Logics and Networks for Human Reasoning

Steffen Hölldobler and Carroline Dewi Puspa Kencana Ramli

International Center for Computational Logic,
TU Dresden, 01062 Dresden, Germany
`sh@iccl.tu-dresden.de`
`http://www.computational-logic.org/~sh/`

Abstract We propose to model human reasoning tasks using completed logic programs interpreted under the three-valued Łukasiewicz semantics. Given an appropriate immediate consequence operator, completed logic programs admit a least model, which can be computed by iterating the consequence operator. Reasoning is then performed with respect to the least model. The approach is realized in a connectionist setting.

Key words: Human Reasoning, Logic Programs, Connectionist Models

1 Introduction

It has been widely argued in the field of Cognitive Science that logic is inadequate for modelling human reasoning (see e.g. [3]). In this context, "logic" is meant to be classical logic and, indeed, classical logic fails to capture some well-documented forms of human reasoning. However, in the field of Artificial Intelligence many non-classical logics have been studied and widely used. These logics try to capture many assumptions or features that occur in commonsense reasoning like, for example, the closed world assumption or non-monotonicity.

Recently, in [17] Stenning and van Lambalgen have suggested that completed logic programs under the three-valued Fitting semantics [7] can adequately model many human reasoning tasks. In addition, they propose a connectionist realization of their approach.

While trying to understand Stenning and van Lambalgen's approach we made the following observations: *(i)* Łukasiewicz semantics [15] seems to be better suited for the approach as the law of equivalence holds under this semantics, whereas it does not hold under Fitting semantics. *(ii)* [17] contains some (minor) errors. *(iii)* The immediate consequence operator introduced in [17] differs in a subtle way from the one in [7] and it turned out, that the latter is inadequate for human reasoning. *(iv)* The core method, a connectionist model generator for logic programs first presented in [10], can easily be adapted to handle Stenning and van Lambalgen's approach.

The paper discusses these observation by presenting three-valued logics, logic programs, their completion semantics as well as their immediate consequence operators in Section 2, by specifying an algorithm for mapping Stenning and van Lambalgen's immediate consequence operator onto a recurrent neural network with feed-forward core in Section 3, by showing how some human reasoning task can be adequately modelled in the proposed logic and its connectionist realization in Section 4, and by discussing our findings in Section 5.

2 Logics, Programs and Consequence Operators

2.1 Three-Valued Logics

In this paper we consider (propositional logic) languages over an alphabet consisting of (propositional) variables, the connectives $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ and paranthesis. We will consider various three-valued logics based on the semantics of their connectives (see Table 1): Łukasiewicz has proposed $\{\neg, \wedge, \vee, \rightarrow_{Ł}, \leftrightarrow_{Ł}\}$ [15], Kleene uses $\{\neg, \wedge, \vee, \rightarrow_K, \leftrightarrow_K\}$ in his *strong three-valued logic* [14] and $\{\neg, \wedge, \vee, \rightarrow_K, \leftrightarrow_c\}$ is suggested by Fitting for logic programming [7] and used by Stenning and van Lambalgen to model human reasoning [17].

An *interpretation* is a mapping from the language to the set of truth values $\{\top, \bot, u\}$. Using Table 1 an interpretation for a given formula is uniquely determined by specifying the values for the propositional variables occurring in it. We will represent interpretations by pairs $\langle I^\top, I^\bot\rangle$, where the set $I^\top$ contains all variables which are mapped to $\top$, the set $I^\bot$ contains all variables mapped to $\bot$, and all variables which occur neither in $I^\top$ nor in $I^\bot$ are mapped to u. We use I_K, I_F and $I_Ł$ to denote that an interpretation I uses Kleene, Fitting or Łukasiewicz semantics, respectively. Furthermore, let $\mathcal{I}$ denote the set of all interpretation. $(\mathcal{I}, \subseteq)$ is a complete semilattice (see [7]). Finally, an interpretation I is said to be a *model* for a formula G iff $I(G) = \top$.

One should observe, that the law of equivalence (for all interpretations I: $I(F \leftrightarrow G) = I((F \rightarrow G) \wedge (G \rightarrow F))$) holds under Łukasiewicz and Kleene semantics, but not under Fitting semantics.

