On Contextual Programs Under Three-Valued Łukasiewicz Logic

Islam Hamada[*]

Technische Universität Dresden, 01062 Dresden, Germany

Abstract
Weak completion semantics is based on a cognitive theory that managed to model human reasoning in a variety of scenarios. It is based on a non-monotonic theory that uses normal logic programs under three-valued Łukasiewicz logic. We discuss contextual programs in this paper, in that it will be shown that the monotonicity of their semantic operators is generally undecidable. It will also be shown that the fixed point of the semantic operator of acyclic contextual programs is not only a model but a minimal model.

Keywords
human reasoning, weak completion semantics, Łukasiewicz logic, contextual programs, monotonicity, minimal model

1. Introduction

The *weak completion semantics* (WCS) is a cognitive theory that models human reasoning using normal logic programs [1]. It is a non-monotonic theory that uses three-valued Łukasiewicz logic. It was first used to model the suppression task [1], then it has been further developed to model human choices in the selection task [2], syllogistic reasoning [3], and ethical decision problems [4]. The semantics can model different types of conditionals [5, 6, 7], as well as disjunctions [8]. It has an operational semantics given by the implementations in Prolog, Python and ASP. Finally the (least) model one reasons with respect to can be computed by a connectionist network [9]. Consider the following example

If today is Sunday, Jack will rest the whole day. Today is Sunday. What do we conclude?

The example will be modeled as a program following Stenning and van Lambalgen in [10]. The first statement is represented by $\{r \leftarrow s \wedge \neg ab_s, ab_s \leftarrow \bot\}$, where r and s denote that *Jack will rest the whole day* and *Today is Sunday*, respectively, and ab_s is an abnormality predicate which is assumed to be false represented by $ab_s \leftarrow \bot$. The second statement is represented by $s \leftarrow \top$, which means s should be mapped to true in the model. Consequently, $s \wedge \neg ab_s$ is mapped to true, therefore r is also mapped to true. So one reasons w.r.t. the model $\langle\{r, s\}, \{ab_s\}\rangle$. The model specifies that r and s are mapped to true, whereas ab_s is mapped to false, and there are no atoms

8th Workshop on Formal and Cognitive Reasoning, September 19, 2022, Trier, Germany
*Corresponding author.
islamhamada222@gmail.com (I. Hamada)
0000-0002-6730-1743 (I. Hamada)

mapped to unknown because there are no other atoms left. Consequently one concludes that *Jack will rest the whole day*.
Sometime ago an operator known as the *context operator (ctxt)* was added to WCS in [10]. Programs that use the context operator are called *contextual programs*. Contextual programs are useful in cases, where default reasoning is involved [11]. The context operator *ctxt* assumes an atom to be false when it is neither assigned true or false, which mimics negation as failure. For example we will extend the previous example into the following:

If today is Sunday, Jack will rest the whole day, unless he knows an exam is coming. If the semester is ending soon, and exam is coming soon. Today is Sunday. What do we conclude?

The first sentence is represented by $\{r \leftarrow s \wedge \neg ab_s, ab_s \leftarrow ctxt\, e\}$, where atom e denotes that *an exam is coming*, and the other atoms denote the same as before. The second sentence is represented by $\{e \leftarrow d \wedge \neg ab_d, ab_d \leftarrow \bot\}$, where d denotes *the semester is ending*. The last sentence is represented by $s \leftarrow \top$. Note that ab_d is mapped to false, while d is unknown, so $d \wedge \neg ab_d$ is unknown, hence e is unknown too. This means that $ctxt\, e$ is mapped to false, consequently ab_s is mapped to false. This means that $s \wedge \neg ab_s$ is mapped to true, hence r is also mapped to false. The model we just built is $\langle\{s, r\}, \{ab_d, ab_s\}$, from which we conclude that *Jack will rest the whole day*. Note that if we remove the context operator from the previous program, we will get the model $\langle\{s\}, \{ab_d\}\rangle$ instead, from which we conclude that it is unknown whether *Jack will rest the whole day*. This means that the operator *ctxt* preceding the atom e enables us to assume that *There is no coming exam* by default, since Jack lacks knowledge about whether *an exam is coming*, hence the utility of the context operator.

Despite their utility, contextual programs gave rise to some questions because they do not share the same properties that non-contextual programs have. For example the models we computed in the two previous examples can be computed using a semantic operator in WCS. This semantic operator is always monotonic for non-contextual programs [12, 10]. Consequently, the semantic operator of a non-contextual program always has a (unique) least fixed point [12]. This fixed point was shown to be the least model of the weak completion of the program. Furthermore, reasoning in WCS is done w.r.t. this fixed point, i.e., w.r.t. the least model of the weak completion of programs. On the other hand, contextual programs often do not have monotonic semantic operators, hence they sometimes do not have any fixed points. This explains our interest in deciding whether a given (contextual) program has a monotonic semantic operator because if a (contextual) program has a monotonic semantic operator, then its semantic operator indeed has a unique fixed by the Knaster-Tarski Fixed Point Theorem [12]. This will be later shown to be an undecidable problem. Finally, we consider a certain subclass of contextual programs, the class of acyclic contextual programs. This is because the semantic operator of an acyclic contextual program always has a unique fixed point [11, 13], this fixed point was shown to be a model of the weak completion of the program, and was conjectured to be a minimal model. In the weak completion semantics minimal models are preferred because a minimal model does not assign true or false to any atoms unnecessarily, i.e., a minimal model is minimal in information (knowledge) ordering. For example in the reasoning scenario above there is nothing about the premises that suggests whether an *exam is coming*, so one should expect the model that we reason with respect to to map e to U, which was the case. It will be shown that the fixed point of the semantic operator of an acyclic contextual contextual is indeed a minimal model.

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

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

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

$\leftarrow$	$\top$	U	$\bot$
$\top$	$\top$	$\top$	$\top$
U	U	$\top$	$\top$
$\bot$	$\bot$	U	$\top$

$\leftrightarrow$	$\top$	U	$\bot$
$\top$	$\top$	U	$\bot$
U	U	$\top$	U
$\bot$	$\bot$	U	$\top$

e	$ctxt\, e$
$\top$	$\top$
$\bot$	$\bot$
U	$\bot$

Table 1
The truth tables of the connectives $\neg$, $\wedge$, $\vee$, $\leftarrow$ and $\leftrightarrow$ in three-valued Łukasiewicz logic [15], plus the truth table of the operator *ctxt*. Note that F denotes a formula, while L denotes a non-contextual literal. Cells where one of the operands is mapped to U are in gray.

