg(\varphi_1 \wedge \cdots \wedge \varphi_n) \tag{22}$$

$$\psi_3^A \stackrel{\mathrm{def}}{=} \varphi_1 \vee \cdots \vee \varphi_n \leftarrow_{\mathrm{CL}} A \tag{23}$$

$$\psi_4^A \stackrel{\mathrm{def}}{=} \neg(\varphi_1 \wedge \cdots \wedge \varphi_n) \leftarrow_{\mathrm{CL}} \neg A \tag{24}$$

Note that, by definition, $\psi_1^A \wedge \psi_3^A$ is equivalent to (17). Besides $\psi_3^A \wedge \psi_4^A$ can be equivalently rewritten as

$$\neg A \leftrightarrow_{\mathrm{CL}} \neg\,(\varphi_1 \vee \ldots \vee \varphi_n) \tag{25}$$

that is equivalent to (18), i.e. $\mathbf{Ł}(\mathrm{wc}(\mathcal{P}))$ and $\mathbf{N}(\mathrm{wc}(\tilde{\mathcal{P}}))$ have the same models. □

Proposition 5. *Given any normal nested program $\mathcal{P}$ and interpretation I, if I is an answer set of $\tilde{\mathcal{P}}$, then I is a model of $\mathbf{Ł}(wc(\mathcal{P}))$.*

Proof. From Lemma 2, I is a model of $\mathbf{N}(\mathrm{wc}(\tilde{\mathcal{P}}))$ iff I is a model of $\mathbf{Ł}(\mathrm{wc}(\mathcal{P}))$. From Proposition 4, the answer sets of $\tilde{\mathcal{P}}$ and $\mathrm{wc}(\tilde{\mathcal{P}})$ are the same. Furthermore, from Proposition 2, the answer sets of $\mathrm{wc}(\tilde{\mathcal{P}})$ are models of $\mathbf{N}(\mathrm{wc}(\tilde{\mathcal{P}}))$ under $\mathbf{N}$-logic. Hence, the answer sets of $\tilde{\mathcal{P}}$ are wc-models of $\mathcal{P}$. □

Given Lemma 2, Proposition 3,4 and 5, we can now show how the definition completion of a program precisely characterizes the WCS in terms of the ASP.

Theorem 2. *Given any wc-normal program $\mathcal{P}$ and interpretation I, the following two statements are equivalent:*

1. *I is the unique wc-model of $\mathcal{P}$,*
2. *I is the unique answer set of $\tilde{\mathcal{P}}$.*

The following two statements are also equivalent:

1. *$\mathcal{P}$ has no wc-model,*
2. *$\tilde{\mathcal{P}}$ has no answer set.*

Proof. Assume that $\mathbf{N}(\mathrm{wc}(\tilde{\mathcal{P}}))$ has no model. From Lemma 2, it follows that $\mathbf{Ł}(\mathrm{wc}(\mathcal{P}))$ has no model either and, thus, there is no wc-model of $\mathcal{P}$. Besides, from Proposition 5, the lack of model of $\mathbf{Ł}(\mathrm{wc}(\mathcal{P}))$ also implies that $\tilde{\mathcal{P}}$ has no answer set. Otherwise, $\mathbf{N}(\mathrm{wc}(\tilde{\mathcal{P}}))$ has a model and, since is a wc-normal program and thus basic, from Proposition 3, we get that there is an interpretation I which the $\subseteq$-least model of $\mathbf{N}(\mathrm{wc}(\tilde{\mathcal{P}}))$ and, thus, the unique answer set of $\mathrm{wc}(\tilde{\mathcal{P}})$. From Proposition 4 this implies that I is the unique answer set of $\tilde{\mathcal{P}}$. Furthermore, from Lemma 2, this also implies that it is the $\subseteq$-least model of $\mathbf{Ł}(\mathrm{wc}(\mathcal{P}))$ and, thus, the unique wc-model of $\mathcal{P}$. □

5 Conclusions and Future Work

We have shown how logic programs under the Weak Completion Semantics can be translated into logic programs under the Answer Set Semantics by using the *definition completion*. This completion adds rules supporting the strong negation of a defined atom whenever all the bodies of all rules defining it are false. This transformation has been illustrated by two examples of Byrne's suppression task.