2.2 Programs

A *(program) clause* is an expression of the form $A \leftarrow B_1 \wedge \ldots \wedge B_n$, $n \geq 1$, where A is an atom and each B_i, $1 \leq i \leq n$, is either a literal (i.e., atom or negated

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

		$\wedge$	$\vee$	$\rightarrow_K$	$\rightarrow_Ł$	$\leftrightarrow_K$	$\leftrightarrow_Ł$	$\leftrightarrow_c$
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$
$\bot$	$\top$	$\bot$	$\top$	$\top$	$\top$	$\bot$	$\bot$	$\bot$
u	$\top$	u	$\top$	$\top$	$\top$	u	u	$\bot$
$\top$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\top$	$\top$	$\top$	$\top$
u	$\bot$	$\bot$	u	u	u	u	u	$\bot$
$\top$	u	u	$\top$	u	u	u	u	$\bot$
$\bot$	u	$\bot$	u	$\top$	$\top$	u	u	$\bot$
u	u	u	u	u	$\top$	u	$\top$	$\top$

Table 1. Truth tables for 3-valued logics.

atom), $\top$ or $\bot$. A is called *head* and $B_1 \wedge \ldots \wedge B_n$ *body* of the program clause. $\top$ is a valid formula, whereas $\bot$ is an unsatisfiable one. One should observe that the body of each clause is non-empty. A clause of the form $A \leftarrow \top$ is called *positive fact*. A clause of the form $A \leftarrow \bot$ is called *negative fact*. A *(logic) program* is a finite set of clauses. Two examples are $\mathcal{P}_1 = \{p \leftarrow q\}$ and $\mathcal{P}_2 = \{p \leftarrow q, q \leftarrow \bot\}$.

2.3 Completion

Let $\mathcal{P}$ be a program. It is turned into a single formula in the following way:

1. All clauses with the same head $A \leftarrow \mathit{Body}_1, \ldots, A \leftarrow \mathit{Body}_n$ are replaced by the single formula $A \leftarrow \mathit{Body}_1 \vee \ldots \vee \mathit{Body}_n$.
2. The resulting set is replaced by its conjunction.
3. If A is a variable occurring in $\mathcal{P}$ with no clause in $\mathcal{P}$ of the form $A \leftarrow \mathit{Body}$, then add $A \leftarrow \bot$.
4. All occurrences of $\leftarrow$ are replaced by $\leftrightarrow$.

The resulting formula is called *completion of* $\mathcal{P}$ and is denoted by $\mathit{comp}(\mathcal{P})$. If the third step is omitted, then the resulting formula is called *weak completion of* $\mathcal{P}$ and is denoted by $\mathit{wcomp}(\mathcal{P})$. For example, $\mathit{wcomp}(\mathcal{P}_1) = (p \leftrightarrow q) \neq \mathit{comp}(\mathcal{P}_1) = (p \leftrightarrow q) \wedge (q \leftrightarrow \bot) = \mathit{comp}(\mathcal{P}_2) = \mathit{wcomp}(\mathcal{P}_2)$.

Let $I = \langle \{p, q\}, \emptyset \rangle$. $I_K(\mathcal{P}_2) = I_F(\mathcal{P}_2) = I_{\text{L}}(\mathcal{P}_2) = \top$, whereas $I_K(\mathit{comp}(\mathcal{P}_2)) = I_F(\mathit{comp}(\mathcal{P}_2)) = I_{\text{L}}(\mathit{comp}(\mathcal{P}_2)) = \bot$. The completion forces all models to map q to $\bot$, and the same holds for the weak completion.

Let $I = \langle \emptyset, \emptyset \rangle$. $I_K(p \leftrightarrow p) = u$, whereas $I_F(p \leftrightarrow p) = \top$. This has led Fitting to consider $\leftrightarrow_c$ in [7]. One should observe that $I_K(p \leftarrow p) = I_F(p \leftarrow p) = u$. In other words, whereas I is a model for the completed program $p \leftrightarrow p$, it is not a model for $p \leftarrow p$ under the Fitting semantics. On the other hand, $I_{\text{L}}(p \leftrightarrow p) = I_{\text{L}}(p \leftarrow p) = \top$.