2. The Weak Completion Semantics

We assume the reader to be familiar with classical logic [14]. Let $\top$, $\bot$ and U be truth constants denoting *true, false, unknown*, respectively. A non-contextual *literal* is an atom or its negation. A contextual literal has the form $ctxt\, A$ or $\neg ctxt\, A$ where A is a non-contextual literal. A (logic) program is a finite set of clauses of the form $B \leftarrow Body$, where B is an atom and $Body$ is either $\top$, $\bot$, or a finite non-empty conjunction of literals. Clauses of the form $B \leftarrow \top$, $B \leftarrow \bot$ and $B \leftarrow L_1 \wedge \ldots \wedge L_n$ are called *facts, assumptions* and *rules*, respectively, where L_i, $1 \leq i \leq n$, are literals. In this paper, $\mathcal{P}$ denotes a program. Program $\mathcal{P}$ is contextual if it has at least one occurrence of a contextual literal, otherwise it is a non-contextual program. An atom B is defined if and only if $\mathcal{P}$ contains a clause of the form $B \leftarrow Body$. The truth tables needed throughout the paper are shown in Table 1.

Consider the following transformation: (1) For all defined atoms B occurring in $\mathcal{P}$, replace all clauses of the form $B \leftarrow Body_1$, $B \leftarrow Body_1 \vee Body_2 \vee \ldots$ (2) Replace all occurrences of $\leftarrow$ by $\leftrightarrow$. The resulting set of equivalences is called the *weak completion* of $\mathcal{P}$: for short, $wc\mathcal{P}$. It differs from the completion defined by [16] in that undefined atoms are not mapped to false, but to unknown instead. As shown by Hölldobler and Kencana Ramli [12], each weakly completed non-contextual program admits a least model under three-valued Łukasiewicz [12]. This model is denoted by $\mathcal{M}_{wc\mathcal{P}}$. It can be computed as the least fixed point of a semantic operator introduced by Stenning and van Lambalgen [10]. Let $\mathcal{P}$ be a program and I a three-valued interpretation represented by the pair $\langle I^\top, I^\bot \rangle$, where $I^\top$ and $I^\bot$ are the sets of atoms mapped to true and false by I, respectively, and atoms which are not listed are mapped to unknown by I. An interpretation $I = \langle I^\top, I^\bot \rangle$ is subsumed by interpretation $I' = \langle I'^\top, I'^\bot \rangle$, i.e., $I \subseteq I'$, iff $I^\top \subseteq I'^\top$ and $I^\bot \subseteq I'^\bot$.

Definition 1. *For a program $\mathcal{P}$ and an interpretation I, $\Phi_\mathcal{P} I = \langle J^\top, J^\bot \rangle$[1], where*

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

where $g(\mathcal{P})$ denote the grounding of program $\mathcal{P}$.

Note that $g(\mathcal{P})$ was used in the above definition instead of $\mathcal{P}$, in case $\mathcal{P}$ is first-order. Note also that $g(F)$ denotes the set of ground instances of formula F, and that for a program $\mathcal{P}$,

[1]Whenever we apply a unary operator like $\Phi_\mathcal{P}$ to an argument like I, we omit parenthesis and write $\Phi_\mathcal{P} I$ instead.

$g(\mathcal{P})$ denotes the program containing all the ground instances of each clause occurring in $\mathcal{P}$. the semantic operator of non-contextual programs was shown to be *monotonic*, i.e., let I and I' be two interpretations, if $I \subseteq I'$, then $\Phi_{\mathcal{P}}I \subseteq \Phi_{\mathcal{P}}I'$, hence the existence of a least fixed point of the semantic operator for non-contextual programs. However, the semantic operator is often not monotonic for contextual programs, and sometimes does not have any fixed points. Nevertheless, we are still interested in the unique fixed points of their semantic operators, if they exist. The class of acyclic contextual programs is a subclass of programs where a unique fixed point of the semantic operator always exists. A program $\mathcal{P}$ is *acyclic* iff $\mathcal{P}$ is *acyclic* w.r.t. some *level mapping*. A *level mapping lvl* is a mapping from the set of ground atoms to the set $\mathbb{N}$ of natural numbers. For our purpose we extend the level mapping *lvl* by $lvl\,\neg A = lvl\,A = lvl\,ctxt\,A = lvl\,ctxt\neg A = lvl\,\neg ctxt\,A = lvl\,\neg ctxt\neg A$. $\mathcal{P}$ is *acyclic w.r.t. lvl* if and only if for every rule $A \leftarrow L_1 \wedge \ldots \wedge L_n$ occurring in $g(\mathcal{P})$, we find $lvl\,A \geq lvl\,L_i$ for all $1 \leq i \leq n$. Note that $lfp(\Phi_{\mathcal{P}})$ denotes the least fixed point of the semantic operator $\Phi_{\mathcal{P}}$, and it is usually used in the context of acyclic contextual programs.

3. Fixed Points Are Minimal Models

The semantic operator of acyclic programs (contextual or non-contextual) is a contraction by the Banach Contraction Mapping Theorem [13]. Consequently, it has a unique fixed point that can be reached by iterating the semantic operator starting with any interpretation. This fixed point has been shown to be a model of the weak completion of contextual programs [11]. We will also show that it is a minimal model in this section.

3.1. Alternative Definition of The Semantic Operator

Definition 1 happens to be equivalent to the following:

Definition 2. *For any program $\mathcal{P}$, an interpretation I, and a ground atom A*

$$\Phi_{\mathcal{P}}I(A) = \begin{cases} I(D) & \text{if } A \text{ is defined in } \mathcal{P} \text{ and } A \leftrightarrow D \in wc(g(\mathcal{P})), \\ \mathsf{U} & \text{if } A \text{ is undefined in } g(\mathcal{P}). \end{cases}$$

Which means that after an iteration of the semantic operator the mapping of a defined atom A is the same as the mapping of the right-hand side of the equivalence in $wc(g(\mathcal{P}))$, which A happens to be its left-hand side. If A is undefined, then it is mapped to U. The equivalence of Definition 1 and Definition 2 can be verified by a direct comparison of both definitions.

3.2. The Least Fixed Point $lfp(\Phi_{\mathcal{P}})$ Is a Minimal Model

The least fixed point $lfp(\Phi_{\mathcal{P}})$ of an acyclic contextual program $\mathcal{P}$ can be reached by iterating the semantic operator starting with any interpretation. Consider the following program $\mathcal{P} = \{a \leftarrow b \wedge c,\ b \leftarrow \top,\ c \leftarrow \bot,\ d \leftarrow e\}$. Iterating the semantic operator starting with the empty interpretation $I = \langle \emptyset, \emptyset \rangle$ leads to the fixed point through the following sequence: $I = \langle \emptyset, \emptyset \rangle \rightarrow \Phi_{\mathcal{P}}I = \langle \{b\}, \{c\} \rangle \rightarrow \Phi^2_{\mathcal{P}}I = \langle \{b\}, \{c, a\} \rangle \rightarrow \Phi^3_{\mathcal{P}}I = \Phi^2_{\mathcal{P}}I$. Note that $\Phi_{\mathcal{P}}I$ maps b to $\top$ by the second clause and maps c to $\bot$ by the third clause. Consequently in the next iteration $\Phi^2_{\mathcal{P}}$ maps a to

$\bot$ by the first clause because $b \wedge c$ is mapped to $\bot$. Note that $\Phi_{\mathcal{P}} I$ maps a to U by the first and due to the fact that $b \wedge c$ was still mapped to U by I. Another observation is that through the sequence of interpretations once an atom got mapped to $\top$ or $\bot$, this mapping does not change in the following iteration. This happens because when an atom gets mapped to $\top$ or $\bot$, this is a necessary consequence of the program itself. To elaborate b is mapped to $\bot$ because of the second clause, c is mapped to $\top$ because of the third clause, finally a is mapped to $\bot$ by the first clause and due to the propagation of the necessary mappings of a and b through the iteration process. All the mappings to $\bot$ or $\top$ were necessary due to the program itself. This (least) fixed point is minimal w.r.t. the mappings of atoms to $\bot$ or $\top$, i.e., the fixed point has to be a minimal model of the weak completion of the program as we will see later.