This result allows us to use all the knowledge representation features of Answer Set Programming, including *default negation*, in combination with this completion and opens two interesting future possibilities: On the one hand, in [5], logic programs under the Weak Completion Semantics were extended with a context operator to capture negation as failure. Hence, an immediate question is whether these *contextual logic programs* can also be translated into logic programs under the Answer Set Semantics by using default negation. On the other hand, it would be interesting to investigate how the twelve cases of the suppression task could be represented by means of default negation or the context operator in order to ensure *elaboration tolerance* [12].

Another interesting observation is that the proof of the correspondence between these two semantics relies on the use of strong negation as a connective in its own right, as opposed to the usual convention of considering that strong negation can only be applied to atoms. This extension was first considered by David Pearce in [14]. It would be worth to investigate how the usual properties of the Answer Set Semantics can be extended to this class of programs and how its use can ease other knowledge representation problems.

Acknowledgements. We are thankful to David Pearce for pointing to Vakarelov's work on the relation between Łukasiewicz logic and classical logic with strong negation.

References

1. Baral, C.: Knowledge Representation, Reasoning, and Declarative Problem Solving. Cambridge University Press, New York, NY, USA (2003)
2. Byrne, R.M.J.: Suppressing valid inferences with conditionals. Cognition 31, 61–83 (1989)
3. Dietz, E.A., Hölldobler, S., Ragni, M.: A computational logic approach to the suppression task. In: Miyake, N., Peebles, D., Cooper, R.P. (eds.) Proc. of the 34th Annual Conference of the Cognitive Science Society, CogSci 2012. pp. 1500–1505. Cognitive Science Society (2012)
4. Dietz, E.A., Hölldobler, S., Wernhard, C.: Modeling the suppression task under weak completion and well-founded semantics. Journal of Applied Non-Classsical Logics 24(1–2), 61–85 (2014)
5. Dietz Saldanha, E.A., Hölldobler, S., Pereira, L.M.: Contextual reasoning: Usually birds can abductively fly. In: Proceedings of 14th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR). pp. 64–77. Springer International Publishing (2017)
6. Ferraris, P., Lifschitz, V.: Weight constraints as nested expressions. TPLP 5(1-2), 45–74 (2005), https://doi.org/10.1017/S1471068403001923
7. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing 9(3), 365–385 (Aug 1991)
8. Hölldobler, S.: Weak completion semantics and its applications in human reasoning. In: Furbach, U., Schon, C. (eds.) Proc. of the Workshop on Bridging the Gap between Human and Automated Reasoning. CEUR Workshop Proc., vol. 1412, pp. 2–16. CEUR-WS.org (2015)
9. Hölldobler, S., Kencana Ramli, C.D.: Logic programs under three-valued Łukasiewicz semantics. In: Hill, P.M., Warren, D.S. (eds.) 25th int. Conference on Logic Programming. LNCS, vol. 5649, pp. 464–478. Springer (2009)
10. Lifschitz, V., Tang, L.R., Turner, H.: Nested expressions in logic programs. Ann. Math. Artif. Intell. 25(3-4), 369–389 (1999), https://doi.org/10.1023/A:1018978005636
11. Łukasiewicz, J.: O logice trójwartościowej. Ruch Filozoficzny 5, 169–171 (1920)
12. McCarthy, J.: Elaboration tolerance. In: Common Sense. vol. 98. Citeseer (1998)
13. Nelson, D.: Constructible falsity. Journal of Symbolic Logic 14(1), 16–26 (03 1949)
14. Pearce, D.: A new logical characterisation of stable models and answer sets. In: Dix, J., Pereira, L.M., Przymusinski, T.C. (eds.) Non-Monotonic Extensions of Logic Programming. pp. 57–70. Springer Berlin Heidelberg, Berlin, Heidelberg (1997)
15. Stenning, K., van Lambalgen, M.: Human Reasoning and Cognitive Science. A Bradford Book, MIT Press (2008)
16. Vakarelov, D.: Notes on n-lattices and constructive logic with strong negation. Studia logica 36(1-2), 109–125 (1977)
17. Van Gelder, A., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. Journal of the ACM 38(3), 619–649 (1991)