2.4 Consequence Operators

In [7] Fitting has defined an immediate consequence operator $\Phi_{F,\mathcal{P}}(I) = \langle J^\top, J^\bot \rangle$, where $J^\top = \{A \mid A \leftarrow \mathit{Body} \in \mathcal{P} \text{ and } I(\mathit{Body}) = \top\}$ and $J^\bot = \{A \mid \text{for all } A \leftarrow \mathit{Body} \in \mathcal{P} : \ I(\mathit{Body}) = \bot\}$. One should note that $I_K(\mathit{Body}) = I_F(\mathit{Body}) = I_{\text{L}}(\mathit{Body})$ because the body of a clause is a conjunction of literals. Fitting has shown that $\Phi_{F,\mathcal{P}}$ is monotone on $(\mathcal{I}, \subseteq)$.

Proposition 1. *$\Phi_{F,\mathcal{P}}$ is continuous and admits a least fixed point $\mathit{lfp}(\Phi_{F,\mathcal{P}})$.*

Proof Because our programs are finite and, thus, the underlying alphabet and all directed subsets of $\mathcal{I}$ are finite, we find that a monotone $\Phi_{F,\mathcal{P}}$ is also continuous (see e.g. [18]). Hence, $\Phi_{F,\mathcal{P}}$ admits a least fixed point, which can be computed by iterating $\Phi_{F,\mathcal{P}}$ starting with the empty interpretation. $\square$

$\mathit{lfp}(\Phi_{F,\mathcal{P}})$ is equal to the least model of $\mathit{comp}(\mathcal{P})$ under Fitting semantics. For example, $\mathit{lfp}(\Phi_{F,\mathcal{P}_1}) = \mathit{lfp}(\Phi_{F,\mathcal{P}_2}) = \langle \emptyset, \{p, q\} \rangle$.

In [17] Stenning and van Lambalgen have defined a slightly different immediate consequence operator: $\Phi_{SvL,\mathcal{P}}(I) = \langle J^\top, J^\perp \rangle$, where $J^\top = \{A \mid A \leftarrow \mathit{Body} \in \mathcal{P}$ and $I(\mathit{Body}) = \top\}$ and $J^\perp = \{A \mid$ there exists $A \leftarrow \mathit{Body} \in \mathcal{P}$ and for all $A \leftarrow \mathit{Body} \in \mathcal{P} : \ I(\mathit{Body}) = \perp\}$. They showed that this operator is also monotone.

Proposition 2. *$\Phi_{SvL,\mathcal{P}}$ is continuous and admits a least fixed point $\mathit{lfp}(\Phi_{SvL,\mathcal{P}})$.*

This result can be proven along the lines of the proof of Propostion 1. For example, $\mathit{lfp}(\Phi_{SvL,\mathcal{P}_1}) = \langle \emptyset, \emptyset \rangle$ and $\mathit{lfp}(\Phi_{SvL,\mathcal{P}_2}) = \langle \emptyset, \{p, q\} \rangle$. In addition, Stenning and van Lambalgen make the following claims, where they assume that models are defined with respect to Fitting semantics:

A. The least fixed point of $\Phi_{SvL,\mathcal{P}}$ can be shown to be the minimal model of $\mathcal{P}$ (Lemma 4(1.) in [17]).
B. All models of $\mathit{comp}(\mathcal{P})$ are fixed points of $\Phi_{SvL,\mathcal{P}}$ (Lemma 4(3.) in [17]).

Both claims are false. Consider $\mathcal{P}_1 = \{p \leftarrow q\}$ and let $I = \langle \emptyset, \emptyset \rangle$. As discussed before, $\mathit{lfp}(\Phi_{SvL,\mathcal{P}_1}) = I$, but $I_F(\mathcal{P}_1) = u$; thus, we have obtained a counter example for A. One should observe that the minimal models of $\mathcal{P}_1$ under Fitting semantics are $\langle \{p\}, \emptyset \rangle$ and $\langle \emptyset, \{q\} \rangle$, both of which are not fixed points of $\Phi_{SvL,\mathcal{P}_1}$. Now let $I'' = \langle \emptyset, \{p, q\} \rangle$ and $I' = \langle \emptyset, \{p\} \rangle$. $I''_F(\mathit{comp}(\mathcal{P}_1)) = \top$, but $\Phi_{SvL,\mathcal{P}_1}(I'') = I'$, $\Phi_{Svl,\mathcal{P}_1}(I') = I$, and $\Phi_{SvL,\mathcal{P}_1}(I) = I$; thus, we have obtained a counter example for B.