Proposition 1. *Let $\mathcal{P}$ be a contextual program such that $\Phi_{\mathcal{P}}$ has a unique fixed point M. If M' is a model for $wc\mathcal{P}$ and $M' \neq M$, then M' must map an undefined atom to true or false.*

Proof. Let $\mathcal{P}$ be a contextual program, M the unique fixed point of $\Phi_{\mathcal{P}}$, and M' be a model for $wc\mathcal{P}$ such that $M' \neq M$. Because M' is a model for $wc\mathcal{P}$, it must map each equivalence $A \leftrightarrow F$ occurring in $wc\mathcal{P}$ to true. The latter holds if and only if $M'(A) = M'(F)$ for each equivalence $A \leftrightarrow F$ occurring in $wc\mathcal{P}$. Because $M' \neq M$ and M is the unique fixed point of $\Phi_{\mathcal{P}}$, we find a ground atom B such that $M'(B) \neq \Phi_{\mathcal{P}} M'(B)$. Assume that B is defined. In this case we find $B \leftrightarrow F \in wc\mathcal{P}$. Because M' is a model for $wc\mathcal{P}$, we find $M'(B) = M'(F)$. But $M'(F) = \Phi_{\mathcal{P}} M'(B)$ by Definition 2 and, thus, $M'(B) = \Phi_{\mathcal{P}} M'(B)$. Consequently, our assumption is wrong and we conclude that B must be undefined. In this case, by definition, $\Phi_{\mathcal{P}} M'(B) = \mathsf{U}$. Consequently, $M'(B)$ must be either true or false. $\square$

Consider the following contextual program: $\mathcal{P} = \{a \leftarrow \neg b \wedge ctxt\, c,\ a \leftarrow \bot,\ b \leftarrow \bot\}$. Program $\mathcal{P}$ is acyclic because it acyclic w.r.t. the following level mapping $lvl\ :\ lvl\, a = 1,\ lvl\, b = 0,\ lvl\, c = 0$. Therefore $\Phi_{\mathcal{P}}$ has a unique fixed point, namely $lfp(\Phi_{\mathcal{P}}) = \langle \emptyset, \{a, b\} \rangle$, which is a model of the weak completion $wc\mathcal{P}$. There are two other models, namely $M_1 = \langle \emptyset, \{a, b, c\} \rangle$ and $M_2 = \langle \{a, c\}, \{b\} \rangle$. Model M_1 maps the undefined atom c to $\bot$, and model M_2 maps the undefined atom c to $\top$. Now we can proceed to prove the main theorem.

Theorem 1. *Let $\mathcal{P}$ be a contextual program. If $\mathcal{P}$ is acyclic, then the unique fixed point of $\Phi_{\mathcal{P}}$ is a minimal model of $wc\mathcal{P}$.*

Proof. Let $\mathcal{P}$ be an acyclic contextual program. The semantic operator $\Phi_{\mathcal{P}}$ has a unique fixed point. Let M be this fixed point. Assume that M is not a minimal model for $wc\mathcal{P}$. Then, we find a model M' for $wc\mathcal{P}$ such that $M' \subseteq M$. By Proposition 1, we find an atom B such that B is undefined in $\mathcal{P}$ and M' maps B to either true or false. Because $M' \subseteq M$, M must map B to either true or false, too. However, because B is undefined in $\mathcal{P}$, $\Phi_{\mathcal{P}} M$ will map B to U, in which case $\Phi_{\mathcal{P}} M \neq M$. In other words, M is not a fixed point anymore and, thus, our assumption is false. Consequently, M is a minimal model for $wc\mathcal{P}$. $\square$

4. Undecidability of The Monotonicity Problem

The monotonicity problem refers to the question whether a given contextual program has a monotonic semantic operator. It is an important property because whenever the semantic operator is monotonic, the Knaster-Tarski Fixed Point Theorem guarantees the existence of a least fixed point of the semantic operator [12]. This problem will be shown to be undecidable by a reduction from the Post correspondence problem.

4.1. The Monotonicity Problem

The *monotonicity problem* is to decide for a given contextual program whether it has a monotonic semantic operator. Some contextual programs have monotonic semantic operators, while some others have non-monotonic semantic operators. Consider the following contextual program $\mathscr{P}_1 = \{a \leftarrow c,\ a \leftarrow ctxt\,b\}$. Program $\mathscr{P}_1$ has a non-monotonic semantic operator because we can find the two interpretations $I = \langle \emptyset, \{c\} \rangle$ and $I' = \langle \{b\}, \{c\} \rangle$ such that $I \subseteq I'$ and $\Phi_{\mathscr{P}_1} I = \langle \emptyset, \{a\} \rangle \nsubseteq \Phi_{\mathscr{P}_1} I' = \langle \{a\}, \emptyset \rangle$. On the other hand the following contextual program $\mathscr{P}_2 = \{a \leftarrow \top,\ a \leftarrow ctxt\,b\}$ has a monotonic semantic operator because we cannot find any two interpretations I and I' where $I \subseteq I'$ and $\Phi_{\mathscr{P}_2} I \nsubseteq \Phi_{\mathscr{P}_2} I'$. Similarly the following program $\mathscr{P}_3 = \{a \leftarrow \neg c,\ a \leftarrow c,\ a \leftarrow ctxt\,b\}$ has a monotonic semantic operator too. Is the monotonicity problem decidable? The monotonicity problem is easily decidable for any propositional program because we can verify for a given program $\mathscr{P}$ whether all pairs of interpretations I and I' satisfy the following statement: if $I \subseteq I'$, then $\Phi_{\mathscr{P}} I \subseteq \Phi_{\mathscr{P}} I'$. This constitutes a sound, complete and terminating procedure for the monotonicity problem because there is a finite number of interpretations that can be defined over the alphabet of a propositional program. However, this is not the case for first-order programs with infinite Herbrand Base because the previous procedure is not terminating due to the possibly infinite number of interpretations. It will shown that the monotonicity problem is undecidable by showing that it is undecidable for a subclass of contextual programs, named class $\mathscr{V}$.