Problem A. can be overcome if we interprete programs under Łukasiewicz semantics. For the discussed example we find $I_{\text{Ł}}(\mathcal{P}_1) = \top$, which is no coincidence as we will show in the sequel.

Proposition 3. *(i) If $I_L(\text{wcomp}(\mathcal{P})) = \top$ then $\Phi_{SvL,\mathcal{P}}(I) \subseteq I$.*
(ii) If $\Phi_{SvL,\mathcal{P}}(I) = I$ then $I_L(\text{wcomp}(\mathcal{P})) = \top$.

Proof Sketch *(i)* If $I_{\text{Ł}}(\mathit{wcomp}(\mathcal{P})) = \top$ then for for each equivalence $A \leftrightarrow \mathit{Body}_1 \vee \ldots \vee \mathit{Body}_n$ occurring in $\mathit{wcomp}(\mathcal{P})$ we find that

$$I_{\text{Ł}}(A) = I_{\text{Ł}}(\mathit{Body}_1 \vee \ldots \vee \mathit{Body}_n). \tag{1}$$

Now let $I = \langle I^\top, I^\perp \rangle$ and $\langle J^\top, J^\perp \rangle = \Phi_{SvL,\mathcal{P}}(\langle I^\top, I^\perp \rangle)$. By definition of $\Phi_{SvL,\mathcal{P}}$ we find $J^\top \subseteq I^\top$ and $J^\perp \subseteq I^\perp$ given (1). Hence, $\Phi_{SvL,\mathcal{P}}(I) \subseteq I$.

(ii) Suppose $\Phi_{SvL,\mathcal{P}}(I) = I$. Let $F := A \leftrightarrow \mathit{Body}_1 \vee \ldots \vee \mathit{Body}_n$ be an arbitrary but fixed conjunct occurring in $\mathit{wcomp}(\mathcal{P})$. If $I_{\text{Ł}}(A) = \top$, then there exists $A \leftarrow \mathit{Body}_i \in \mathcal{P}$, such that $I_{\text{Ł}}(\mathit{Body}_i) = \top$. Hence, $I_{\text{Ł}}(\mathit{Body}_1 \vee \ldots \vee \mathit{Body}_n) = \top$ and, consequently, $I_{\text{Ł}}(F) = \top$. The cases $I_{\text{Ł}}(A) = \perp$ and $I_{\text{Ł}}(A) = u$ follow similarly. Because F was arbitrary but fixed we conclude $I_{\text{Ł}}(\mathit{wcomp}(\mathcal{P})) = \top$. □

Proposition 4. *If $I = \mathit{lfp}(\Phi_{SvL,\mathcal{P}})$ then $I_L(\text{wcomp}(\mathcal{P})) = I_L(\mathcal{P}) = \top$.*

Proof Sketch From Proposition 3*(ii)* we learn that $I = \mathit{lfp}(\Phi_{SvL,\mathcal{P}})$ entails $I_{\text{Ł}}(\mathit{wcomp}(\mathcal{P})) = \top$. Moreover, $I_{\text{Ł}}(\mathit{wcomp}(\mathcal{P})) = \top$ iff for each equivalence $A \leftrightarrow \mathit{Body}_1 \vee \ldots \vee \mathit{Body}_n$ occurring in $\mathit{wcomp}(\mathcal{P})$ we find that $I_{\text{Ł}}(A) = I_{\text{Ł}}(\mathit{Body}_1 \vee \ldots \vee \mathit{Body}_n)$. A careful case analysis reveals that $I_{\text{Ł}}(\mathcal{P}) = \top$ holds. □

3 The Core Method

In [10] a connectionist model generator for propositional logic programs using recurrent networks with feed-forward core was presented. It was later called the *core method* [2]. The core method has been extended and applied to a variety of programs including modal (see e.g. [5]) and first-order logic programs [1]. It is based on the idea that feed-forward connectionist networks can approximate almost all functions arbitrarily well [12,9] and, hence, they can also approximate – and in some cases compute – the immediate consequence operators associated with logic programs. Moreover, if such an operator is a contraction mapping on a complete metric space, then Banach's contraction mapping theorem ensures that a unique fixpoint exists such that the sequence constructed from applying the operator iteratively to any element of the metric space converges to the fixed point [8]. Turning the feed-forward core into a recurrent network allows to compute or approximate the least model of a logic program [11].

Kalinke has applied the core method to logic programs under the Fitting semantics presented in Section 2 [13]. In particular, her feed-forward cores compute $\Phi_{F,\mathcal{P}}$ for any given program $\mathcal{P}$. Seda and Lane showed that the core method can be extended to many-valued logic programs [16]. Restricted to three-valued logic programs considered here, their cores also compute $\Phi_{F,\mathcal{P}}$. In the sequel, these approaches are modified in order to compute $\Phi_{SvL,\mathcal{P}}$.

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

1. The input and output layer is a vector of binary threshold units of length $2m$ representing interpretations. The $2i-1$-st unit in the layers, denoted by $i^\top$, is active iff the i-th variable is mapped to $\top$. The $2i$-th unit in the layers, denoted by $i^\perp$, is active iff the i-th variable is mapped to $\perp$. Both, the $2i-1$-st and the $2i$-th unit, are passive iff the i-th variable is mapped to u. The case where both, the $2i-1$-st and the $2i$-th unit, are active is not allowed.
   The threshold of each unit occurring in the input layer is set to $\frac{\omega}{2}$. The threshold of each $2i-1$-st unit occurring in the output layer is set to $\frac{\omega}{2}$. The threshold of each $2i$-th unit occurring in the output layer is set to $max\{\frac{\omega}{2}, l - \frac{\omega}{2}\}$, where l is the number of clauses with head i in $\mathcal{P}$.
   In addition, two units representing $\top$ and $\perp$ are added to the input layer. The threshold of these units is set to $-\frac{\omega}{2}$.
2. For each clause of the form $A \leftarrow B_1 \wedge \ldots \wedge B_k$ occurring in $\mathcal{P}$, do the following.
   (a) Add two binary threshold units $h^\top$ and $h^\perp$ to the hidden layer.
   (b) Connect $h^\top$ to the unit $A^\top$ in the output layer. Connect $h^\perp$ to the unit $A^\perp$ in the output layer.
   (c) For each B_j, $1 \leq j \leq k$, do the following.
      i. If B_j is an atom, then connect the units $B_j^\top$ and $B_j^\perp$ in the input layer to $h^\top$ and $h^\perp$, respectively.

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

(d) Set the threshold of $h^\top$ to $l - \frac{\omega}{2}$, where l is the number of clauses with head A in $\mathcal{P}$. Set the threshold of $h^\perp$ to $\frac{\omega}{2}$.

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

Proposition 5. *For each program $\mathcal{P}$, there exists a core of binary threshold units computing $\Phi_{SvL,\mathcal{P}}$.*

Proof Assume that the input layer is actived at time t such that it represents an interpretation I. Then, at time $t+1$ an $h^\top$-unit representing $A \leftarrow B_1 \wedge \ldots \wedge B_k$ in the hidden layer becomes active iff all units representing $B_1, \ldots, B_n$ in the input layer are active, i.e., if $I(B_1) = \ldots = I(B_n) = \top$. Likewise, at time $t+1$ an $h^\perp$-unit representing $A \leftarrow B_1 \wedge \ldots \wedge B_k$ in the hidden layer becomes active iff one unit representing the negation of $B_1, \ldots, B_n$ in the input layer is active, i.e., if $I(\neg B_1) \vee \ldots \vee I(\neg B_n) = \top$. At time $t+2$ a unit representing A in the output later becomes active iff there is an active $h^\top$-unit representing $A \leftarrow B_1 \wedge \ldots \wedge B_k$ at time $t+1$. Likewise, at time $t+2$ a unit representing $\neg A$ in the ouput layer becomes active iff all $h^\perp$ units reprenting rules with head A are active at time $t+1$. Thus, the core is a direct encoding of $\Phi_{SvL,\mathcal{P}}$. $\square$

Given a program $\mathcal{P}$ and its core, then a recurrent network can be constructed by connecting each unit in the output layer to its corresponding unit in the input layer with weight 1.