4.2. Class $\mathscr{V}$ of Programs

A program belonging to class $\mathscr{V}$ has the following form

$$\{r \leftarrow body_1,\ r \leftarrow body_2, \ldots,\ r \leftarrow body_n,\ r \leftarrow ctxt\,q\},$$

where r and q are nullary predicate symbols and $body_i$ is a non-contextual conjunction of literals such that r and q do not occur in $body_i$ for all $1 \le i \le n$, and $n \in \mathbb{Z}^+$. Note the three contextual program $\mathscr{P}_1$, $\mathscr{P}_2$ and $\mathscr{P}_3$ that we used in the previous subsection belong to class $\mathscr{V}$. Note also that every program of class $\mathscr{V}$ has clauses that share the same head. We are interested in class $\mathscr{V}$ because a program $\mathscr{P}$ of class $\mathscr{V}$ is non-monotonic iff $\mathscr{P}$ is equivalent to $\mathscr{P}' \cup \mathscr{P}_v$, where $\mathscr{P}_v$ belongs to class $\mathscr{V}$, and $\mathscr{P}'$ shares no defined atoms with $\mathscr{P}_v$, in case $\mathscr{P}'$ is not empty, and most importantly $\mathscr{P}_v$ has a non-monotonic semantic operator too. For example program $\mathscr{P} = \{a \leftarrow c,\ a \leftarrow ctxt\,b,\ b \leftarrow \neg c \wedge ctxt\,a,\ c \leftarrow \top\}$ has a non-monotonic semantic operator $\Phi_{\mathscr{P}}$ because $\mathscr{P} = \mathscr{P}' \cup \mathscr{P}_v$, where $\mathscr{P}' = \{b \leftarrow \neg c \wedge ctxt\,a,\ c \leftarrow \top\}$ and $\mathscr{P}_v = \{a \leftarrow c,\ a \leftarrow ctxt\,b\}$, and $\mathscr{P}_v$ belongs to class $\mathscr{V}$, shares no defined atoms with $\mathscr{P}'$, and program $\mathscr{P}_v$ can be shown to have a non-monotonic semantic operator using the pair of interpretations $I = \langle \emptyset, \{c\} \rangle$ and $I' = \langle \{b\}, \{c\} \rangle$, because $I \subseteq I'$ but $\Phi_{\mathscr{P}_v} I \nsubseteq \Phi_{\mathscr{P}_v} I'$. Note that this same pair of interpretations can be used to show that $\Phi_{\mathscr{P}}$ is non-monotonic.

4.3. The Post Correspondence Problem (PCP)

The *Post correspondence problem* is a well-known undecidable problem. The PCP problem well be introduced next using the notation in [17]. An instance K of the PCP can be defined as a finite sequence $K = (\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)$ of pairs (α_i, β_i) were $\alpha_i' s$ and $\beta_i' s$ are nonempty finite words over $\{0, 1\}$. The question is whether where exists a nonempty sequence $i_1, i_2, \ldots, i_k$ of indices $i_1, \ldots, i_k \in \{1, \ldots, n\}$ such that $\alpha_{i_1} \cdot \alpha_{i_2} \cdot \ldots \cdot \alpha_{i_k} = \beta_{i_1} \cdot \beta_{i_2} \cdot \ldots \cdot \beta_{i_k}$, where $\cdot$ is a binary concatenation operator. It has been established that a PCP instance K has a solution iff the formula G_K is valid in classical two-valued logic as shown in [18]. The formula $G_K = (F_{1K} \wedge F_{2K}) \rightarrow H_K$, such that

$$F_{1K} = \bigwedge_{i=1}^{n} p(f_{\alpha_i}(c), f_{\beta_i}(c)), F_{2K} = (\forall X, Y)(p(X, Y) \rightarrow \bigwedge_{i=1}^{n} p(f_{\alpha_i}(X), f_{\beta_i}(Y))), H_K = (\exists Z) p(Z, Z).$$

The formula G_K will be used later in the form derived below.

$$\begin{aligned}
\neg F_{1K} &\equiv_2 \bigvee_{i=1}^{n} \neg p(f_{\alpha_i}(c), f_{\beta_i}(c)) \\
\neg F_{2K} &\equiv_2 \neg(\forall X, Y)(\neg p(X, Y) \vee \bigwedge_{i=1}^{n} p(f_{\alpha_i}(X), f_{\beta_i}(Y))) \\
&\equiv_2 (\exists X, Y)(p(X, Y) \wedge (\bigvee_{i=1}^{n} \neg p(f_{\alpha_i}(X), f_{\beta_i}(Y)))) \\
&\equiv_2 \bigvee_{i=1}^{n} (\exists X, Y)(p(X, Y) \wedge \neg p(f_{\alpha_i}(X), f_{\beta_i}(Y)))
\end{aligned}$$

$$\begin{aligned}
G_K &= (F_{1K} \wedge F_{2K}) \rightarrow H_K \\
&\equiv_2 \neg F_{1K} \vee \neg F_{2K} \vee H_K \\
&\equiv_2 \bigvee_{i=1}^{n} \neg p(f_{\alpha_i}(c), f_{\beta_i}(c)) \\
&\vee \bigvee_{i=1}^{n} (\exists X, Y)(p(X, Y) \wedge \neg p(f_{\alpha_i}(X), f_{\beta_i}(Y))) \\
&\vee (\exists Z) p(Z, Z)
\end{aligned}$$

We use $\equiv_2$ to denote equivalence in classical two-valued logic. Note that G_K uses a constant symbol c, two unary function symbols f_0 and f_1, a binary predicate symbol p. For simplicity we use the following abbreviation: $f_{\sigma_1 \ldots \sigma_m}(X) := f_{\sigma_m}(\ldots(f_{\sigma_1}(X))\ldots)$, where $\sigma_i \in \{0, 1\}$ for all $1 \leq i \leq m$. Note c denotes ϵ, the empty word, f_0 denotes the function "concatenate 0 to the right", and f_1 denotes the function "concatenate 1 to the right". Thereby $f_{\alpha_i}(X)$ concatenates to the right of word X. For example $f_{01}(00) = f_0(f_1(00)) = f_0(001) = 0010$. We will show that a PCP instance K has a solution iff G_K is valid in classical two-valued logic iff a program $\mathscr{P}_K$ has a monotonic semantic operator. *How does the monotonicity of the semantic operator have anything to do with validity in classical two-valued logic?*