Proposition 6. *For each program $\mathcal{P}$, the corresponding recurrent network initialized by the empty interpretation will converge to a stable state which corresponds to the least fixed point of $\Phi_{SvL,\mathcal{P}}$.*

Proof The result follows immediately from the construction of the recurrent network using Propositions 4 and 5. $\square$

The construction and the behavior of the networks is illustrated in Figure 1.

4 Human Reasoning

In this section we will discuss some examples taken from [3]. These examples were used by Byrne to show that classical logic cannot appropriately model human reasoning. Stenning and van Lambalgen argue that a three-valued logic programs under a completion semantics can well model human reasoning [17]. Moreover, as we will see, the core method presented in Section 3 serves as a connectionist model generator in these cases.

Consider the following sentences: *If Marian has an essay to write, she will study late in the library. She has an essay to write.* In [3] 96% of all subjects

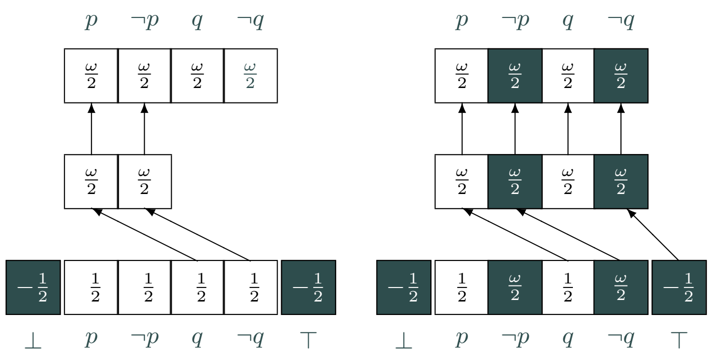

Figure 1. The stable states of the feed-forward cores for $\mathcal{P}_1$ (left) and $\mathcal{P}_2$ (right), where all connections have weight ω, active units are shown in grey and passive units in white. The recurrent connections between corresponding units in the output and input layer are not shown.

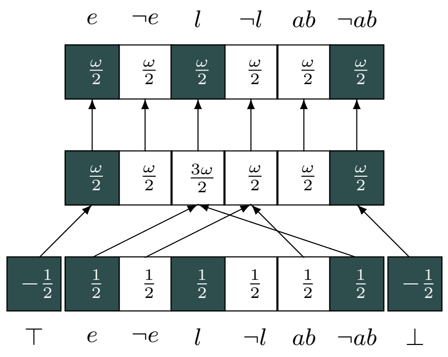

Figure 2. The stable state of the feed-forward core for $\mathcal{P}_3$.

conclude that *Marian will study late in the library.* The two sentences can be represented by the program $\mathcal{P}_3 = \{l \leftarrow e \wedge \neg ab,\ e \leftarrow \top,\ ab \leftarrow \bot\}$. The first sentence is interpreted as a licence for a conditional and the atom ab is used to cover all additional preconditions that we may be unaware of. As we know of no such preconditions, the rule $ab \leftarrow \bot$ is added. The corresponding network as well as its stable state are shown in Figure 2. From $lfp(\Phi_{SvL,\mathcal{P}_3}) = \langle\{l, e\}, \{ab\}\rangle$ follows that Marian will study late in the library.

Suppose now that the antecedent is denied: *If Marian has an essay to write, she will study late in the library. She does not have an essay to write.* In [3] 46% of subjects conclude that Marian will not study late in the library. These subject err with respect to classical logic. But they do not err with respect to the non-classical logic considered here. The two sentences can be represented by the program $\mathcal{P}_4 = \{l \leftarrow e \wedge \neg ab,\ e \leftarrow \bot, ab \leftarrow \bot\}$. The corresponding network as well as its stable state are shown in Figure 3. From $lfp(\Phi_{SvL,\mathcal{P}_4}) = \langle\emptyset, \{ab, e, l\}\rangle$ follows that Marian will not study late in the library.

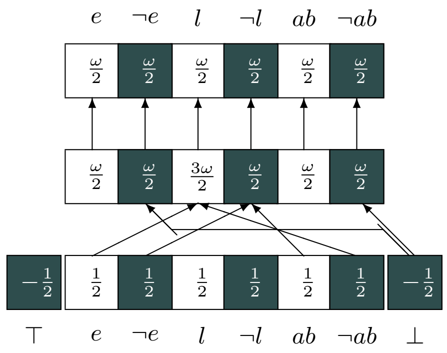

Figure 3. The stable state of feed-forward core for $\mathcal{P}_4$.

Now consider an alternative argument: *If Marian has an essay to write, she will study late in the library. She does not have an essay to write. If she has textbooks to read, she will study late in the library.* In [3] 4% of subjects conclude that Marian will not study late in the library. These sentences can be represented by $\mathcal{P}_5 = \{l \leftarrow e \wedge \neg ab_1,\ e \leftarrow \bot,\ ab_1 \leftarrow \bot,\ l \leftarrow t \wedge \neg ab_2, ab_2 \leftarrow \bot\}$. Due to lack of space we leave the construction of the network to the interested reader. From $lfp(\Phi_{SvL,\mathcal{P}_5}) = \langle \emptyset, \{ab_1, ab_2, e\}\rangle$ follows that it is unknown whether Marian will study late in the library. One should observe that $lfp(\Phi_{F,\mathcal{P}_5}) = \langle \emptyset, \{ab_1, ab_2, e, t, l\}\rangle$ and, consequently, one would conclude that Marian will not study late in the library. Thus, Fitting's operator leads to a wrong answer with respect to human reasoning, whereas Stenning and van Lambalgen's operator does not.

As final example consider the presence of an additional argument: *If Marian has an essay to write, she will study late in the library. She has an essay to write. If the library stays open, she will study late in the library.* In [3] 38% of subjects conclude that Marian will study late in the library. These sentences can be represented by $\mathcal{P}_6 = \{l \leftarrow e \wedge \neg ab_1,\ e \leftarrow \top,\ l \leftarrow o \wedge \neg ab_2,\ ab_1 \leftarrow \neg o,\ ab_2 \leftarrow \neg e, \}$. As argued in [17] the third sentence gives rise to an additinal argument for studying in the library, viz. that the library is open. Likewise, there must be a reason for going to the library like, for example, writing an essay. The corresponding network as well as its stable state are shown in Figure 4. From $lfp(\Phi_{SvL,\mathcal{P}_6}) = \langle \{e\}, \{ab_2\}\rangle$ follows that it is unknown whether Marian will study late in the library.

5 Discussion

We propose to use the Łukasiewicz semantics for three-valued logic programs in the area of human reasoning. Returning to our last example, $lfp(\Phi_{SvL,\mathcal{P}_6}) = \langle \{e\}, \{ab_2\}\rangle$ is a model for both, the weak completion of $\mathcal{P}_6$ and $\mathcal{P}_6$ itself, under the Łukasiewicz semantics, whereas it is – somewhat surprisingly – a model for the weak completion of $\mathcal{P}_6$ but not for $\mathcal{P}_6$ under the Fitting semantics. Under the

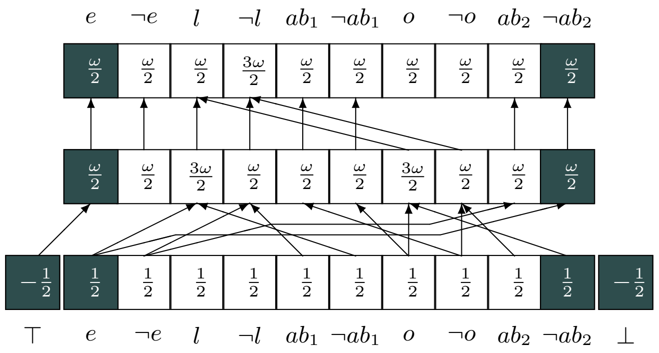

Figure 4. The stable state of feed-forward core for $\mathcal{P}_6$.

Łukasiewicz semantics the completion of a three-valued logic program is exactly what completion was originally thought of, viz., the addition of the only-if halves to a program specifying the if-halves [4]. Nevertheless, whether Łukasiewicz semantics is adequate for human reasoning in a broader sense remains to be seen.