4.4. Monotonicity and Validity in Classical Two-valued Logic

To see the connection between the monotonicity problem and validity in classical two-valued logic, let us have another look at the examples we discussed in this section: $\mathscr{P}_1 = \{a \leftarrow \boldsymbol{c},\ a \leftarrow ctxt\,b\}$, $\mathscr{P}_2 = \{a \leftarrow \top,\ a \leftarrow ctxt\,b\}$, and $\mathscr{P}_3 = \{a \leftarrow \neg\boldsymbol{c},\ a \leftarrow \boldsymbol{c},\ a \leftarrow ctxt\,b\}$. Program $\mathscr{P}_1$ was shown to have a non-monotonic semantic operator, while program $\mathscr{P}_2$ and $\mathscr{P}_3$ were shown to have monotonic semantic operators. All of those programs share common features such as belonging to class $\mathscr{V}$ and containing the clause $a \leftarrow ctxt\,b$. So the difference has to arise from the bodies of the rest of the clauses. Note that the disjunction of the bodies of the rest of the clauses are the following: (1) $\boldsymbol{c}$, (2) $\top$, and (3) $\neg\boldsymbol{c} \vee \boldsymbol{c}$. (1) is not valid in classical two-valued logic, whereas (2), (3) are valid in classical two-valued logic. This is not a coincidence, and it will be formally proved to be the reason why $\Phi_{\mathscr{P}_1}$ is non-monotonic, whereas $\Phi_{\mathscr{P}_2}$ and $\Phi_{\mathscr{P}_3}$ are monotonic.

To that end, one needs to show the link between validity in three-valued Łukasiewicz logic and classical two-valued logic. Before we proceed to do so, a few propositions on the relation between interpretations that are subsumed by each other will be presented next.

4.5. Subsumed Interpretations

The relations between subsumed interpretations will be shown in this section in four statements, each consisting of two parts. In these statements we assume that I_1 and I_2 are interpretations and $I_1 \subseteq I_2$. Note that a formula is non-contextual, when no contextual literal occurs in it. Now we proceed to prove the four statements.

Observation 1. *For a ground atom A(1) If $I_1(A) = \top$, then $I_2(A) = \top$, (2) If $I_1(A) = \bot$, then $I_2(A) = \bot$.*

Proof. Let $I_1 = \langle I_1^\top, I_1^\bot \rangle$ and $I_2 = \langle I_2^\top, I_2^\bot \rangle$. Because $I_1 \subseteq I_2$, $I_1^\top \subseteq I_2^\top$ and $I_1^\bot \subseteq I_2^\bot$. If $I(A) = \top$, then $A \in I_1^\top$, consequently $A \in I_2^\top$. So $I_2(A) = \top$. The second part can be proved similarly. □

Proposition 2. *For a non-contextual literal L (1) If $I_1(L) = \top$, then $I_2(L) = \top$, (2) If $I_1(L) = \bot$, then $I_2(L) = \bot$.*

Proof. If $L = A$, where A is an atom, then the proposition follows by Observation 1. If $L = \neg A$, where A is an atom, then $I_1(L) = I_1(\neg A)$. If $I_1(\neg A) = \top$, then $I_1(A) = \bot$. By Observation 1, $I_2(A) = \bot$. Consequently $I_2(\neg A) = \top$. The second part can be proved in a similar fashion. □

Proposition 3. *For a non-contextual conjunction of literals C (1) If $I_1(C) = \top$, then $I_2(C) = \top$, (2) If $I_1(C) = \bot$, then $I_2(C) = \bot$.*

Proof. If C is the empty conjunction, C denotes $\top$. Consequently $I_1(C) = I_2(C) = \top$, which satisfies the proposition trivially. Otherwise C is a non-empty conjunction of literals, $C = L_1 \wedge \ldots \wedge L_n$, where L_i is a non-contextual literal for all i where $1 \leq i \leq n$. Then we will prove each claim separately. (1) Assume $I_1(C) = \top$. Then $I_1(L_1 \wedge \ldots \wedge L_n) = \top$. So for all i where $1 \leq i \leq n$, $I_1(L_i) = \top$. By Proposition 2, $I_2(L_i) = \top$ for all i where $1 \leq i \leq n$. So $I_2(L_1 \wedge \ldots \wedge L_n) = \top$, and $I_2(C) = \top$. (2) Assume $I_1(C) = \bot$. Then $I_2(L_1 \wedge \ldots \wedge L_n) = \bot$. So for some i where $1 \leq i \leq n$, $I_2(L_i) = \bot$. By Proposition 2, $I_2(L_i) = \bot$. Consequently $I_2(L_1 \wedge \ldots \wedge L_n) = \bot$, and $I_2(C) = \bot$. □

Proposition 4. *For a non-contextual disjunction of conjunctions of literals D (1) If $I_1(D) = \top$, then $I_2(D) = \top$, (2) If $I_1(D) = \bot$, then $I_2(D) = \bot$.*

Proof. If D is the empty disjunction, then D denotes $\bot$. Consequently, $I_1(D) = I_2(D) = \bot$, which satisfies the proposition trivially. Since D is a non-contextual disjunction of conjunctions of literals, $D = C_1 \vee \ldots \vee C_n$, where C_i is a non-contextual conjunction of literals. Then we will prove each claim separately. (1) Assume $I_1(D) = \top$. Then $I_1(C_1 \vee \ldots \vee C_n) = \top$. So for some i where $1 \leq i \leq n$, $I_1(C_i) = \top$. By Proposition 3, $I_2(C_i) = \top$. Consequently $I_2(D) = \top$. (2) Assume $I_2(D) = \bot$. Then $I_1(C_1 \vee \ldots \vee C_n) = \bot$. So for all i where $1 \leq i \leq n$, $I_1(C_i) = \bot$. By Proposition 3, $I_2(C_i) = \bot$ for all i where $1 \leq i \leq n$. Consequently $I_2(C_1 \vee \ldots \vee C_n) = \bot$ and $I(D) = \bot$. □

For example assume we have a disjunction of conjunctions of literals $D = (\neg b \wedge c) \vee d$. Assume we have two interpretations $I_1 = \langle \{d\}, \{c\} \rangle$ and $I_2 = \langle \{d\}, \{b, c\} \rangle$. I_1 maps D to $I_1(D) = (\neg \mathsf{U} \wedge \bot) \vee \top = \top$. Because $I_1 \subseteq I_2$, I_2 must map D to $\top$ too, $I_2(D) = (\neg \bot \wedge \bot) \vee \top = \top$. Note that Proposition 2 does not hold for a contextual literal *ctxt a*. Consider the two interpretations $I_1 = \langle \emptyset, \emptyset \rangle$, $I_2 = \langle \{a\}, \emptyset \rangle$. Although $I_1 \subseteq I_2$, interpretation I_1 maps the contextual literal *ctxt a* to $I_1(ctxt\, a) = \bot$, and interpretation I_2 maps the same contextual literal to $I_2(ctxt\, a) = \top$. The same example can be used to show why Proposition 3 and Proposition 4 do not hold for contextual conjunctions of literals and contextual disjunctions of conjunctions of literals.

4.6. Validity in Classical Two-valued Logic and Three-Valued Łukasiewicz Logic

Here we establish the link between validity in classical two-valued logic and three-valued Łuasiewicz Logic.

Proposition 5. *If there are two interpretations I and J, where I is two-valued and J is three-valued, and that for each ground atom A occurring in a formula F we find $I(A) = J(A)$, then $I(F) = J(F)$.*

Proof. The truth tables of the operators of classical logic and three-valued Łukasiewicz logic are identical when none of the operands are mapped to U, and J maps no atom A to U and I cannot map A to U, hence $I(A) = J(A)$. See Table 1. $\square$

Proposition 6. *If D is a non-contextual disjunction of conjunctions of literals, then there is a two-valued interpretation I where $I(D) = \bot$ iff there is a three-valued interpretation J where $J(D) = \bot$*

Proof. To prove the **if-half**, assume there is a three-valued interpretation J where $J(D) = \bot$. Construct another three-valued interpretation J' where for each ground A: If $J(A) \neq \mathsf{U}$, then $J'(A) = J(A)$, and If $J(A) = \mathsf{U}$, then $J'(A) = \bot$. By the definition of J', $J \subseteq J'$. So $J'(D) = \bot$ by Proposition 4. Construct a two-valued Herbrand interpretation I such that for each ground atom A, $I(A) = J(A)$. I is a two-valued interpretation because J does not map any atom to U by the definition of J'. By Proposition 5, $I(D) = J(D) = \bot$. To prove the **only-if-half**, assume there is a two-valued interpretation I where $I(D) = \bot$. Construct a three-valued interpretation J where for each ground atom A, $J(A) = I(A)$. By Proposition 5, $I(D) = J(D) = \bot$. $\square$

For example assume we have a disjunction of conjunctions of literals $D_1 = a \vee (b \wedge \neg c)$, and a two-valued interpretation $I = \{a\}$, i.e., the interpretation I maps the atom a to $\top$ and the rest of the atoms, b and c, to $\bot$. The interpretation I maps the disjunction D_1 to $I(D_1) = \top \vee (\bot \wedge \top) = \top$. By Proposition 6 this entails the existence of a three-valued interpretation J such that $J(D_1) = \top$. For instance the three-valued interpretation $J = \langle \{a\}, \emptyset \rangle$ maps the disjunction D_1 to $J(D_1) = \top \vee (\mathsf{U} \wedge \mathsf{U}) = \top$. On the other hand for a disjunction of conjunctions of literals such as $D_3 = a \vee \neg a$, D_3 is clearly valid in classical two-valued logic, so by the previous proposition we should be able to find a three-valued interpretation J such that $J(D_3) = \bot$, which is the case because for all three-valued interpretations, D_3 can either be mapped to $\top$ or U.

Corollary 1. *A non-contextual disjunction of conjunctions of literals D is valid in classical two-valued logic iff D cannot be falsified in three-valued Łukasiewicz logic.*

Proof. This can be shown by negating both sides of the equivalence of Proposition 6. □

4.7. Herbrand's Theorem

Here the weak version of Herbrand's theorem is introduced as it is needed for the undecidability proof. In this subsection we only speak about validity in classical two-valued logic.

Theorem 2. *[19] Let $\mathscr{T}$ be a finitely axiomatized by universal formulas. Let $F = (\exists X_1, \dots, X_n)G(X_1, \dots, X_n)$ be a purely existential formula. Then $\mathscr{T} \models F$ iff there are gound terms $(t_{i,j})_{1 \leq i \leq p,\, 1 \leq j \leq n}$ such that $\mathscr{T} \models \bigvee_{i=1}^{p} G(t_{i,1}, \dots, t_{i,n})$.*

Corollary 2. *Let $F = (\exists X_1, \dots, X_n)G(X_1, \dots, X_n)$ be a purely existential sentence and G is a quantifier-free formula. Then F is valid iff there are ground terms $(t_{i,j})_{1 \leq i \leq p,\, 1 \leq j \leq n}$ such that $\bigvee_{i=1}^{p} G(t_{i,1}, \dots, t_{i,n})$ is valid.*

Proof. A special case of Theorem 2 where $\mathscr{T} = \emptyset$ and F is a sentence. □

Remember that $g(F)$ denote the set of all ground instances of a formula F.

Corollary 3. *Let $F = (\exists X_1, \dots, X_n)G(X_1, \dots, X_n)$ be a purely existential sentence and G a quantifier-free formula. Then F is valid iff $\bigvee_{H \in g(G)} H$ is valid.*

Proof. To prove the **if-half**, assume F is valid, then there are ground terms $(t_{i,j})_{1 \leq i \leq p, 1 \leq j \leq n}$ such that $\bigvee_{i=1}^{p} G(t_{i,1}, \dots, t_{i,n})$ is valid by Corollary 2. For each $i \leq p$, $G(t_{i,1}, \dots, t_{i,n})$ is a ground instance of G, i.e., $G(t_{i,1}, \dots, t_{i,n}) \in g(G)$. Hence $\bigvee_{H \in g(G)} H \equiv_2 H' \vee \bigvee_{i=1}^{p} G(t_{i,1}, \dots, t_{i,n})$, where H' is a disjunction of all the ground instances of G that do not occur in the disjunction $\bigvee_{i=1}^{p} G(t_{i,1}, \dots, t_{i,n})$. Consequently, $\bigvee_{H \in g(G)} H$ is valid too. To prove the **only-if-half**, assume $\bigvee_{H \in g(G)} H$ is valid. For each $H \in g(G)$, $H = G(t_1, \dots, t_n)$ for some sequence of ground terms $t_1, \dots, t_n$. We can give each $H \in g(G)$ a unique natural number say i such that $H = G(t_{i,1}, \dots, t_{i,n})$. Consequently, there are ground terms $(t_{i,j})_{1 \leq i \leq p, 1 \leq j \leq n}$, where p is the number of ground instances of G such that $\bigvee_{i=1}^{p} G(t_{i,1}, \dots, t_{i,n})$ is valid. By Corollary 2, F is valid. □

For example the following formula $(\exists X, Y)p(X, Y)$ is valid iff $\bigvee_{H \in g(p(X,Y))} H$ is valid by the corollary above. Later we will deal with a formula that is a disjunction of purely existential sentences, hence we need the following proposition.