We showed that the core method can be adapted to implement the revised immediate consequence operator of Stenning and van Lambalgen. We presented an algorithm for constructing the networks and proved that the networks settle down in a state encoding the least fixed point of the operator. Although our networks consist of logical threshold units, we claim that they can be replaced by bipolar sigmoidal ones while preserving the relationship to logic programs by applying the method first presented in [6] (see also [5]). The modified networks can then be trained using backpropagation or related techniques, rule extraction methods can be applied to the trained networks and the neural-symbolic cycle can be closed.

In our networks units come in pairs, where the first (second) element represents the fact that the corresponding variable or formula is mapped to true (false). In [17] similar networks are proposed – albeit in a three-dimensional setting – and, in addition, the elements of each pair inhibit each other. From a logical point of view such an inhibition is unnecessary – it can never be the case that both elements are active – as long as the units of the input layer are not externally activited. In a general setting, however, where context information is used to activate the input units, such inhibitory connections establishing a winner-take-all behaviour are very useful. Unfortunately, the presented theoretical results concerning the core networks and their relation to logic programs do not apply in this case anymore unless we would be able to show that $\Phi_{SvL,\mathcal{P}}$ is a contraction.

In [17] additional techniques like integrity constraints and abduction are suggested to handle additional human reasoning tasks. At the moment, we do not know how to map these into the core method.

In the field of Logic Programming Fitting's three-valued (first-order) logic has been used in termination analysis. It remains to be seen whether these results carry over to Łukasiewicz semantics. How important is Stenning and van Lambalgen's operator from a logic programming perspective?

References

1. S. Bader, P. Hitzler, S. Hölldobler, and A. Witzel. A fully connectionist model generator for covered first-order logic programs. In Manuela M. Veloso, editor, *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pages 666–671, Menlo Park CA, January 2007. AAAI Press.
2. S. Bader and S. Hölldobler. The core method: Connectionist model generation. In *Proceedings of the 16th International Conference on Artificial Neural Networks (ICANN)*, volume 4132 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2006.
3. R.M.J. Byrne. Suppressing valid inferences with conditionals. *Cognition*, 31:61–83, 1989.
4. K. L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 293–322. Plenum, New York, 1978.
5. A.S. d'Avila Garcez, K. Broda, and D.M. Gabbay. *Neural-Symbolic Learning Systems: Foundations and Applications*. Springer, 2002.
6. A.S. d'Avila Garcez, G. Zaverucha, and L.A.V. de Carvalho. Logic programming and inductive learning in artificial neural networks. In Ch. Herrmann, F. Reine, and A. Strohmaier, editors, *Knowledge Representation in Neural Networks*, pages 33–46, Berlin, 1997. Logos Verlag.
7. M. Fitting. A Kripke–Kleene semantics for logic programs. *Journal of Logic Programming*, 2(4):295–312, 1985.
8. M. Fitting. Metric methods – three examples and a theorem. *Journal of Logic Programming*, 21(3):113–127, 1994.
9. K.-I. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2:183–192, 1989.
10. S. Hölldobler and Y. Kalinke. Towards a massively parallel computational model for logic programming. In *Proceedings of the ECAI94 Workshop on Combining Symbolic and Connectionist Processing*, pages 68–77. ECCAI, 1994.
11. S. Hölldobler, Y. Kalinke, and H.-P. Störr. Approximating the semantics of logic programs by recurrent neural networks. *Applied Intelligence*, 11:45–59, 1999.
12. K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.
13. Y. Kalinke. Ein massiv paralleles Berechnungsmodell für normale logische Programme. Master's thesis, TU Dresden, Fakultät Informatik, 1994. (in German).
14. S. C. Kleene. *Introduction to Metamathematics*. North-Holland, 1952.
15. J. Łukasiewicz. O logice trójwartościowej. *Ruch Filozoficzny*, 5:169–171, 1920. English translation: On Three-Valued Logic. In: *Jan Łukasiewicz Selected Works.* (L. Borkowski, ed.), North Holland, 87-88, 1990.
16. A.K. Seda and M. Lane. Some aspects of the integration of connectionist and logic-based systems. In *Proceedings of the Third International Conference on Information*, pages 297–300, International Information Institute, Tokyo, Japan, 2004.
17. K. Stenning and M. van Lambalgen. *Human Reasoning and Cognitive Science.* MIT Press, 2008.
18. J. E. Stoy. *Denotational Semantics.* MIT Press, Cambridge, 1977.