Proposition 7. *Let $F_i = (\exists X_{i,1}, \dots, X_{i,m_i})G_i(X_{i,1}, \dots, X_{i,m_i})$ be a purely existential sentence and G_i a quantifier-free formula for any $i \in \mathbb{N}$. Then $F_1 \vee \dots \vee F_n$ is valid iff $(\bigvee_{H_1 \in g(G_1)} H_1) \vee \dots \vee (\bigvee_{H_n \in g(G_n)} H_n)$ is valid.*

Proof. Note that

$$\begin{aligned} F_1 \vee \dots \vee F_n &= ((\exists X_{1,1}, \dots, X_{1,m_1})G_1 \vee \dots \vee (\exists X_{n,1}, \dots, X_{n,m_n})G_n) \\ &\equiv_2 (\exists X_{1,1}, \dots X_{1,m_1}, \dots, X_{n,1}, \dots, X_{n,m_n})(G_1 \vee \dots \vee G_n) \end{aligned}$$

By Corollary 3, $F_1 \vee \dots \vee F_n$ is valid iff $\bigvee_{H \in g(G_1 \vee \dots \vee G_n)} H$ is valid. It is not hard to see that $\bigvee_{H \in g(G_1 \vee \dots \vee G_n)} H \equiv_2 (\bigvee_{H_1 \in g(G_1)} H_1) \vee \dots \vee (\bigvee_{H_n \in g(G_n)} H_n)$ because G_1, ..., and G_n share no variables pairwise. □

For example the following formula $(\exists X, Y)(p(X,Y))) \vee (\exists Z)(q(Z))$ is valid iff $\bigvee_{H_1 \in g(p(X,Y))H_1} \vee \bigvee_{H_2 \in g(q(Z))} H_2$, by Proposition 7.

4.8. Validity And The Monotonicity in Class $\mathscr{V}$

Here the connection between the previous validity results and the monotonicity of programs in class $\mathscr{V}$ will be shown.

Proposition 8. *For a program $\mathscr{P}$ of class $\mathscr{V}$, an interpretation I, a ground atom A*

$$\Phi_{\mathscr{P}} I(A) = \begin{cases} I(ctxt\, q \vee (\bigvee_{1\leq i\leq n} \bigvee_{C \in g(body_i)} C)) & A = r \\ \mathsf{U} & otherwise \end{cases}$$

Proof. $\Phi_{\mathscr{P}} I(r) = I(D)$, for a disjunction of conjunctions of literals D where $r \leftrightarrow D \in wc(g(\mathscr{P}))$. Note that $g(\mathscr{P}) = \{r \leftarrow ctxt q\} \cup \bigcup_{1\leq i\leq n}\{r \leftarrow C | C \in g(body_i)\}$. Hence $D = ctxt q \vee (\bigcup_{1\leq i\leq n} \bigcup_{C \in g(body_i)} C)$. Consequently $\Phi_{\mathscr{P}} I(r) = I(ctxt\, q \vee (\bigvee_{1\leq i\leq n} \bigvee_{C \in g(body_i)} C))$. □

Proposition 9. *For any program $\mathscr{P}$ of class $\mathscr{V}$, the semantic operator $\Phi_{\mathscr{P}}$ is non-monotonic iff $\bigvee_{1\leq i\leq n}(\exists body_i)$ is not valid in classical two-valued logic.*

Proof. To prove the **if-half**, assume the sentence $\bigvee_{1\leq i\leq n}(\exists body_i)$ is not valid in classical two-valued logic. By Proposition 7, the formula $D = \bigvee_{1\leq i\leq n} \bigvee_{C \in g(body_i) C}$ is not valid in classical two-valued logic either. Hence, we can find a two-valued interpretation I such that $I(D) = \bot$. By Corollary 1, we can find a three-valued Łukasiewicz interpretation I' where $I'(D) = \bot$. So one can construct an interpretation I_1 where: (1) For any ground atom A, if $A \neq q$, then $I_1(A) = I'(A)$; (2) $I_1(q) = \mathsf{U}$. Note that $I_1(D) = \bot$, since $I(D) = \bot$ and q does not occur in D. Also $I_1(ctxt q) = \bot$ by definition of I_1. Therefore $I_1(ctxt q \vee D) = \bot$. Construct another interpretation I_2 where: (1) For any ground atom A, if $A \neq q$, then $I_2(A) = I'(A)$; (2) $I_2(q) = \top$. Note that $I_2(D) = \bot$, since $I'(D) = \bot$ and q does not occur in D. Also $I_2(ctxt q) = \top$ by definition of I_2. Therefore $I_2(ctxt q \vee D) = \top$. Note also that $I_1 \subseteq I_2$ because they only differ on the mapping of q where $I_1(q) = \mathsf{U}$ and $I_2(q) = \top$. By Proposition 7 $\Phi_{\mathscr{P}} I_1(r) = I_1(ctxt q \vee D) = \bot$ and $\Phi_{\mathscr{P}} I_2(r) = I_2(ctxt q \vee D) = \top$. Hence $\Phi_{\mathscr{P}} I_1 \nsubseteq \Phi_{\mathscr{P}} I_2$, while $I_1 \subseteq I_2$. So $\Phi_{\mathscr{P}}$ is non-monotonic. To prove the **only-if-half**, assume $\Phi_{\mathscr{P}}$ is non-monotonic. Then There are two interpretations I_1 and I_2 where $I_1 \subseteq I_2$ and $\Phi_{\mathscr{P}} I_1 \nsubseteq \Phi_{\mathscr{P}} I_2$. Due to $\Phi_{\mathscr{P}} I_1 \nsubseteq \Phi_{\mathscr{P}} I_2$, for some ground atom A, $\Phi_{\mathscr{P}} I_1(A) \neq \Phi_{\mathscr{P}} I_2(A)$ and $\Phi_{\mathscr{P}} I_1(A) \neq \mathsf{U}$. $\Phi_{\mathscr{P}} I_1(A) \neq \mathsf{U}$ implies that A is a defined atom otherwise $\Phi_{\mathscr{P}} I_1(A) = \Phi_{\mathscr{P}} I_2(A) = \mathsf{U}$ by Definition 2. Since r is the only defined atom, $\Phi_{\mathscr{P}} I_1(r) \neq \Phi_{\mathscr{P}} I_2(r)$ and $\Phi_{\mathscr{P}} I_1(r) \neq \mathsf{U}$. By Proposition 3, $I_1(ctxt q \vee D) \neq I_2(ctxt q \vee D)$ and $I_1(ctxt q \vee D) \neq \mathsf{U}$ where $D = \bigvee_{1\leq i\leq n} \bigvee_{C \in g(body_i)} C$. Since $I_1(ctxt q \vee D) \neq \mathsf{U}$, $I_1(ctxt q \vee D)$ is either $\top$ or $\bot$. Assume $I_1(ctxt q \vee D) = \top$. Assume $I_1(ctxt q) = \top$. Then $I_1(q) = \top$. Since $I_1 \subseteq I_2$, $I_2(q) = \top$ and $I_2(ctxt q) = \top$ so $I_2(ctxt q \vee D) = \top$ but $I_1(ctxt q \vee D) \neq I_2(ctxt q \vee D)$. A contradiction. So $I_1(ctxt q) \neq \top$. Consequently $I_1(D) = \top$, which implies $I_2(D) = \top$, by Proposition 4. So $I_2(ctxt q \vee D) = \top$ but $I_1(ctxt q \vee D) \neq I_2(ctxt q \vee D)$. A contradiction. $I_1(D) \neq \top$ either. Hence $I_1(ctxt q \vee D) \neq \top$, subsequently $I_1(ctxt q \vee D) = \bot$. Since $I_1(ctxt q \vee D) = \bot$, $I_1(D) = \bot$. There is a three-valued Łukasiewicz interpretation I_1 where $I_1(D) = \bot$. By Corollary 1, D is not valid in classical two-valued logic. By Proposition 7, the sentence $\bigvee_{1\leq i\leq n}(\exists body_i)$ is not valid in classical two-valued logic either. □

$$G_K \equiv_2 \bigvee_{i=1}^{n} \neg p(f_{\alpha_i}(c), f_{\beta_i}(c)) \\ \vee \bigvee_{i=1}^{n} (\exists X, Y)(p(X,Y) \wedge \neg p(f_{\alpha_i}(X), f_{\beta_i}(Y))) \\ \vee (\exists Z) p(Z,Z).$$

$$\begin{aligned} \mathscr{P}_K = \{ & r \leftarrow \neg p(f_{\alpha_1}c, f_{\beta_1}c), \\ & \vdots \\ & r \leftarrow \neg p(f_{\alpha_n}c, f_{\beta_n}c), \\ & r \leftarrow p(X,Y) \wedge \neg p(f_{\alpha_1}X, f_{\beta_1}Y), \\ & \vdots \\ & r \leftarrow p(X,Y) \wedge \neg p(f_{\alpha_n}X, f_{\beta_n}Y), \\ & r \leftarrow p(Z,Z), \\ & r \leftarrow ctxt\, q \qquad \}. \end{aligned}$$

Figure 1: For a PCP instance K, the formula G_K is shown on the left, and program $\mathscr{P}_K$ is shown on the right.

4.9. Reduction

As mentioned before a PCP instance K has solution iff G_K is valid, see Section 4.3. Note that program $\mathscr{P}_K$ shown on the right has a non-monotonic semantic operator iff G_K is not valid by Proposition 9. So program $\mathscr{P}_K$ has a monotonic semantic operator iff G_K is valid. Consequently, program $\mathscr{P}_K$ has a monotonic semantic operator iff PCP instance K has a solution.

Theorem 3. *The monotonicity of the semantic operator of a given contextual program $\mathscr{P}$ is undecidable*

Proof. If the monotonicity of $\Phi_{\mathscr{P}}$ is decidable, then the PCP problem is decidable by the previous reduction. Hence, the monotonicity of $\Phi_{\mathscr{P}}$ is undecidable. □

5. Conclusion

Contextual programs use the context operator *(ctxt)* to model cases where default reasoning is preferred. There is always a least fixed point of the semantic operator of non-contextual programs. This is possible because the semantic operator of non-contextual programs is monotonic [12, 10]. On the other hand the semantic operator of contextual programs is often non-monotonic and their semantic operators often do not have any fixed points. Nonetheless, for the class of acyclic contextual programs, there is always a least fixed point of the semantic operator by the Banach Contraction Mapping Theorem [11, 13]. This fixed point was shown to be a model in [13], and has been shown to be a minimal model by Theorem 1. Contextual programs often do not have monotonic semantic operators, that is why there is no guarantee a fixed point exists. The monotonicity of the semantic operator of a contextual given program has been shown to be generally undecidable in Theorem 3 using a reduction from PCP. However, it is decidable for programs with finite Herbrand base because the number of interpretations is finite in this case.

References

[1] E.-A. Dietz, S. Hölldobler, C. Wernhard, Modeling the suppression task under weak completion and well-founded semantics, Journal of Applied Non-Classical Logics 24 (2014) 61–85. doi:10.1080/11663081.2014.911520.

[2] E.-A. Dietz, S. Hölldobler, M. Ragni, A computational logic approach to the abstract and the social case of the selection task, in: Proceedings Eleventh International Symposium on Logical Formalizations of Commonsense Reasoning, 2013.

[3] A. O. da Costa, E.-A. D. Saldanha, S. Hölldobler, M. Ragni, A computational logic approach to human syllogistic reasoning., in: CogSci, 2017.

[4] S. Hölldobler, Ethical decision making under the weak completion semantics, in: Bridging@ IJCAI/ECAI, 2018.

[5] M. Cramer, S. Hölldobler, M. Ragni, Modeling human reasoning about conditionals, in: 19 th International Workshop on Non-Monotonic Reasoning, 2021, p. 223.

[6] E.-A. Dietz, S. Hölldobler, L. M. Pereira, On conditionals., in: GCAI, volume 36, 2015, pp. 79–92. doi:10.29007/7p4b.

[7] E.-A. D. Saldanha, S. Hölldobler, I. L. Rocha, Obligation versus factual conditionals under the weak completion semantics., in: YSIP, 2017, pp. 55–64.

[8] I. Hamada, S. Hölldobler, On disjunctions and the weak completion semantics, Proceedings of the Virtual MathPsych/ICCM. via mathpsych. org/presentation/571 (2021).

[9] E.-A. D. Saldanha, S. Hölldobler, C. D. P. K. Ramli, L. P. Medinacelli, A core method for the weak completion semantics with skeptical abduction, Journal of Artificial Intelligence Research 63 (2018) 51–86. doi:10.1613/jair.1.11236.

[10] K. Stenning, M. Van Lambalgen, Human reasoning and cognitive science, MIT Press, 2012.

[11] E.-A. Dietz Saldanha, S. Hölldobler, L. M. Pereira, Contextual reasoning: Usually birds can abductively fly, in: International Conference on Logic Programming and Nonmonotonic Reasoning, Springer, 2017, pp. 64–77. doi:10.1007/978-3-319-61660-58.

[12] S. Hölldobler, C. D. P. K. Ramli, Logic programs under three-valued łukasiewicz semantics, in: International Conference on Logic Programming, Springer, 2009, pp. 464–478. doi:10.1007/978-3-642-02846-5_37.

[13] S. Hölldobler, C. K. Ramli, Contraction properties of a semantic operator for human reasoning, in: Proceedings of the fifth international conference on information, volume 228, 2009, p. 231.

[14] D. Van Dalen, D. van Dalen, Logic and structure, volume 3, Springer, 1994.

[15] J. Łukasiewicz, L. Borkowski, Jan Łukasiewicz, North-Holland Publishing Company, 1970.

[16] K. L. Clark, Negation as failure, in: Logic and data bases, Springer, 1978, pp. 293–322. doi:10.1007/978-1-4684-3384-5_11.

[17] S. Hölldobler, Logik und Logikprogrammierung, volume 1: Grundlagen, Heidelberg: Synchron Publishers GmbH, 2001.

[18] U. Schöning, Logic for computer scientists, Birkhäuser Boston, 2008, pp. 64–66.

[19] A. Alcolei, P. Clairambault, M. Hyland, G. Winskel, The true concurrency of herbrand's theorem, in: 27th EACSL Annual Conference on Computer Science Logic (CSL 2018), Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CSL.2018.5.